



Julie C. Meloni
Michael Morrison

Tanuljuk meg a HTML és a CSS használatát 24 óra alatt

A fordítás a következő angol eredeti alapján készült:

Julie C. Meloni, Michael Morrison: SAMS Teach Yourself HTML and CSS in 24 Hours

Authorized translation from the English language edition, entitled SAMS TEACH YOURSELF HTML AND CSS IN 24 HOURS, 1st Edition, ISBN 0672330970, by MELONI, JULIE C.; MORRISON, MICHAEL, published by Pearson Education, Inc., publishing as SAMS Publishing.

Copyright © 2010 by SAMS Publishing.

Translation and Hungarian edition © 2011 Kiskapu Kft.

All rights reserved. No part of this book, including interior design, cover design, and icons, may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording, or otherwise) without the prior written permission of the publisher.

Trademarked names appear throughout this book. Rather than list the names and entities that own the trademarks or insert a trademark symbol with each mention of the trademarked name, the publisher states that it is using the names for editorial purposes only and to the benefit of the trademark owner, with no intention of infringing upon that trademark.

Fordítás és magyar változat © 2011 Kiskapu Kft. Minden jog fenntartva!

A könyv egyetlen része sem sokszorosítható semmilyen módszerrel a Kiadó előzetes írásos engedélye nélkül. Ez a korlátozás kiterjed a belső tervezésre, a borítóra és az ikonokra is. A könyvben bejegyzett védjegyek és márkanevek is felbukkanhatnak. Ahelyett, hogy ezt minden egyes helyen külön jeleznénk, a Kiadó ezennel kijelenti, hogy a műben előforduló valamennyi védett nevet és jelzést szerkesztési célokra, jóhiszeműen, a név tulajdonosának érdekeit szem előtt tartva használja, és nem áll szándékában az azokkal kapcsolatos jogokat megszegni, vagy kétségbe vonni.

A szerzők és a kiadó a lehető legnagyobb körültekintéssel járt el a kiadvány elkészítésekor. Sem a szerző, sem a kiadó nem vállal semminemű felelősséget vagy garanciát a könyv tartalmával, teljességével kapcsolatban. Sem a szerző, sem a kiadó nem vonható felelősségre bármilyen baleset vagy káresemény miatt, mely közvetve vagy közvetlenül kapcsolatba hozható a kiadvánnyal.

Sorozatszerkesztő: *Szy György*

Lektor: *Rézműves László*

Fordítás: *Gilicze Bálint, Rézműves László, Varga Péter*

Műszaki szerkesztő: *Csutak Hoffmann Levente*

Tördelés: *Kis Péter*

Felelős kiadó a Kiskapu Kft. ügyvezető igazgatója

© 2011 Kiskapu Kft.

1134 Budapest, Csángó u. 8.

Fax: (+36-1) 303-1619

<http://www.kiskapukiado.hu/>

e-mail: kiado@kiskapu.hu

ISBN: 978 963 9637 78 8

Nyomdai előállítás: *Radin Group*

Felelős vezető: *Antun Basic*

Kereskedelmi képviselő: *Kvadrát '97 Kft.*

Tel./Fax: +361 319 1599. **Mobil:** +36 30 280 6656

E-mail: kvadrat97@gmail.com

Tartalomjegyzék

1. óra • A Web működése	1
A HTML és a Világháló rövid története	2
A webes tartalmak létrehozása	3
A webes tartalmak kézbesítésének módja	4
A webgazda kiválasztása	6
Tesztelés több webböngészőben	8
Összefoglalás.	9
2. óra • A webes tartalom közzététele	13
Az óra példafájljának elkészítése	13
Fájlok átvitele az FTP segítségével	14
A fájlok helye a webkiszolgálón.	18
A tartalom közzététele webkiszolgáló nélkül	22
A webes tartalom ellenőrzése.	24
Összefoglalás.	25
3. óra • A HTML és az XHTML alapjai	27
Az egyszerű weboldalak készítésének alapjai	29
Az XHTML-oldalakon kötelezően felhasználandó HTML-elemek	33
Oldalszerkezet kialakítása bekezdésekkel és sortörésekkel	35
A tartalom tagolása címsorok segítségével	38
A webes tartalom érvényességének vizsgálata	40
HTML, XML, XHTML és HTML 5 – útmutató a szabványok dzsungeléhez	42
Összefoglalás.	44
4. óra • A stíluslapok világa	49
A CSS működése	50
Egyszerű stíluslap létrehozása	52
A CSS-stílusok használatának alapjai	57
Stílusosztályok használata.	61
Stílusazonosítók használata	63
Belső stíluslapok és kódon belüli stílusok.	64
Összefoglalás.	67
5. óra • Szövegblokkok és listák	71
Szöveg igazítása az oldalon belül.	72
A HTML három listatípusa	76
Listák a listákban.	78
Összefoglalás.	83

6. óra • Betűformázás	87
Félkövér, dőlt és egyéb betűalakok	88
Haladó betűbeállítások	92
Különleges karakterek használata	96
Összefoglalás	99
7. óra • Információk megjelenítése táblázatok segítségével	103
Egyszerű táblázat létrehozása	104
Táblázat méretének beállítása	107
Tartalom igazítása és átívelése a táblázaton belül	110
Oldalelrendezés kialakítása táblázatokkal	113
Összefoglalás	114
8. óra • Külső és belső hivatkozások használata	119
Webcímek használata	120
Oldalon belüli hivatkozások horgonyokkal	122
Hivatkozások a saját oldalaink között	126
Hivatkozás külső webes tartalomra	129
Hivatkozás elektronikus levélcímre	130
Hivatkozás megnyitása új böngészőablakban	132
Hivatkozások formázása CSS-stílusokkal	132
Összefoglalás	136
9. óra • A színek használata	141
A színválasztás fogásai	142
Webszínek	143
Hexadecimális színek kódok	145
A háttér, a szöveg és a szegélyek színének beállítása CSS-kódokkal	147
Összefoglalás	149
10. óra • Képek készítése webes használatra	153
A grafikai alkalmazás kiválasztása	154
Amit a grafikáról mindenképp tudnunk kell	155
A fényképek előkészítése	156
Reklámcsíkok és nyomógombok készítése	162
Hogyan csökkenthető a színek száma egy képen?	164
Átlátszó képek használata	165
Mozaikos háttér kialakítása	166
Animációt tartalmazó webes grafika készítése	168
Összefoglalás	168
11. óra • Képek használata a webhelyen	173
Képek elhelyezése egy weboldalon	174
Képek szöveges leírása	177
A kép szélességének és magasságának megadása	178

Képek igazítása	179
Képek átalakítása hivatkozássá	183
Háttérképek használata	186
Képtérképek készítése	188
Összefoglalás	194

12. óra • Multimédia a webhelyen 199

Hivatkozás multimédiafájlokra	201
Multimédiafájlok beágyazása	204
További ötletek a multimédia használatához	207
Összefoglalás	209

13. óra • Keretek használata 213

Mik azok a keretek?	214
Keretváz kialakítása	216
Hivatkozások keretek és ablakok között	219
Belső keretek használata	221
Összefoglalás	223

14. óra • A külső és a belső margó, az igazítások és az úsztatás használata 229

Margók használata	231
Az elemek belső margója	238
Elemek igazítása	242
A float tulajdonság	243
Összefoglalás	246

15. óra • A CSS dobozmodellje és az elemek elhelyezése 249

A CSS dobozmodellje	250
Nagykanállal az elhelyezésről	253
Az egymás tetejére kerülő elemek elhelyezkedésének szabályozása	258
A szöveg folyásirányának szabályozása	260
Összefoglalás	261

16. óra • Szemrevalóbb listák és egyébek – a CSS használatával 265

Ismétlés – a HTML listái	266
A CSS-dobozmodell hatása a listákra	267
A felsorolásjelek elhelyezése	270
Képtérképek készítése listaelemek és CSS használatával	272
Összefoglalás	276

17. óra • Navigációs felület kialakítása a CSS segítségével 279

Miben különböznek a navigációs listák a hagyományos társaiktól?	280
Függőleges navigációs sáv készítése CSS-kód segítségével	281
Vízszintes navigációs sáv készítése CSS-kód segítségével	290
Összefoglalás	294

18. óra • Szövegmódosítás egérműveletekkel	297
Lebegő leírások létrehozása CSS segítségével	298
Egyéb szövegrészletek megjelenítése CSS-kód segítségével	301
Események elérése	303
Összefoglalás.	310
19. óra • Rögzített és folyékonyelrendezések létrehozása	313
Rögzített elrendezések	314
Folyékony elrendezések	316
Rögzített-folyékony kevert elrendezések	319
Összefoglalás.	329
20. óra • Nyomtatóbarát weboldalak készítése	331
Mi teszi nyomtatóbaráttá a weboldalakat?	332
A megjelenési formához illő stíluslap használata	335
A nyomtatható oldalakhoz való stíluslap elkészítése	337
Az oldalak nyomtatási előnézetének megtekintése	341
Összefoglalás.	342
21. óra • Dinamikus webhelyek	345
A parancsfájlok különféle típusai	346
JavaScript-kód a HTML nyelvű oldalakon	347
Véletlenszerű tartalom megjelenítése	349
A dokumentum-objektummodell	352
Képváltás felhasználói műveletek alapján	353
Összefoglalás.	355
22. óra • Webes űrlapok	357
A HTML-űrlapok működése	358
Űrlap létrehozása	359
Szöveges adatok fogadása	363
Az űrlapelemek elnevezése	364
Rejtett adatok az űrlapokon	364
Az űrlapok beviteli vezérlői	365
Az űrlapadatok elküldése	369
Összefoglalás.	369
23. óra • Webhelyek összeállítása és kezelése	373
Amikor egy oldal is elég.	374
Egyszerű webhely felépítése	376
Nagyobb webhely összeállítása	380
Fenntartható HTML-kód készítése	383
Összefoglalás.	386

24. óra • Segítség a kereséshez 389

A webhelyünk népszerűsítése	390
Weboldalaink felvétele a nagyobb keresők adatbázisaiba	391
Tippek a keresőknek	393
A keresésoptimalizálás további fogásai	399
Összefoglalás.	400

A függelék • HTML- és CSS-források az Interneten 403

Általános információk a HTML-lel, az XHTML-lel és a CSS-sel kapcsolatban . . .	403
Webböngészők	404
Weboldalak tervezése	404
Szoftver	404
Színek és grafika	405
Multimédia.	405
Fejlesztői források haladóknak.	406
Bérelt webtárhely-szolgáltatás	406
Webhely-üzemeltetési szolgáltatások.	406

B függelék • XHTML 1.1 és CSS 2 gyorstalpaló 407

Szerkezeti XHTML-elemek	408
XHTML-szövegtömbök és -bekezdések	411
Szövegformázó XHTML-elemek	413
XHTML-listák.	414
XHTML-hivatkozások	415
XHTML-táblázatok.	416
Beágyazott tartalmat szabályozó XHTML-elemek	422
XHTML-stílusok.	424
XHTML-űrlapok.	424
XHTML-parancskódok	427
Általános XHTML-jellemzők	427
Méretbeállító CSS-stílustulajdonságok	428
Szöveg- és betűstílus-beállító CSS-tulajdonságok	429
Háttérbeállító CSS-stílustulajdonságok.	431
Szegélybeállító CSS-stílustulajdonságok.	432
Margóbeállító CSS-stílustulajdonságok.	434
Kitöltésbeállító CSS-stílustulajdonságok.	435
Elrendezési és megjelenítési CSS-stílustulajdonságok	435
Lista- és felsorolásijel-beállító CSS-stílustulajdonságok	437
Táblázatformázó CSS-stílustulajdonságok	438

Tárgymutató 439

A szerzőkről

Julie C. Meloni az i2i nevű, a kaliforniai Los Altosban működő multimédia-vállalat műszaki igazgatója, és emellett a digitális humán tudományok szakértője, akinek a webes programozási nyelvek és az adatbázisok témakörében számos könyve és cikke jelent meg, köztük a sikeres *Sams Teach Yourself PHP*, a *Sams Teach Yourself MySQL* vagy az *Apache All in One*.

Michael Morrison íróként, programfejlesztőként, játékfeltalálóként és különféle számítógép-tudományi könyvek és interaktív webes tanfolyamok szerzőjeként tevékenykedik. Elsődleges szakmája az írás, de szabadúszó szakértőként bárkinek a rendelkezésére áll, és a feleségével, Masheed-del alapított Stalefish Labs szórakoztatóipari cég kreatív vezetését is magára vállalta.

Mondja el a véleményét!

Az olvasó a legfontosabb kritikus, akinek a véleménye nekem és a kiadónak is roppant értékes. Szeretnénk tudni, mit csinálunk jól, mi az, amin javíthatnánk, milyen könyveket kellene megjelentetnünk és így tovább. Felhívjuk a figyelmet, hogy a könyv témájával kapcsolatos szakmai kérdésekben nem tudunk segíteni, és a nagy számban beérkező levelek miatt nem biztos, hogy minden üzenetre válaszolunk.

Kérjük, ha ír, tüntesse fel a könyv szerzőjét és címét, valamint a saját nevét, telefonszámát vagy e-mail címét. Megjegyzéseit alaposan áttanulmányozzuk, és továbbítjuk a könyv szerzőjének és szerkesztőinek.

E-mail: webdev@sampublishing.com

Levél: Mark Taber
Associate Publisher
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Olvasószolgálat

A webhelyünket meglátogatva az informit.com/register címen lehet bejegyeztetni a könyvet, ezáltal könnyen hozzáférhetünk a könyvvel kapcsolatos frissítésekhez, le-töltésekhez, illetve hibajegyzékekhez.

Bevezetés

A 2009-es adatok szerint több mint 1,5 milliárd ember rendelkezik interneteléréssel – csak az Amerikai Egyesült Államokban 220 millióan. Adjuk hozzá a 338 millió kínai, 55 millió német, 48 millió brit, 38 millió orosz és 67 millió brazil felhasználót, és máris láthatjuk, mit jelent a „világ” szó a Világháló kifejezésben. Az internetfelhasználók jelentős része egyben alakítja is a Web tartalmát – ha szeretnénk, mi is közéjük tartozhatunk. A weboldalak számáról ugyan nemigen áll rendelkezésre pontos felmérés, de a Google legfrissebb adatai szerint az indexelt oldalak mennyisége 2008 közepén átlépte az 1 trillió határt.

A következő 24 órában új oldalak százmilliói fognak megjelenni az Internet nyilvánosan elérhető részén, és legalább ugyanennyit helyeznek majd el belső magánhálózaton a helyi hálózatot használó üzletemberek számára. Az említett oldalak mindegyike – a már hozzáférhető 1 trillió oldalhoz hasonlóan – a hiperszöveges jelölőnyelvet, röviden HTML-t (Hypertext Markup Language) használja.

Ha elvégezzük az ebben a kötetben található 24 egyórás leckét, mi is közzétehetjük a weboldalainkat az Interneten. Ezek a leckék emellett abban is segítenek, hogy elsajátítsuk a ma a világon legfontosabb készséget: a HTML használatát. De valóban meg lehet tanulni a csúcsmínőségű weboldalak önálló elkészítését különleges szoftver nélkül és kevesebb idő alatt, mint amennyi ahhoz kell, hogy időpontot egyeztessünk egy jól fizetett HTML-varázslóval? Képes lehet ez a viszonylag rövid, olvasmányos kötet felruházni minket azzal a képességgel, hogy önállóan megtanuljuk a weboldalak közzétételének legújabb módszereit?

A válasz: igen. Valójában már a könyv első két leckéje elég ahhoz, hogy az is, aki egyáltalán nem rendelkezik HTML-előismeretekkel, közzé tudjon tenni egy weboldalt a Weben. De hogyan tanulhatjuk meg a Web nyelvét ilyen gyorsan? Nos, példákon keresztül. Ez a könyv a HTML-ismereteket egyszerű lépések sorozatára bontja, és bemutatja, hogyan birkózhatunk meg ezekkel a lépésekkel. Minden elkészítendő weboldal képe előtt közvetlenül ott szerepel a szükséges HTML-kód – láthatjuk, hogyan kell megírni; tömör, világos magyarázatot kapunk a működéséről; és rögtön lehetőséget kapunk arra is, hogy a tanultakat a saját weboldalunkon alkalmazzuk. Tíz perccel később pedig már jöhet is a következő lépés.

24 órányi munka, és el fogunk csodálkozni, milyen hatásos oldalakat tudunk készíteni az Internetre.

A HTML-en túl

Ez a könyv nem csupán a HTML-lel foglalkozik, mert nem a HTML az egyetlen, amit ma ismernünk kell ahhoz, hogy webes tartalmat hozzunk létre. Kötetünk célja az, hogy az Olvasót ellássa mindazokkal a készségekkel, amelyek a modern, szabványkövető webhelyek készítéséhez szükségesek, még hozzá mindössze 24 rövid, egyszerű leckén keresztül. A könyvben a következő kulcsfontosságú témakörökkel és technológiákkal foglalkozunk:

- Az XHTML (eXtensible Hypertext Markup Language, bővíthető hiperszöveges jelölőnyelv) a weboldalak létrehozásának jelenlegi szabványa. A kötet minden példája teljes mértékben XHTML-megfelelő, és ahol csak lehet, a HTML 5-tel is foglalkozunk.
- A könyv minden példáját ellenőriztük, hogy megfelel-e az összes főbb böngésző legújabb változatának, beleértve az Apple Safari, a Google Chrome, a Microsoft Internet Explorer, a Mozilla Firefox és az Opera böngészőket. Már a kezdetektől megtanuljuk, hogyan egyeztessük össze az oldalainkat a múlttal, miközben felkészítjük őket a jövőre is.
- Kimerítően tárgyaljuk a többszintű stíluslapokat (CSS, Cascading Style Sheets), amelyek lehetővé teszik, hogy a weboldalaink elrendezésének, betűtípusainak, színeinek és formázásának minden részletét kézben tarthassuk. Ha valóban ámulatba ejtő weboldalakat szeretnénk készíteni, a CSS-sel sokkal messzebbre juthatunk, mint amit pusztán hagyományos HTML-oldalakkal elérhetünk. Tudtuk például, hogy a CSS azt is lehetővé teszi, hogy egy oldal tartalmát kifejezetten nyomtatáshoz szabjuk a szokványos webes megjelenítés mellett?
- A 10–12. leckék a multimédia-alkalmazások világába nyújtanak bevezetést, ismertetve többek között, hogy hol találhatunk olyan, szabványnak számító programokat, amelyeket szabadon letölthetünk és kipróbálhatunk.
- A technikai részletek ismerete önmagában nem elegendő, ezért a kötet arra nézve is tanácsokkal lát el minket, hogy miként alakíthatjuk ki a webhelyeinket úgy, hogy elérjük a célunkat. A legfontosabb témákat – hatékony oldal-elrendezések kialakítása, az oldalak közzététele az Interneten FTP-program segítségével, az oldalak rendszerezése és kezelése, illetve előkelő hely biztosítása az oldalaknak a főbb internetes keresők találatai között – kivétel nélkül kellő mélységben tárgyaljuk ahhoz, hogy túllendülhessünk a kezdeti nehézségeken.

Az említett létfontosságú ismeretekre fordított figyelem tette e kötet hét korábbi angol kiadását sikerkönyvvé, és ez a frissített (az angol kiadást tekintve nyolcadik) változat sem adja alább. Minden példát frissítettünk, és a tartalom jelentős részét átdolgoztuk, hogy az új technológiáknak is helyet adjunk.

Vizuális példák

A kötet minden példáját kétféle szemszögből mutatjuk be:

- Először az adott HTML-oldal elkészítéséhez beírandó szöveget láthatjuk a HTML- és CSS-kódelemekkel együtt.
- Ezt követően szerepel az eredményként kapott weboldal, úgy, ahogy a felhasználók látják majd a legnépszerűbb webböngészőkben.

A példákat gyakran anélkül is a saját oldalainkhoz igazíthatjuk, hogy elolvassánk a hozzájuk tartozó szöveget.

A könyv valamennyi példája szabványkövető, és működik az Apple Safari, a Google Chrome, a Microsoft Internet Explorer, a Mozilla Firefox és az Opera böngészőkben. A képernyőképeket ugyan a Firefoxban készítettük, de nyugodjunk meg: minden kódot kipróbáltunk a többi böngészőben is.

Különleges elemek

A leckék során különféle széljegyzetek segítenek, hogy a tanultakat rögtön alkalmazhassuk is a saját weboldalainkra:



A feltűnő keretben elhelyezett tippek olyan tanácsokat adnak, amelyeket megfogadva értékes időt takaríthatunk meg.



A megjegyzések bővebb információkat nyújtanak az adott témáról.



Ha valamire különösen ügyelnünk kell, arra egy keretes Figyelmeztetésben hívjuk fel a figyelmet.

Kérdezz-felelek, ismétlő kérdések és gyakorlatok

Minden óra végén egy rövid kérdezz-felelek részt találhatunk, amelyben azokra a „buta kérdésekre” kell válaszolnunk, amelyeket mindenki szeretne feltenni, de senki sem mer. Ezt követően ugyancsak rövid, de kimerítő ismétlő kérdések következnek, amelyeknek a segítségével ellenőrizhetjük, hogy mennyire sikerült megértenünk a fejezetben tanultakat. A leckét végül egy vagy több nem kötelező gyakorlat zárja, amelyek lehetőséget adnak az újonnan elsajátított készségek gyakorlására, mielőtt továbblépnénk.



1. ÓRA

A Web működése

A lecke tartalma:

- A Világháló története röviden
- Mit jelent a „weboldal” kifejezés, és miért nem tükrözi mindig az összes kapcsolódó tartalmat?
- Hogyan jut el a tartalom a személyi számítógépünkről mások webböngészőjébe?
- Hogyan válasszunk webgazdát?
- Milyen hatással vannak a különféle webböngészők és eszköztípusok a tartalomra?

Mielőtt megtanulhatnánk a HTML és a CSS finomságait, fontos, hogy szilárd tudással rendelkezünk azokról a technológiákról, amelyek segítenek ezeket a sima szöveges fájlokat gazdag multimédiás tartalommal alakítani a számítógépünkön vagy egy mobileszközön, amikor a Világhálón böngészünk.

Egy HTML- és CSS-kódot tartalmazó fájl semmit nem ér egy webböngésző nélkül, amellyel megtekinthetjük, és rajtunk kívül senki nem fogja látni az általunk készített tartalmat, ha nem gondoskodik a közzétételéről egy webkiszolgálón. A webkiszolgálók teszik az általunk kínált tartalmat azok számára elérhetővé, akik a webböngészőikkel ellátogatnak egy címre, és várják, hogy a kiszolgáló információkat küldjön nekik.

A folyamatnak mi is részesei vagyunk, mivel nekünk kell elkészítenünk a fájlokat, és elhelyeznünk azokat egy kiszolgálón, hogy hozzáférhetővé váljanak, és nekünk kell gondoskodnunk róla, hogy a tartalom úgy jelenjen meg a végfelhasználók szeme előtt, ahogy szerettük volna.

A HTML és a Világháló rövid története

Réges-régen, amikor még nem voltak emberi lábnyomok sem a Holdon, élt néhány jövőbe látó ember, aki kíváncsi volt rá, hogy össze tud-e kötni néhány nagyobb számítógépes hálózatot. A nevektől és az anekdotáktól most megkímélnénk az Olvasót (pedig mindkettőből van jónéhány) – elég annyi, hogy a tevékenységük eredménye végül „minden hálózat anyja” lett: az, amit ma Internet néven ismerünk.

1990-ig az információk elérése az Interneten keresztül meglehetősen műszaki jellegű feladat volt. Valójában annyira nehéz feladatnak bizonyult, hogy még a doktori címmel rendelkező fizikusok is gyakran akadályokba ütköztek, amikor megpróbálták adatokat cserélni. Az egyik ilyen fizikus, a ma már világhírű (és azóta lovaggá ütött) Sir Tim Berners-Lee, ezért kitalált egy módszert arra, hogy egyszerűen lehessen kereszthivatkozásokat létrehozni az Interneten található szövegek között: a „hiperszöveges” hivatkozások, röviden hipervivatkozások rendszerét.

Az ötlet nem volt új, de Berners-Lee egyszerű hiperszöveges jelölő- vagy leírónyelve (Hypertext Markup Language, HTML) fennmaradt, miközben a nagyratörőbb hiperszöveges megoldások elbuktak. A *hiperszöveg* eredetileg olyan, elektronikus formában tárolt szöveget jelentett, amely kereszthivatkozásokkal kapcsolt össze oldalakat. A kifejezést ma már szélesebb értelemben használják, és szinte bármilyen objektumra (szövegre, képekre és más fájlokra) vonatkozhat, amelyet más objektumokhoz kapcsolhatunk. A *hiperszöveges jelölőnyelv* írja le, hogy a szövegek, grafikák és más információkat tartalmazó fájlok hogyan épülnek fel és kapcsolódnak egymáshoz.



Ha többet szeretnénk tudni a Világháló történetéről, olvassuk el a Wikipedia cikkét a témáról a http://en.wikipedia.org/wiki/History_of_the_Web címen.

1993-ban a világon csupán mintegy 100 számítógép volt képes HTML-oldalak kiszolgálására. Ezeket az összekapcsolt oldalakat *World Wide Webnek* (WWW, Világháló) nevezték, és számos webböngésző programot írtak, hogy lehetővé tegyék a felhasználóknak a weboldalak megtekintését. A Web növekvő népszerűsége miatt néhány programozó hamarosan olyan webböngészőket kezdett készíteni, amelyek a szövegek mellett képek megjelenítésére is képesek voltak. Innen kezdve a webböngésző programok fejlődése és a HTML – illetve az XHTML – nyelv szabványosítása elvezetett ahhoz a világhoz, amelyben ma élünk, és ahol több mint 110 millió webkiszolgáló válaszol a 25 milliárdnál is több szöveges és multimédia-állományra irányuló kérélmekre.

Ebben a néhány bekezdésben csupán nagyon röviden foglalhattuk össze a történelemnek azt a szeletét, amelyben figyelemre méltó változások zajlottak le. A mai középiskolások már soha nem éltek olyan időkben, amikor a Világháló nem létezett, és az „azonnal hozzáférhető” információk és mindenütt jelen levő számítógépek az életünk minden területét áthatják. Mire ennek a könyvnek a végére érünk, a webes tartalmak létrehozását és kezelését nem olyan készségnek fogjuk tekinteni, amelyet csak néhány műszaki érdeklődésű személy (hívjuk őket „gyíkoknak” vagy „kockafejeknek”) birtokolhat, hanem olyan tudásnak, amelyet bárki elsajátíthat, függetlenül attól, hogy mennyire „kocka” legbelül.

A webes tartalmak létrehozása

Talán felfigyeltünk rá, hogy a „weboldalak” helyett a „webes tartalom” kifejezést használtuk – szándékosan. Bár azt mondjuk, hogy „felkeresünk egy weboldalt”, valójában azt értjük alatta, hogy „megtekintjük azokat a szövegeket és képeket a számítógépünkön, amelyek egy adott webcímen találhatók”. A szöveget, amelyet olvasunk, és a képeket, amelyeket látunk, a webböngészőnk állítja elő az egyes fájlokban található utasítások alapján.

Az említett fájlok jelölő- vagy *leírókóddal* ellátott szöveget tartalmaznak; ezek a HTML-kódok mondják meg a böngészőnek, hogy miként kell megjeleníteni a szöveget – címsorként, bekezdésként, piros betűkkel, és így tovább. Egyes HTML-leírókódok arra utasítják a böngészőt, hogy egyszerű szöveg helyett egy képet vagy egy mozgóképes állományt jelenítsen meg, és ezzel máris visszakanyarodtunk a lényeghez: a webböngészőnk különféle típusú tartalmakat kap, ezért a „weboldal” kifejezés nem fed le megfelelően a tartalmat. Helyette így a „webes tartalom” kifejezést használjuk, amelybe az Interneten található szövegek, képek, illetve hang-, videó- és egyéb multimédia-fájlok teljes körét beleértjük.

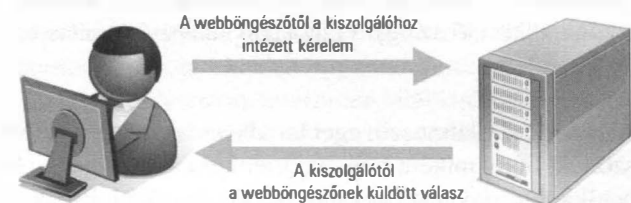
A későbbi leckékben elsajátítjuk az alapjait annak, hogy miként kapcsolhatjuk össze vagy hozhatjuk létre a különféle multimédiás webes tartalmakat, amelyek a webhelyeken találhatók. Most csak annyit kell megjegyeznünk, hogy *mi* tartjuk kézben a tartalmat, amelyet a felhasználó lát, amikor felkeresi a webhelyünket. A megjelenítendő szöveget, illetve a kiszolgált képek elküldésére utasító kódokat tartalmazó fájlal kezdve nekünk kell megterveznünk és elkészítenünk minden alkotórészt, amelyből végül felépül a webes jelenlétünk. Ahogy a könyv során megtanuljuk majd, mindez nem bonyolult, ha értjük az egyes lépéseket.

A webes tartalom alapvetően egy egyszerű szövegfájlal kezdődik, amely HTML- vagy XHTML-leírókódot tartalmaz. Az XHTML a HTML egyik változata – az „X” az eXtensible, vagyis „bővíthető” rövidítése; az XHTML-ről a leckék során többet is megtudunk majd. Most elég annyit tudnunk, hogy a kötet minden példája megfelel a HTML 4 és az XHTML követelményeinek, ami azt jelenti, hogy a mai webböngészők és a jövőbeli,

következő generációs böngészők ugyanúgy fogják leképezni őket. Ez az egyik előnye annak, ha szabványkövető kódot írunk: nem kell aggódnunk amiatt, hogy a jövőben vissza kell térnünk a kódhoz, hogy módosítsuk, mert „nem működik”. A kódunk valószínűleg mindig „működni” fog, legalábbis amíg a webböngészők tartják magukat a szabványokhoz (vagyis remélhetőleg hosszú-hosszú ideig).

A webes tartalmak kézbesítésének módja

A végeredményként megjelenített webes tartalom előállítására különféle, más-más helyeken zajló műveleteket igényel. Ezek a műveletek nagyon gyorsan – ezredmásodpercek alatt – hajtnak végre, a színfalak mögött. Más szavakkal, miközben azt hisszük, hogy csupán megnyitunk egy webböngészőt, beírunk egy webcímet, és máris látjuk a kért tartalmat, a háttérben működő technológia keményen dolgozik értünk. A böngésző és a kiszolgáló közötti alapszintű kommunikációt az 1.1. ábra szemlélteti.



1.1. ábra

A böngészőtől érkező kérelem és a kiszolgálótól kapott válasz

Az ábrán látható folyamat mindazonáltal több lépésből áll, és akár több kört is megtehet a böngésző és a kiszolgáló között, mielőtt a kért webhely tartalma teljes egészében megjelenne.

Tegyük fel, hogy egy keresést szeretnénk végrehajtani a Google-ben. Kötelességtudóan beírjuk hát a `http://www.google.com` címet a címsávba, vagy kiválasztjuk a Google könyvjelzőjét a könyvjelzőlistánkból. A böngésző szinte azonnal megjeleníti az 1.2. ábrán láthatóhoz hasonló tartalmat.

Az 1.2. ábrán egy olyan webhelyet láthatunk, amely szöveget, valamint egy képet (a Google emblémáját) tartalmaz. Az említett szövegnek és képnek a webkiszolgálóról történő lehívásához, illetve a képernyőnkön való megjelenítéséhez szükséges műveletek leegyszerűsítve a következők:

1. A webböngésző kiad egy kérelmet a `http://www.google.com/` címen található `index.html` fájlra. Az `index.html` fájlnevet nem kell beírunk a cím részeként a címsávba (az `index.html` fájlról a 2. órán bővebben is tanulunk).
2. Amikor a webkiszolgálói folyamat egy adott fájlra irányuló kérelmet kap, megkeresi a fájlt a könyvtári tartalomjegyzékében, megnyitja, és elküldi a tartalmát a webböngészőnek.



1.2. ábra

A *www.google.com* weboldala

3. A webböngésző megkapja az `index.html` fájl tartalmát, amely HTML-kódokkal jelölt szövegből áll, és az említett HTML-kódok alapján leképezi a tartalmat. A tartalom leképezése során a böngésző ráakad az 1.2. ábrán látható Google-embléma HTML-kódjára. Ez a HTML-kód így fest:

```

```

A fenti címke olyan jellemzőket tartalmaz, amelyek elárulják a böngészőnek a fájl forráshelyét (`src`), a kép szélességét (`width`) és magasságát (`height`), a kép keretének (szegélyének) típusát (`border`), valamint a kép helyettesítő szövegét (`alt`), amelyek az embléma megjelenítéséhez szükségesek. A jellemzőkről a későbbi leckékben bővebben is tanulunk.

4. A böngésző megvizsgálja az `` elem `src` jellemzőjét, hogy megtudja, hol kell keresnie a képfájlt. Esetünkben a `logo.gif` kép a logos könyvtárban található ugyanazon a webcímen (`www.google.com`), mint ahonnan a böngésző lekérte a HTML-fájlt.
5. A böngésző elkéri a `http://www.google.com/logos/logo.gif` webcímen található fájlt.
6. A webkiszolgáló feldolgozza a kérelmet, megkeresi a fájlt, és elküldi a tartalmát a kérelmező webböngészőnek.
7. A webböngésző megjeleníti a képet a monitorunkon.

Ahogy a webes tartalom kézbesítési folyamatának leírásából láthatjuk, a webböngészők nem pusztán képkeretek, amelyekben tartalmat tekinthetünk meg. A böngésző állítja össze a webes tartalom összetevőit, és rendezi el azokat a fájlban található HTML-parancsoknak megfelelően.

A webes tartalmat „helyben”, vagyis a saját merevlemezünkről is megtekinthetjük, anélkül, hogy szükség lenne egy webkiszolgálóra. A tartalom lekérésének és megjelenítésének folyamata ebben az esetben is ugyanazokból a lépésekből áll, mint amelyeket fentebb ismertettünk – a böngésző megvizsgálja és értelmezi a HTML-fájlban található tartalmat és kódokat – de az út rövidebb, mert a böngészőnek a saját számítógépünk merevlemezén kell kutakodnia, nem pedig egy távoli gépen. Mindazonáltal a fájlokba beágyazott, kiszolgálóoldali programozási nyelven írt utasítások feldolgozásához ekkor is szükséges egy webkiszolgáló, de ennek tárgyalása kívül esik könyvünk keretein. Valójában a könyv összes leckéjén végigrághatjuk magunkat anélkül, hogy webkiszolgálóra lenne szükségünk, de akkor senki sem látná a mesterművünket.

A webgazda kiválasztása

Annak ellenére, hogy éppen az imént mondtuk, hogy a könyv összes leckéjén végigrághatjuk magunkat anélkül, hogy webkiszolgálóra lenne szükségünk, ajánlott egy webkiszolgálót igénybe vennünk. Ne aggódjunk – egy webgazda-szolgáltatáshoz általában gyorsan, fájdalommentesen és viszonylag olcsón hozzájuthatunk. Valójában egy saját tartománynév beszerzése és egy évnyi tárhelybérlet alig kerül többbe, mint ez a könyv.

Ha beírjuk a *web hosting provider* kifejezést a kedvenc keresőnkbe, több millió keresési eredményt és fizetett találatok (értsd: reklámok) végtelen listáját kapjuk. A világon persze nem tevékenykedik ennyi webgazda-szolgáltatás, csak úgy tűnik, mintha ilyen sokan lennének. De ha csak a létező webtárhely-szolgáltatók listáját tekintjük át, már az is hatalmas mennyiség – különösen ha csupán egy olyan helyet keresünk, ahol elhelyezhetjük a saját vagy a cégünk egyszerű kis webhelyét.

A keresést célszerű azokra a szolgáltatókra leszűkítenünk, amelyek a leginkább megfelelnek az igényeinknek. A webgazda kiválasztásának legfontosabb szempontjai a következők:

- **Megbízhatóság/kiszolgálói „ébredés”** – Ha webes jelenlétet szeretnénk, biztosnak kell lennünk benne, hogy a felhasználók állandóan elérhetik a webhelyünket.
- **Ügyfélszolgálat** – Érdemes megvizsgálunk, hogy a szolgáltató többféle csatornán keresztül (telefonon, elektronikus levélben, csevegőszolgáltatással) is rendelkezésre áll-e, valamint hogy van-e online tájékoztató a gyakrabban előforduló problémákról.
- **Kiszolgálói tárterület** – A szolgáltatáscsomag elegendő kiszolgálói területet nyújt ahhoz, hogy a webhelyünkön tervezett összes multimédiás fájl (képek, hang- és videófájlok) elférjen?
- **Sávszélesség** – A szolgáltatáscsomag elegendő sávszélességet biztosít ahhoz, hogy a webhelyünk minden látogatója letölthesse a tartalmakat anélkül, hogy többletdíjat kellene fizetnünk?

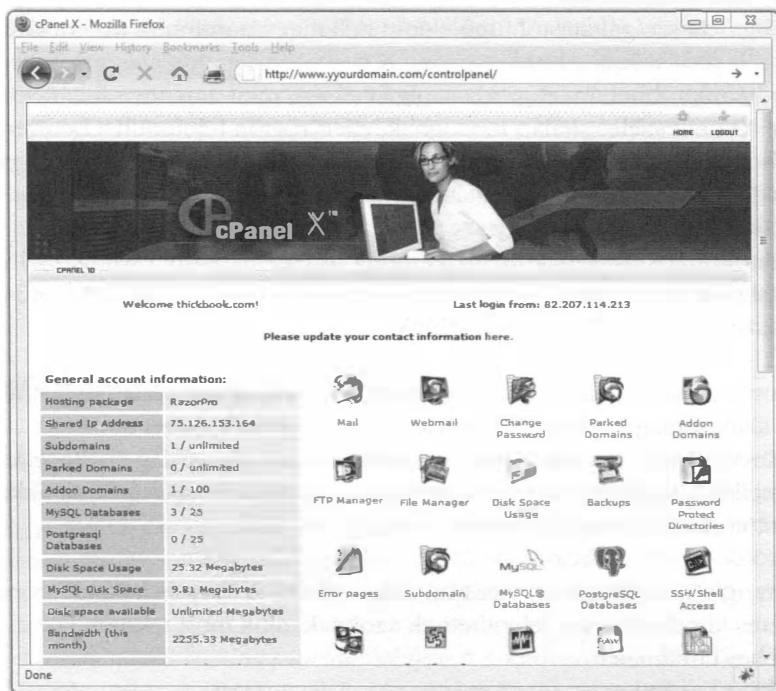
- **Tartománynév-vásárlás és -kezelés** – A csomag egyedi tartománynevet is tartalmaz, vagy a webgazda-szolgáltatástól függetlenül kell megvásárolnunk és fenntartanunk azt?
- **Ár** – Ne fizessünk többet a webgazda-szolgáltatásért, mint amennyit feltétlenül szükséges! Az árak széles skálán mozognak, ezért rögtön felmerülhet bennünk a kérdés, hogy milyen különbség mutatkozik a szolgáltatásban. A különbségnek azonban ritkán van köze a szolgáltatás minőségéhez – csupán a szolgáltató költségeit tükrözi, illetve azt, hogy szerintük mennyi pénzt lehet elkérni az ügyfelektől. Alapszabálynak megfelel, hogy ha egy alapszintű webgazda-szolgáltatási csomagért és egy tartománynévért évi 75 dollárnál többet fizetünk, akkor valószínűleg túl sokat kérnek tőlünk.

Az Amerikai Egyesült Államokban három megbízható webtárhely-szolgáltatót említhetünk, akiknek az alapsomagja bőséges tárhelyterületet és sávszélességet tartalmaz (valamint tartományneveket és egyéb többletszolgáltatásokat), viszonylag alacsony áron. Ha nem is ezek mellett a szolgáltatók mellett döntünk, az alapsomagjuk leírását akkor is hasznos útmutatónak találhatjuk a keresgéléshez.

- **A Small Orange** (<http://www.asmallorange.com>) – A Tiny és a Small csomagok tökéletes kiindulópontot jelenthetnek azoknak, akik most tesznek közzé először webes tartalmakat.
- **DailyRazor** (<http://www.dailyrazor.com>) – A RazorLIMIT és a RazorSTARTER szolgáltatáscsomagok minden szükséges szolgáltatást tartalmaznak, és megbízhatóak.
- **LunarPages** (<http://www.lunarpages.com>) – A Basic szolgáltatáscsomag a legtöbb személyes és kisvállalati webhely számára megfelel.

A jó webtárhely-szolgáltatók egyik ismérve, hogy a rendelkezésünkre bocsátanak egy „vezérlőpultot”, amelyen beállításokat adhatunk meg a fiókunk számára. Az 1.3. ábrán a saját RazorPRO-fiókom vezérlőpultját láthatjuk a Daily Razor szolgáltatónál. Ezt a konkrét vezérlőpult-szoftvert vagy valamelyik ehhez hasonló felépítésű programot sok webgazda alkalmazza, mert az egyértelmű nevű ikonok segítségével mindazokat a feladatokat végrehajthatjuk, amelyek a fiókunk beállításához és kezeléséhez szükségesek.

Lehet, hogy a vezérlőpult használatára soha nem lesz szükségünk, de ha a rendelkezésünkre áll, az (többek között) egyszerűbbé teszi az adatbázisok és más programok telepítését, a webes statisztikák megtekintését, vagy az e-mail címek felvételét. Ha képesek vagyunk utasításokat követni, saját magunk kezelhetjük a webkiszolgálónkat – nincs szükség külön tanfolyam elvégzésére.



1.3. ábra
Szokványos
vezérlőpult

Tesztelés több webböngészőben

Most, hogy megismertük a webes tartalom kézbesítésének folyamatát, illetve a webkiszolgáltató beszerzésének mikéntjét, furcsának tűnhet, hogy visszalépünk, és arról beszélünk, hogy a webhelyünket több webböngészővel is tesztelnünk kell. Mielőtt azonban nekiláthatnánk, hogy mindent megtanuljunk arról, hogyan hozhatunk létre webhelyeket a HTML és a CSS segítségével, az eszünkbe kell vésnünk egy nagyon fontos dolgot: valószínű, hogy a webhelyünk minden látogatója a miénktől eltérő hardver- és szoftverösszeállítást fog használni. Ne feledjük, hogy nincs beleszólásunk abba, hogy a látogatóink a webhely megtekintéséhez milyen eszközöket (asztali vagy hordozható számítógépet, okostelefont, iPhone-t), milyen képernyőfelbontást, milyen böngészőt, a böngészőben milyen ablakméretet, és milyen sebességű kapcsolatot használnak.

Bár minden webböngésző azonos módon dolgozza fel és kezeli az információkat, vannak közöttük bizonyos különbségek, amelyek eredményeképpen a különböző böngészőkben más-más módon jelennek meg az oldalak. Még az ugyanannak a böngészőnek ugyanazt a változatát használó felhasználók is megváltoztathatják az oldalak képét, ha módosítják a megjelenési beállításokat vagy az ablakok méretét. Minden elterjedtebb böngésző lehetővé teszi a felhasználónak, hogy felülbírálja a weboldal

szerzője által meghatározott háttérteret és betűjellemzőket, és a saját beállításait juttassa érvényre. A képernyő felbontása, az ablakméret és a beépülő eszköztárak ugyancsak befolyásolják, hogy a felhasználó mennyit lát egy oldalból a képernyőjén. Mi csak azt biztosíthatjuk, hogy a HTML- és CSS-kódunk szabványkövető legyen.

Semmilyen körülmények között ne töltsünk órákat vég nélkül azzal, hogy valamit „tökéletesre” csiszolunk a saját számítógépünkön, hacsak nem állunk készen a csalódásra, ami akkor ér minket, amikor megtekintjük a művünket egy barátunk gépén, a sarki kávézó terminálján vagy az iPhone-unkon.

A webhelyeinket célszerű mindig ellenőrizni legalább a következő böngészőkkel:

- Apple Safari (<http://www.apple.com/safari/>) Mac-re és Windowsra
- Google Chrome (<http://www.google.com/chrome>) Windowsra
- Mozilla Firefox (<http://www.mozilla.com/firefox/>) Mac-re, Windowsra és Linuxra
- Microsoft Internet Explorer (<http://www.microsoft.com/ie>) Windowsra
- Opera (<http://www.opera.com/>) Mac-re, Windowsra és Linux/UNIX-ra

Összefoglalás

Ezen az órán a webes tartalom előállításának annak az elvével ismerkedtünk meg, amely szövegfájlok HTML-kódokkal történő megjelölésén alapul. Azt is megtanultuk, hogy a webes tartalom nem csupán „oldalakat” jelent, hanem a képek, valamint a hang- és videófájlok is a részét képezik. Mindez a tartalom egy webkiszolgálón kap helyet, vagyis egy olyan távoli számítógépen, amely általában nagyon messze található a saját gépünktől. A számítógépünkön és más eszközökön egy webböngésző segítségével hívjuk le, dolgozzuk fel és jelenítjük meg a webes tartalmat a képernyőnkön.

Láttuk, hogy milyen szempontokat kell figyelembe vennünk, amikor eldöntjük, hogy egy adott webgazda megfelel-e az igényeinknek, és azt is megtanultuk, mennyire fontos, hogy a munkánkat több böngészőben is ellenőrizzük, miután elhelyeztük egy webkiszolgálón. Ha érvényes, szabványkövető HTML- és CSS-kódot írunk, az segít, hogy a webhelyünk többé-kevésbé egyformán jelenjen meg minden látogató számára, de ettől függetlenül célszerű a tervezés során nem csak a fejlesztőcsapatunk, hanem a megcélzott közönség visszajelzéseit is figyelembe venni – a kívülről kapott visszajelzések különös fontosak, ha mi vagyunk a „fejlesztőcsapat” egyetlen tagja.

Kérdezz-felelek

- K: *„Weboldal” helyett „webes tartalmat” mondtunk, de sokan mégis „weboldalakról” beszélnek. Mit értenek alatta? Miben különböznek ezek a kifejezések a „honlap” vagy a „webhely” fogalmától?*
- V: A Világhálót mindig is könnyen el lehetett képzelni úgy, mint egy könyvtárat, amelyben az egyes webhelyek a könyvek, a webhelyeken elhelyezett, tartalmat hordozó fájlok pedig a könyvek „oldalai”. Egy „webhely” egy vagy több, egyidejűleg létrehozott és a tartalmát tekintve egymással kapcsolatban álló oldalból áll. A „honlap” általában az első oldalt jelenti, amelyet a látogatók látnak, amikor megnyitnak egy webhelyet. Probléma akkor adódhat, ha valaki azt mondja, hogy „látogasd meg a weboldalamat”, amikor valójában azt érti alatta, hogy „látogass el a webhelyemre” – egy webhely ugyanis számos oldalból áll. Ha webes tartalmak egy gyűjteményére nem webhelyként, hanem oldalként hivatkozunk, akkor azt hihetjük, hogy nem igazán értjük, hogyan működik a Web – vagy azért, mert nem tudjuk, hogyan alkotnak a webes tartalmak együttesen egy webhelyet, vagy azért, mert eddig csak olyan webhelyeket terveztünk és valósítottunk meg, amelyek egyetlen oldalból állnak.
- K: *Megnéztem néhány weboldal „HTML-forrását” az Interneten, és nagyon bonyolultnak tűnik. Úgy kell gondolkodnom, mint egy számítógép-programozónak, ha meg szeretném tanulni, hogyan működik a kód?*
- V: Bár az összetett HTML-oldalak kódja valóban ijesztő lehet, a HTML használatát sokkal könnyebb megtanulni, mint a szoftverprogramozási nyelvekét (amilyen például a C++ vagy a Java). A HTML jelölő- vagy leírónyelv, nem pedig programozási nyelv: szöveget címkézünk fel vele, hogy a böngésző azt adott módon jeleníthesse meg. Ez teljesen más megközelítést igényel, mint egy számítógépes program fejlesztése. Nincs szükség semmilyen programozói tudásra vagy tapasztalatra ahhoz, hogy sikeresen állíthassunk elő webes tartalmat. Annak, hogy sok kereskedelmi webhelyen bonyolultnak tűnik a HTML-kód, az az egyik oka, hogy valószínűleg valamilyen grafikus (vizuális) webtervező eszközzel – úgynevezett WYSIWYG („what you see is what you get” – „azt kapod, amit látsz”) szerkesztővel, amely behelyettesíti azt a leírókódot, amelyet az adott körülmények között a szoftver fejlesztője által beleprogramozott utasítás szerint be kell írnia – hozták létre, nem pedig kézi módszerrel, amikor is mi tartjuk kézben teljesen az eredményként kapott leírókódot. Ebben a könyvben a kódolást az alapoktól kezdve tanuljuk meg, ami általában tiszta, könnyen olvasható forráskódot eredményez. A grafikus webtervező eszközök hajlamosak nehezen olvashatóvá tenni a kódot, illetve olyan kódot előállítani, amely túlbojlyított, és nem követi a szabványokat.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Határozzuk meg a *webes tartalom* kifejezés jelentését!
2. Hány fájlt kell tárolnunk a webkiszolgálón egyetlen, szöveget és két képet tartalmazó weboldal előállításához?
3. Milyen szolgáltatásokat kell górcső alá vennünk egy webgazda esetében?

Válaszok

1. A *webes tartalom* kifejezés azoknak a szövegeknek, képeknek, hang- és videófájloknak és egyéb multimédiás állományoknak a teljes körét lefedi, amelyeket a webkiszolgálóról kézbesítenek a webböngészőknek.
2. Hármat: egyet magához a weboldalhoz, amelyben a szöveg és a HTML-leírókód kap helyet, és egyet-egyet a két kép számára.
3. A megbízhatóságot, az ügyfélszolgálatot, a webes tárterületet és a sávszélességet, a tartománynév-szolgáltatásokat, a webhely-felügyeleti kiegészítő szolgáltatásokat, valamint az árat.

Gyakorlatok

- Tisztázzuk a webgazda kérdését: úgy szeretnénk elvégezni a kötet leckéit, hogy a fájlokat helyben, a saját számítógépünkön tekintjük meg, vagy egy webtárhelyszolgáltatót szeretnénk igénybe venni? A döntésben segíthet, ha tudjuk, hogy a legtöbb webgazda még azon a napon elindítja a szolgáltatást, amikor bérbe vesszük a tárhelyet.



2. ÓRA

A webes tartalom közzététele

A lecke tartalma:

- Alapszintű HTML-fájlok létrehozása szövegszerkesztővel
- A fájlok átvitele a webkiszolgálóra az FTP segítségével
- Hol helyezzük el a fájlokat a webkiszolgálón?
- A webes tartalom közzététele webkiszolgáló nélkül
- Hogyan alkalmazzuk az egyéb közzétételi módszereket, például a webnaplókat?

Az előző órán megtudtuk, hogy kérhetünk webes tartalmat egy webböngészőn keresztül, és hogyan válaszol a webkiszolgáló a kérelmekre. Ezen az órán azt tanuljuk meg, hogy tartalomkészítőként mi a szerepünk az Interneten elérhető webes tartalmak közzétételében – a tartalmat ugyanis közzé kell tenni egy webkiszolgálón, hogy mások hozzáférhessenek.

Az óra példafájljának elkészítése

Mielőtt elkezdenénk az órát, vessünk egy pillantást a 2.1. példára. A példában látható kód egyszerű webes tartalmat takar: néhány sornyi HTML-kód, amely a „Hello World! Welcome to My Web Server.” szöveget írja ki, nagy, félkövér betűkkel, két sorban, a böngésző ablakában középre igazítva.

2.1. példa A HTML-példafájlunk

```
<html>
<head>
<title>Hello World!</title>
</head>
<body>
<h1 align="center">Hello World!<br/>Welcome to My Web Server.</h1>
</body>
</html>
```

Ennek a tartalomnak a hasznosításához nyissuk meg a kedvenc sima szöveges szerkesztőnket, például a Jegyzettömböt (Notepad, Windowson) vagy a TextEditet (Macen). WordPadet, Microsoft Wordöt vagy más teljes körű szolgáltatást nyújtó szövegszerkesztő programot ne használjunk, mert ezek másfajta fájlokat hoznak létre, nem pedig olyan sima szövegfájlokat, amelyekre a webes tartalomhoz szükségünk van.



A szövegszerkesztőkről bővebben is tanulunk a 3. órán; most csak annyit szeretnénk volna, hogy legyen egy példafájlunk, amelyet elhelyezhetünk egy webkiszolgálón.

Írjuk be a 2.1. példában látható tartalmat, és mentjük a fájlt `sample.html` néven.

A `.html` kiterjesztés árulja el a webkiszolgálónak, hogy a fájl HTML-kódot tartalmaz, és amikor a kiszolgáló elküldi a fájl tartalmát a böngészőnek, a böngésző is innen tudja, hogy HTML-ről van szó, és ennek megfelelően képezi le a fájlt. Most, hogy van egy HTML-példafájlunk, amelyet használhatunk – és remélhetőleg elhelyezhetünk valahol, például egy webtárhely-szolgáltatói fiókban –, rátérhetünk a webes tartalom közzétételére.

Fájlok átvitele az FTP segítségével

Ahogy már megtanultuk, a webes tartalmat egy webkiszolgálón kell elhelyeznünk, hogy mások számára hozzáférhetővé tegyük. Ezt a műveletet rendszerint a *fájlátviteli protokoll* (File Transfer Protocol, FTP) segítségével hajtjuk végre. Az FTP használatához egy FTP-ügyfélre van szükségünk, vagyis egy olyan programra, amellyel átvihetjük a fájlokat a számítógépünkől a webkiszolgálóra.

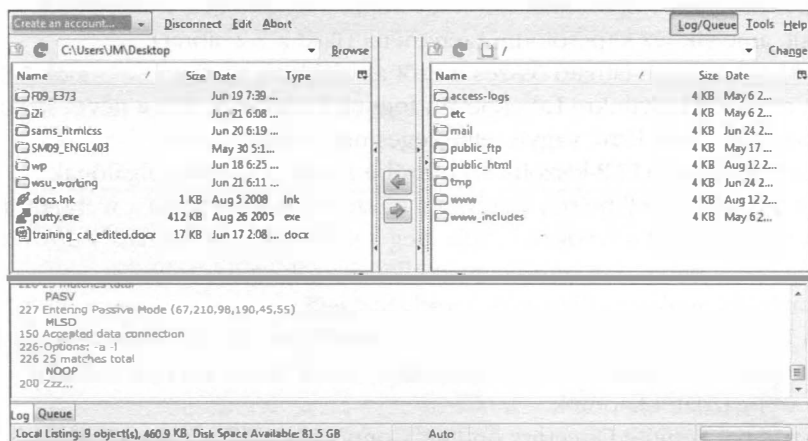
Az FTP-ügyfelek háromféle információt igényelnek ahhoz, hogy kapcsolatot tudjanak teremteni a webkiszolgálóval. Ezeket az információkat a webgazda adja meg nekünk, amikor felállítjuk a fiókunkat:

- Az állomásnév (hostname) vagy cím, amelyhez majd kapcsolódunk
- A fiókunk felhasználóneve (user name)
- A fiókunk jelszava (password)

Ha rendelkezünk ezekkel az adatokkal, készen állunk arra, hogy egy FTP-ügyfél segítségével tartalmat vigyünk át a webkiszolgálónkra.

Az FTP-ügyfél kiválasztása

Függetlenül attól, hogy milyen FTP-ügyfelet használunk, az FTP-ügyfél általában ugyanolyan felületet nyújt a számunkra. A 2.1. ábrán a FireFTP felületét láthatjuk; ezt az FTP-ügyfelet használja a Firefox webböngésző. A helyi számítógép (a mi számítógépünk) könyvtárlistája a képernyő bal oldalán jelenik meg, a távoli gép (a webkiszolgáló) tartalma pedig jobboldalt. Ahogy a 2.1. ábrán is látható, jellemzően kapunk egy jobbra, valamint egy balra mutató nyilat ábrázoló gombot. A jobbra mutató nyíllal a kiválasztott fájlokat a számítógépünkről a webkiszolgálóra küldhetjük, míg a balra mutató nyíllal a webkiszolgálóról vihetünk át fájlokat a saját gépünkre. Sok FTP-ügyfél azt is lehetővé teszi, hogy egyszerűen kijelöljük a kívánt fájlokat, és az egérrel áthúzzuk azokat a célgép könyvtárlistájába.



2.1. ábra
A FireFTP felülete

Számos szabadon hozzáférhető FTP-ügyfél áll a rendelkezésünkre, de fájlokat átvihetünk a webes alapú File Manager (Fájlkezelő) eszköz segítségével is, amelyet valószínűleg megtalálunk a webkiszolgálónk vezérlőpultján. A fájlátvitelnek ez a módszere azonban általában további lépésekkel bővíti a folyamatot, tehát nem olyan egyszerű, mint egy FTP-ügyfél telepítése a számítógépünkre. Íme néhány népszerű ingyenes FTP-ügyfélprogram:

- Classic FTP (<http://www.nchsoftware.com/classic/>) Mac-re és Windowsra
- Cyberduck (<http://cyberduck.ch/>) Mac-re
- Fetch (<http://fetchsoftworks.com/>) Mac-re
- FileZilla (<http://filezilla-project.org/>) minden rendszerre
- FireFTP (<http://fireftp.mozdev.org/>) Firefox-bővítmény minden rendszerre

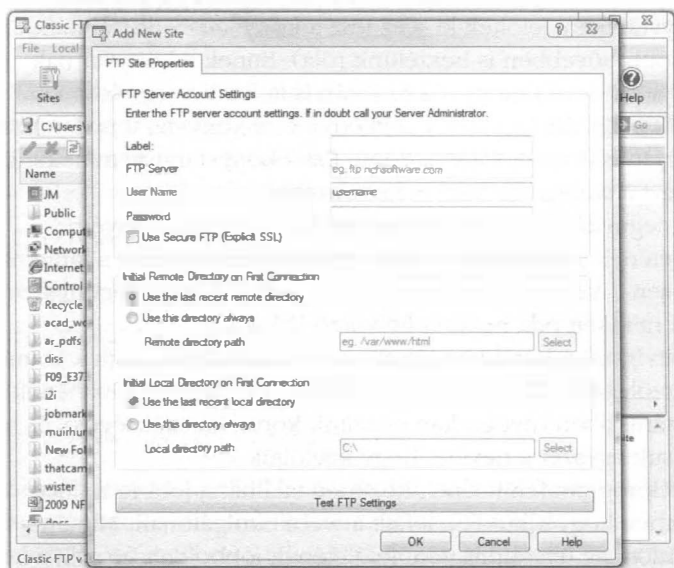
Miután kiválasztottunk egy FTP-ügyfelet, és telepítettük azt a számítógépünkre, készen állunk rá, hogy fájlokat töltsünk fel a webkiszolgálóra, illetve fájlokat töltsünk le onnan. A következő részben azt mutatjuk be, hogyan működik mindez az óra elején elkészített példafájl esetében.

Az FTP-ügyfél használata

Az alábbiakban bemutatott lépések azt írják le, hogy miként kapcsolódhatunk a Classic FTP segítségével a webkiszolgálóhoz, és vihetünk át egy fájlt. Mivel azonban minden FTP-ügyfél hasonló – ha nem pontosan ugyanilyen – felületet használ, ha megértjük a lépéseket, bármelyik FTP-ügyfél használatára képessé válunk.

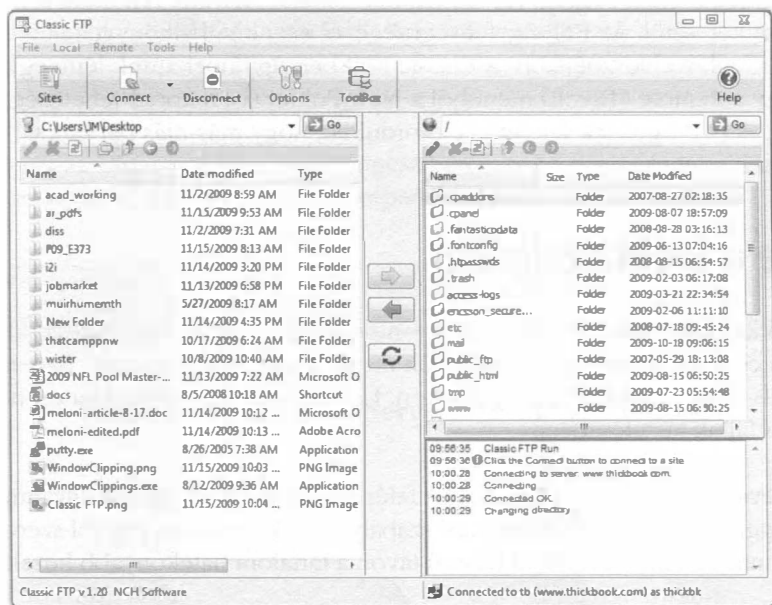
Ne feledjük, hogy először is szükségünk lesz az állomásnévre, a fiókhoz kapcsolódó felhasználónévre, valamint a fiók jelszavára.

1. Indítsuk el a Classic FTP programot, és kattintsunk a Connect (Kapcsolódás) gombra. A program először arra kér, hogy adjuk meg annak a webhelynek az adatait, amelyikhez kapcsolódni szeretnénk (lásd a 2.2. ábrát).
2. Töltsük ki a 2.2. ábrán látható összes mezőt az alábbiak szerint:
 - A saját webhelyünkre Label néven fogunk hivatkozni. Ezt a nevet senki más nem fogja látni, vagyis tetszőleges nevet beírhatnánk.
 - Az FTP Server (FTP-kiszolgáló) mezőbe annak a webkiszolgálónak az FTP-címét kell beírni, amelyiken el szeretnénk helyezni a weboldalainkat. Ezt a címet a webgazda adja meg a számunkra. A formája valószínűleg *tartomány.com* lesz, de azért ellenőrizzük az információt, amit akkor kaptunk, amikor előfizettünk a szolgáltatásra.
 - A User Name (Felhasználónév) és Password (Jelszó) mezőket ugyancsak azoknak az adatoknak a felhasználásával kell kitöltenünk, amelyeket a webgazdától kaptunk.
 - Az Initial Remote Directory on First Connection (Távoli kezdőkönyvtár az első kapcsolódásnál) és az Initial Local Directory on First Connection (Helyi kezdőkönyvtár az első kapcsolódásnál) beállításokon ne változtassunk, amíg hozzá nem szoktunk az ügyfélprogram használatához, és ki nem alakítottuk a számunkra megfelelő munkafolyamatot.
3. Ha végeztünk a beállításokkal, az OK gombra kattintva mentjük őket, és hozzuk létre a kapcsolatot a webkiszolgálóval.
Ekkor egy párbeszédablakot fogunk látni, amely arról tájékoztat, hogy a Classic FTP megkísérel kapcsolódni a webkiszolgálóhoz. Sikeres kapcsolatfelvétel esetén a 2.3. ábrán láthatóhoz hasonló felület jelenik meg, a helyi könyvtár tartalmával a bal, és a webkiszolgáló tartalmával a jobb oldalon.
4. Most már *majdnem* készen állunk rá, hogy fájlokat vigyünk át a webkiszolgálónkra. Csak annyi van hátra, hogy átváltunk a webkiszolgálón az úgynevezett *dokumentumgyökérre* (document root) vagy *gyökérkönyvtárra*.



2.2. ábra

Kapcsolódás egy új webhelyhez a Classic FTP-ben



2.3. ábra

Sikeres kapcsolódás egy távoli webkiszolgálóhoz a Classic FTP segítségével

A webkiszolgáló dokumentumgyökere az a könyvtár, amelyet a webes tartalom legfelső szintű könyvtáraként jelöltek ki – ez lesz a könyvtárszerkezet kiindulópontja (az órán később bővebben is beszélünk róla). Ennek a könyvtárnak gyakran `public_html` a neve (ahogy a 2.3. ábrán is látható), vagy `www` (ezt szintén láthatjuk a 2.3. ábrán, mert álnévként a `www` könyvtárat is létrehozták a `public_html` helyettesítésére), vagy `htdocs`. Ezt a könyvtárat nem nekünk kell létrehoznunk; ezt a webgazda teszi meg helyettünk.

A gyökérkönyvtár megnyitásához kattintsunk duplán a könyvtár nevére.

Az FTP-ügyfél felületének jobb oldalán ekkor ennek a könyvtárnak a tartalma kell, hogy megjelenjen. (A könyvtár ezen a ponton még valószínűleg üres, hacsak a webgazda nem tett oda nekünk helyőrző fájlokat.)

5. A célunk az, hogy átvigyük a korábban létrehozott `sample.html` fájlt a számítógépünkről a webkiszolgálóra. Keressük meg a fájlt az FTP-ügyfél felületének bal oldalán levő könyvtárlistában (nyugodtan nézzünk körül, ha szükséges), és ha megtaláltuk, kattintsuk egyszer a nevére, hogy kijelöljük.
6. Kattintsunk az ügyfélprogram felületének közepén található, jobbra mutató nyilat ábrázoló gombra, hogy elküldjük a fájlt a webkiszolgálónak. Miután a fájltvitel befejeződött, az ügyfélprogram felületének jobboldalt frissülnie kell, hogy jelezze, hogy a fájl célba ért.
7. Kattintsunk a Disconnect (Kapcsolat bontása) gombra a kapcsolat bezárásához, és lépünk ki a Classic FTP programból.

Ezek a lépések elvileg mindig megegyeznek, amikor fájlokat akarunk feltölteni a webkiszolgálóra FTP-n keresztül. Az FTP-ügyfél segítségével ezenkívül alkönyvtárakat is létrehozhatunk a távoli webkiszolgálón. A Classic FTP-ben úgy hozhatunk létre egy alkönyvtárat, hogy a Remote (Távoli) menüből a New Folder (Új mappa) lehetőséget választjuk. Az egyes FTP-ügyfelek felületén előfordulhat, hogy más-más paranccsal érhetjük el ugyanezt a lehetőséget.

A fájlok helye a webkiszolgálón

A webes tartalom karbantartása szempontjából fontos, hogy hogyan rendezzük a tartalmat – nem csak azért, hogy a felhasználók megtalálják, amit keresnek, hanem azért is, hogy mi is eligazodjunk a kiszolgálónkon. Ha a fájlokat könyvtárakba csoportosítjuk, az segít a fájlok kezelésében.

A könyvtárak elnevezése és a könyvtárszerkezet felépítése a webkiszolgálón ugyanúgy teljesen tőlünk függ, mint a fájlok karbantartási szabályainak kialakítása. Egy jól szervezett kiszolgáló fenntartása mindazonáltal hosszú távon a tartalom hatékonyabb kezelését teszi lehetővé.

Alapszintű fájlkezelés

A Weben böngészve talán észrevettük, hogy az URL-ek megváltoznak, miközben mozgunk egy webhelyen belül. Például ha egy vállalat webhelyét nézegetjük, és egy grafikus navigációs elemre kattintunk, hogy eljussunk a cég termékeihez vagy szolgáltatásaihoz, az URL valószínűleg erről

`http://www.cegnev.com`

erre változik:

`http://www.cegnev.com/products/`

Vagy erre:

`http://www.cegnev.com/services/`

Az előző részben anélkül használtuk a *dokumentumgyökér* kifejezést, hogy ténylegesen elmagyaráztuk volna, mit is jelent. A webkiszolgáló dokumentumgyökere vagy gyökérkönyvtára lényegében a teljes URL végén álló perjel. Ha a tartományunk például a `tartomany.com`, és az URL-ünk a `http://www.tartomany.com/`, akkor a dokumentumgyökér a záró perjel (/) által jelölt könyvtár. A dokumentumgyökér a webkiszolgálón kialakított könyvtárszerkezet kiindulópontja – az a hely, ahol a webkiszolgáló először keresi a böngészők által kért fájlokat.

Ha a `sample.html` fájlt a korábbi utasításnak megfelelően a dokumentumgyökérben helyezük el, akkor a böngészőből a következő URL-en érhetjük el:

`http://www.tartomany.com/sample.html`

Ha ezt az URL-t beírjuk a webböngészőnkbe, a 2.4. ábrán látható leképezett `sample.html` fájl jelenik meg a képernyőnkön.



2.4. ábra

A sample.html fájl elérése egy webböngészőn keresztül

Ha azonban egy új könyvtárat hoztunk létre a dokumentumgyökéren belül, és a `sample.html` fájlt ebbe a könyvtárba tettük, akkor a fájl az alábbi URL-lel érhető el:

`http://www.tartomany.com/ujkonyvtar/sample.html`

Ha a `sample.html` fájlt abba a könyvtárba tettük, amelyet először láttunk, amikor kapcsolódtunk a kiszolgálóhoz – vagyis *nem* váltottunk könyvtárat, és a fájlt *nem* a dokumentumgyökérben helyeztük el –, akkor a `sample.html` fájl egyetlen URL-en sem lesz elérhető a webkiszolgálón. A fájl természetesen ott lesz azon a számítógépen, amelyet webkiszolgálóként ismerünk, de mivel nem a dokumentumgyökérben található – amelyről a kiszolgálószoftver tudja, hogy ott kell először keresnie a fájlokat –, senki nem tudja majd elérni egy webböngészőn keresztül.

Mi a tanulság? Nos, az, hogy mindig váltsunk a webkiszolgáló gyökérkönyvtárára, mielőtt hozzálátnánk a fájlok átviteléhez.

Ez különösen igaz a grafikákra és más multimédia-fájlokra. A webkiszolgálókon általában található egy `images` (képek) nevű könyvtár, ahol – amint kitalálhattuk – az összes képfájlnkat tárolhatjuk. Ehhez hasonló népszerű könyvtár például a `css`, amely a stíluslapfájlok tárolójaként szolgál (ha egynél több stíluslapfájlt használunk), vagy a `js`, ahol a külső JavaScript-állományokat helyezhetjük el. Vagy, ha tudjuk, hogy a webhelyen lesz majd egy terület, ahonnan a látogatók különféle típusú fájlokat tölthetnek le, létrehozhatunk a számukra egy `downloads` (letöltések) nevű könyvtárat.

Akár egy tömörített ZIP fájlról van szó, amelyben a művészi portfóliónkat tároljuk, vagy egy Excel-táblázatról, amelyben eladási mutatók találhatók, gyakran hasznos olyan fájlokat is közzétenni az Interneten, amelyek nem egyszerűen weboldalak. Ha egy olyan állományt szeretnénk hozzáférhetővé tenni a Weben, amely nem HTML-fájl, csak annyit kell tennünk, hogy feltöltjük a fájlt a webhelyünkre, *mintha* HTML-fájl lenne, az óra korábbi részében a feltöltésre vonatkozóan ismertetett utasításokat követve. Miután a fájlt feltöltöttük a webkiszolgálóra, létrehozhatunk egy rá mutató hivatkozást (ennek módját a későbbi órákon tanuljuk meg). Más szavakkal, a webkiszolgálónk sokkal többre képes egyszerű HTML-fájlok szolgáltatásánál.

Lássunk egy példát a HTML-kódra, amelyről majd később tanulunk! Az alábbi kóddal az `artfolio.zip` nevű állományra hivatkozhatunk, amelyet a webhelyünk letöltési könyvtárában helyeztünk el, a hivatkozás szövege pedig a „Download my art portfolio!” (Tölts le a művészi portfóliómat!) lesz:

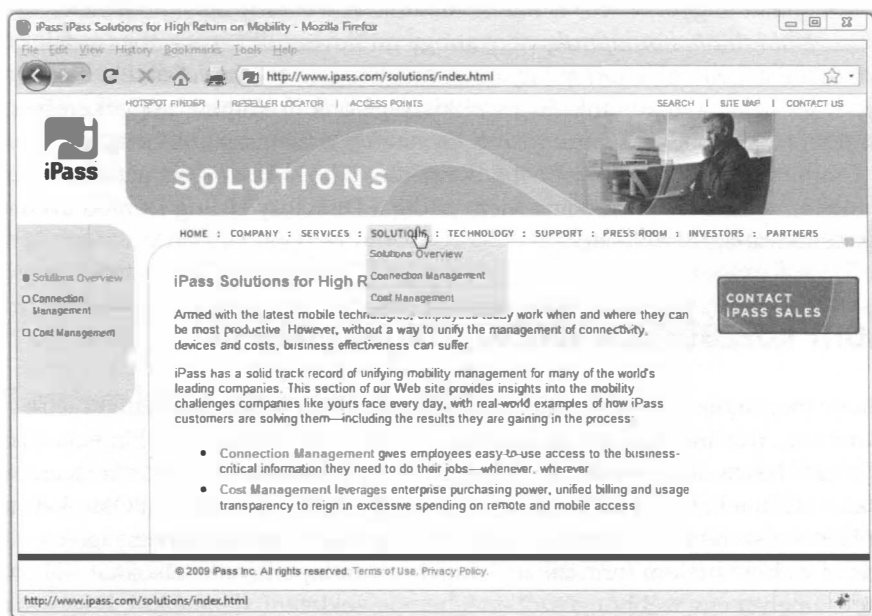
```
<a href="/downloads/artfolio.zip ">Download my art portfolio!</a>
```

Indexoldalak használata

Amikor „indexről” hallunk, valószínűleg a könyvek végén található tárgymutató jut az eszünkbe, amelyben különféle kulcsszavak és témakörök szerint kereshetünk oldalakat. A webkiszolgáltatók könyvtáraiban elhelyezett indexfájlok is betölthetik ezt a szerepet, ha úgy tervezzük meg őket – valójában innen kapták a nevüket.

Az `index.html` nevet annak a fájlnek (röviden *indexfájl*nak, ahogy általában hivatkoznak rá) adjuk, amelyet alapértelmezett oldalként szeretnénk megjeleníteni a felhasználók számára, amikor felkeresik a webhelyünk egy adott könyvtárát. Ha az indexoldalt a használhatóságot szem előtt tartva készítettük el, a felhasználók erről az oldalról minden tartalmat elérhetnek, ami a webhely adott részén található.

A 2.5. ábrán látható lenyíló menü és bal oldali navigációs sáv hivatkozásai például egyaránt három oldalra mutatnak: a Solutions Overview (ez maga a szakasz indexoldala), a Connection Management és a Cost Management oldalakra. Az – `index.html` nevű, a solutions könyvtárban elhelyezett – oldal maga is tartalmaz hivatkozásokat a másik két oldalra, amely a solutions szakaszban található. Amikor a felhasználók ennek a szakasznak az indexoldalára érkeznek az adott webhelyen, a szakasz bármelyik oldalát elérhetik onnan (még hozzá háromféleképpen).



2.5. ábra

Egy jó szakasz-indexoldal

Az indexoldal másik célja az, hogy ha a felhasználók felkeresik a webhelyünk egyik könyvtárát, amely rendelkezik indexoddallal, de nem adják meg az oldal nevét, akkor is a szakasz – vagy maga a webhely – kezdőoldalára kerüljenek.

A fenti esetben például a felhasználó bármelyiket beírhatja az alábbi két URL közül, akkor is a webhely solutions szakaszának kezdőoldalára kerül:

```
http://www.ipass.com/solutions/
http://www.ipass.com/solutions/index.html
```

Ha a solutions könyvtárban nincs index.html oldal, a művelet eredménye a webkiszolgáló beállításaitól függ. Ha a kiszolgálót úgy állították be, hogy ne engedje meg a könyvtárakban való tallózást, akkor a felhasználó a „Directory Listing Denied” (A könyvtártartalom kiíratása megtagadva) üzenetet kapná, ha oldalnév megadása nélkül kísérli meg elérni az URL-t. Ha azonban a kiszolgálón engedélyezték a könyvtárakban való tallózást, a könyvtárban található fájlok listája jelenik meg.

A kiszolgáló beállításait a webgazda már valószínűleg megadta helyettünk. Amennyiben a szolgáltató megengedi, hogy a vezérlőpulton keresztül módosítsuk a kiszolgálói beállításokat, módosíthatjuk azokat, hogy a kiszolgálónk a kérelmekre a mi igényeink szerint válaszoljon.

Indexfájlt nem csak az alkönyvtárakban, hanem a webhelyünk legfelső szintű könyvtárban (a dokumentumgyökérben) is használhatunk. A webhely első oldalának – a *kezdőoldalnak*, *főoldalnak*, *honlapnak*, vagy ahogy mi nevezzük azt a webes tartalmat, amelyet először szeretnénk a tartományunkat meglátogató felhasználók elé tárni – az index.html nevet kell adnunk, és a webkiszolgálónk dokumentumgyökerében kell elhelyeznünk. Ezzel gondoskodunk róla, hogy amikor a felhasználók beírják a `http://www.tartomany.com/` címet a böngészőjükbe, a kiszolgáló azt a tartalmat küldje el nekik, amit szeretnénk volna (nem pedig a Directory Listing Denied üzenetet vagy más nemkívánatos tartalmat).

A tartalom közzététele webkiszolgáló nélkül

Annak, hogy megtanuljuk a HTML használatát, és azt, hogy miként hozhatunk létre webes tartalmat, nyilvánvalóan az az elsődleges célja, hogy HTML és multimédia-fájlokat tegyünk közzé. Lehetnek azonban olyan helyzetek is, amikor nincs is más kivitelezhető módja a közzétételnek. Előfordulhat például, hogy CD-ROM-okat, DVD-ROM-okat vagy USB-meghajtókat szeretnénk terjeszteni egy vásáron, amelyeken reklámanyagokat helyezünk el webes tartalom formájában – vagyis olyan hiperhivatkozásokkal ellátott szöveggént, amelyet egy webböngészőben lehet megtekinteni, de anélkül, hogy szükség lenne egy webkiszolgálóra. Az is lehet, hogy egy tanfolyamon szeretnénk a hallgatóknak HTML alapú oktatóanyagokat kiosztani hordozható meghajtókon. Ez csak két példa arra, hogy miként használhatók a HTML-oldalak az Internettől független közzétételre.

Az említett megoldásokat úgy is szokták nevezni, hogy *helyi webhelyek* (local site) létrehozása. Bár webkiszolgálóra nincs szükség hozzájuk, az ilyen hiperszöveges tartalmakat is *webhelyeknek* hívják. A „helyi” kifejezés arra utal, hogy a fájlokat helyben érjük el, nem pedig egy távoli kapcsolaton (egy webkiszolgálón) keresztül.

Tartalom közzététele helyben

Tegyük fel, hogy egy olyan helyi webhelyet kell készítenünk, amelyet aztán USB-meghajtókon terjeszthetünk. A mai USB-kulcsok közül még a legolcsóbbak is olyan sok adatot képesek tárolni – és az alapszintű hiperszöveges fájlok kimondottan kis méretűek –, hogy egy teljes webhelyet elhelyezhessünk rajtuk, egy *teljes értékű webböngészővel* együtt.

Amikor helyi webhelyet hozunk létre és terjesztünk, nem kell feltétlenül mellékelnünk egy webböngészőt is, bár kedves gesztus. Joggal feltételezhetjük, hogy a felhasználók rendelkeznek saját webböngészővel, és valamelyik könyvtár `index.html` fájljának megnyitásával fogják kezdeni a hiperhivatkozásokkal összekapcsolt tartalom böngészését. Ha azonban egy webböngészőt is fel szeretnénk tenni az USB-meghajtóra, keressük fel a <http://www.portableapps.com/> címet, és ott keressük meg a Portable Firefox (hordozható Firefox) böngészőt.

Egyszerűen gondoljunk úgy az USB-meghajtó könyvtárszerkezetére, mintha az a webkiszolgálónké lenne. Az USB-meghajtó könyvtárszerkezetének legfelső szintje tekinthető a dokumentumgyökérnek. Vagy, ha a tartalomhoz egy webböngészőt is mellékelünk, két könyvtárunk is lehet – az egyik például browser (böngésző), a másik pedig content (tartalom) néven. Ebben az esetben a content könyvtár lesz a gyökérkönyvtárunk. A dokumentumgyökéren belül további alkönyvtáraink is lehetnek, amelyekben tartalmat és kiegészítő multimédia-fájlokat helyezhetünk el.

A jó rendszerezés egy helyi webhely esetében is éppen olyan fontos, mint egy távoli webhelynél, ezért ügyeljünk rá, hogy ne legyenek hibás hivatkozások a HTML-fájljainkban. A fájlok összekapcsolásáról egy későbbi leckében részletesebben is tanulunk majd.

Tartalom közzététele webnaplóban

Lehet, hogy már van webnaplónk („bloggal”) egy olyan külső szolgáltatónál, mint a Blogger vagy a WordPress, és így már tettünk közzé tartalmat anélkül, hogy rendelkezünk volna kijelölt webkiszolgálóval, vagy bármit is tudtunk volna a HTML-ről. Ezek a szolgáltatók a *forrásszerkesztők* mellett *grafikus szerkesztőket* is kínálnak, ami azt jelenti, hogy anélkül írhatunk be szöveget, és formázhatjuk azt félkövér, dőlt vagy különféle színű betűkkel, hogy tudnánk, milyen HTML-kód szükséges ehhez. Mindazonáltal, amikor a szerkesztő Publish (Közzététel) gombjára kattintunk, a tartalomból HTML-kód jön létre.

Azzal a tudással felvértezve azonban, amelyet ebből a könyvből szerezhetünk, a webnaplónkat magasabb szintre emelhetjük, mivel a webnapló tartalmát és sablonjait a forrásszerkesztő segítségével alakíthatjuk, így jobban kézben tarthatjuk a webnapló kinézetét. Ennek módja különbözik attól, ahogy a HTML-fájlokat létrehozzuk, és FTP-n keresztül feltöltjük a kijelölt webkiszolgálónkra, de nem szabad elfelejtenünk, hogy a webnaplóírás is egyfajta webes közzététel.

A webes tartalom ellenőrzése

Amikor fájlokat viszünk át a webkiszolgálóra, vagy hordozható lemezen helyezük el őket helyi böngészés céljára, azon nyomban célszerű alaposan ellenőriznünk minden oldalt. Az alábbi ellenőrzőlista segít majd, hogy biztosítsuk, hogy a webes tartalom úgy viselkedjen, ahogy elvárjuk tőle. Megjegyzendő, hogy a használt kifejezések némelyike most még ismeretlen lesz a számunkra, de ahogy haladunk a könyvben, visszatérhetünk ide, és akár nagyobb webhelyeket is építhetünk:

- Mielőtt átvinnénk a fájlokat, teszteljük őket helyben, a számítógépünkön, hogy meggyőződjünk a hivatkozások működéséről, illetve arról, hogy a tartalom a kívánt képi elrendezést tükrözi-e. Miután átvittük a fájlokat a webkiszolgálóra vagy hordozható eszközre, próbáljuk ki őket ismét.
- A teszteket annyi böngészővel hajtsuk végre, amennyivel csak tudjuk – a Chrome, a Firefox, az Internet Explorer, az Opera és a Safari mindenképpen legyen rajta a listán –, és az ellenőrzést Mac és Windows rendszeren egyaránt végezzük el. Ha lehetséges, vizsgáljuk meg a tartalmat alacsony (800 x 600-as) és magas (1600 x 1200-as) felbontásnál is.
- Kapcsoljuk ki az automatikus képbetöltést a böngészőnkben, mielőtt nekilátnánk a tesztelésnek, hogy lássuk, hogyan festenek az oldalak a képek nélkül. Ellenőrizzük a képek helyettesítő szövegét, majd kapcsoljuk vissza a képbetöltést, hogy a képek is megjelenjenek, és vizsgáljuk át ismét gondosan az oldalakat.
- A böngésző szövegbeállításainak segítségével nézzük meg az oldalakat különböző betűméretekkel, hogy biztosak legyünk benne, hogy az elrendezés nem esik szét akkor sem, ha a felhasználók felülbírálják a szövegre vonatkozó beállításainkat.
- Várjuk meg minden oldal esetében, hogy teljesen betöltődjön, majd görgessünk le teljesen a lap aljáig, hogy meggyőződjünk róla, hogy minden kép ott jelenik meg, ahol kell.
- Mérjük le, hogy mennyi ideig tart az egyes oldalak betöltése. Ha a betöltés néhány másodpercnél többet vesz igénybe, az adott oldalon található információk elég értékesek ahhoz, hogy a felhasználók ne távozzanak az oldalról a betöltés befejeződése előtt? A szélessávú elérés ma már elterjedt, de ez nem jelenti azt, hogy az oldalakat 1 megabájtos képekkel kell telezsúfolnunk.

Ha a fenti teszteken minden oldalunk átmegy, nyugodtan hátradőlhetünk – a webhelyünk készen áll arra, hogy a nyilvánosság elé tárjuk.

Összefoglalás

Ezt az órát azzal kezdtük, hogy elkészítettünk egy nagyon egyszerű HTML-fájlt, amelyet tesztfájlként használtunk a webkiszolgálóra történő fájlátvitelhez. Megtanultuk, hogyan működik a fájlátvitel, és hogy milyen szoftverre van szükségünk az ávitel végrehajtásához (egy FTP-ügyfélre). A webkiszolgáló könyvtárszerkezetéről és a fájlkezelésről is megtudtunk ezt-azt, valamint tisztáztuk, hogy milyen fontos szerepe van az `index.html` fájlnek a webkiszolgáló egy adott könyvtárában. Megtanultuk emellett, hogy webes tartalmat hordozható eszközön is terjeszthetünk, és szót ejtettünk arról is, hogy miként célszerű elrendezni a fájlokat és könyvtárakat, ha a tartalmat egy távoli webkiszolgáló használata nélkül szeretnénk megtekinteni. Végül megtanultuk azt is, hogyan ellenőrizzük a fájljainkat, mielőtt a webhelyünket nyilvános fogyasztásra alkalmasnak nyilváníthatnánk.

Kérdezz-felelek

- K: *A javasolt tesztek mindegyike tovább tart, mint az oldalak elkészítése! Nem lehetne kevesebb ellenőrzéssel megúszni?*
- V: Ha az oldalainkkal nem pénzt szeretnénk keresni, vagy valamilyen fontos szolgáltatást nyújtani, akkor valószínűleg nem számít annyira, ha egyes felhasználóknál furcsán jelennek meg, vagy egyszer-egyszer hibát okoznak. Ilyen esetben elég, ha megnézzük az oldalakat néhány különböző böngészőben. Ha azonban profizmust szeretnénk sugározni, semmi nem helyettesítheti a szigorú tesztelést.
- K: *Most komolyan, ki törődik azzal, hogy miként rendezem el a webes tartalmat?*
- V: Hisszük vagy sem, az általunk kínált webes tartalom elrendezése a keresőprogramoknak és a webhelyünk leendő látogatóinak is számít (erről bővebben is beszélünk a 24. órán), de általában véve is segít az előreláthatólag gyakran frissítendő tartalom nyomon követésében, ha a webkiszolgálón rendezett könyvtárszerkezetet tartunk fenn. Ha például külön könyvtárat hozunk létre a képek vagy a multimédiás fájlok számára, pontosan tudni fogjuk, hol keressük a frissíteni kívánt fájlt – nincs szükség tehát arra, hogy más tartalmat tároló könyvtárakban kutakodjunk.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Melyik három információ szükséges ahhoz, hogy FTP-n keresztül kapcsolódhassunk a webkiszolgálónkhoz?
2. Mi a szerepe az `index.html` fájlnak?
3. Muszáj a webhelyünknek könyvtárszerkezetet tartalmaznia?

Válaszok

1. Az állomásnév, a fiókunkhoz tartozó felhasználónév, és a fiókunk jelszava.
2. Az `index.html` fájl általában egy könyvtár alapértelmezett fájlja a webkiszolgálón, amely lehetővé teszi, hogy a `http://www.tartomany.com/valamilyen_konyvtar/` címet fájlnev nélkül érjük el, és mégis a megfelelő helyre kerüljünk.
3. Nem. A fájlok könyvtárszerkezetben való elrendezése egyáltalán nem kötelező, de melegen ajánlott, mert egyszerűbbé teszi a tartalom karbantartását.

Gyakorlatok

- Az FTP-ügyfélprogramunk segítségével hozzunk létre egy alkönyvtárat a webhelyünk dokumentumgyökerében. Másoljuk le és illesszük be a `sample.html` fájl tartalmát egy másik, `index.html` nevű fájlba, és módosítsuk a `<title>` és `</title>`, illetve a `<h1>` és `</h1>` címkék között álló szöveget valami másra. Mentsük a fájlt, és töltsük fel az új alkönyvtárba. A webböngészőnkkel nyissuk meg a webkiszolgálón az újonnan létrehozott könyvtárat, és vegyük észre, hogy az `index.html` fájl tartalma jelenik meg. Ezt követően az FTP-ügyfelünk segítségével töröljük az `index.html` fájlt a távoli alkönyvtárból, majd térjünk vissza az URL-re a böngészővel, töltsük újra az oldalt, és figyeljük meg, hogyan reagál a kiszolgáló az `index.html` fájl hiányában.
- Hozzuk létre ismét a fenti gyakorlatban használt fájlokat, és helyezzük el azokat valamilyen hordozható eszközön, például CD-ROM-on vagy USB-kulcsan. A böngészőnkkel nyissuk meg a példawebhelyünknek ezt a helyi változatát, és gondoljuk végig, milyen használati utasítást kellene adnunk a hordozható eszköz mellé a felhasználóknak.



3. ÓRA

A HTML és az XHTML alapjai

A lecke tartalma:

- Egyszerű HTML-oldalak készítése
- Hogyan helyezhetünk el minden HTML-elemet, amelyet minden weboldalnak tartalmaznia kell?
- A weboldalak elrendezése bekezdésekkel és sortörésekkel
- A tartalom tagolása címsorok segítségével
- A tartalom érvényességének ellenőrzése
- A HTML, az XML, az XHTML és a HTML 5 közötti különbségek

Az első két órán megismerkedtünk a webes tartalom létrehozásának alapjaival, és megtudtuk, hogyan jeleníthetjük meg a munkánkat a hálózaton, illetve – amennyiben nincs tárhelyszolgáltatónk – a saját gépünkön. A következőekben megtanuljuk, melyek azok az elemek, amelyeknek feltétlenül meg kell jelenniük a HTML-fájljainkban.

Az óra végére tisztázzuk, mi a különbség a HTML és az XHTML között, és arra is választ kapunk, hogy miért létezhet egymás mellett két nyelv, amelynek a célja ugyanaz – webes tartalom létrehozása. Összefoglalva, ezen az órán gyors áttekintést adunk a HTML és

az XHTML alapjairól, és némi útmutatást nyújtunk ahhoz, hogy miként érdemes eljárunk a weboldalaink készítése és közzététele során. Az elmélet mellett gyakorlati tanácsokat is kapunk: látni fogunk egy valódi weboldalt és a működését biztosító HTML-kódot.

Mindenekelőtt azonban tekintsük át, hogy mire van szükségünk ahhoz, hogy követni tudjuk a könyv további részét:

1. Szerezzünk egy számítógépet. Jómagam egy Windows Vista rendszerrel ellátott gépet használtam a webes mintatartalom tesztelésére és a könyvben szereplő képernyőfelvételek elkészítésére, de ez nem jelent megkötést – a saját webes tartalmaink elkészítéséhez és megjelenítéséhez akármilyen Windows, Macintosh vagy Linux/UNIX rendszer megteszi.
2. Csatlakozzunk az Internethez. Az, hogy betárcsázós, drót nélküli vagy szélessávú kapcsolattal, a weboldalaink létrehozása és megjelenítése szempontjából lényegében mindegy, jöllehet minél gyorsabb a kapcsolat, annál jobb lesz a felhasználói élmény. A beállításban segíthet az internetszolgáltatónk, iskolánk vagy cégünk, de rengeteg olyan nyilvános hely – kávézó, könyvesbolt és könyvtár – is létezik, ahol ingyenes drót nélküli kapcsolatot ajánlanak, így ha a laptopunk WiFi-képes, máris nyert ügyünk van.



Ha tanácstalanok vagyunk az internetszolgáltató kiválasztásában, az a legjobb, ha ellátogatunk egy összehasonlító oldalra (egy barátunk gépéről vagy egy nyilvános, internetkapcsolattal rendelkező számítógépről). Ilyen oldal az Egyesült Államokban a <http://www.thelist.com/>, Magyarországon pedig a <http://www.internetszolgáltatok.hu>.

3. Szerezzünk be egy böngészőt. Egy ilyen programra feltétlenül szükségünk lesz ahhoz, hogy a számítógépünkre letöltsük és ott megjelenítsük a webes tartalmakat. Az első órából már tudjuk, hogy a legnépszerűbb böngészők (ábécésorrendben) a következők: Apple Safari, Google Chrome, Mozilla Firefox, Microsoft Internet Explorer és Opera. Érdemes többet is telepíteni közülük, így meggyőződhetünk róla, hogy a weboldalaink tartalma egységes képet mutat mindegyikükön – nem élhetünk semmiféle feltételezéssel arra nézve, hogy milyen böngészőket használnak majd a látogatóink.
4. Induljunk felfedezőútra! A böngészőnk segítségével nézzünk körül az Interneten, és keressünk olyan webhelyeket, amelyeknek a tartalma vagy megjelenése hasonlít ahhoz, amit magunk is szeretnénk. Jegyezzük fel, hogy mi idegesít az egyes oldalakon, mi csábít mások olvasására, és mi az, ami miatt újra és újra visszatérnénk valahová. Ha van olyan téma, ami különösen érdekel minket, keressünk rá valamelyik ismertebb kereső, például a Google (<http://www.google.com/>) vagy a Bing (<http://www.bing.com/>) segítségével.



Jóllehet a böngészők általánosságban hasonlóan kezelik és dolgozzák fel a kapott adatokat, akadnak közöttük apró különbségek, amelyek miatt az oldalak nem mindig ugyanúgy jelennek meg a böngészőablakban. Ezért fontos, hogy a munkánkat többféle böngészőben is megismerjük, hogy biztosak lehessünk benne, hogy jobbra egységes képet kapunk.



Amint arról az 1. órán szót ejtettünk, ha az Interneten szeretnénk webes tartalmat közzétenni (nem pedig lemezen vagy helyi hálózaton), egy olyan számítógépre kell feltöltenünk azt, amely 24 órás internetkapcsolattal rendelkezik. Ha az internetszolgáltatást az iskolánktól vagy munkahelyünkötől kapjuk, meglehet, hogy webtárhelyet is igénybe vehetünk ugyanitt – ha mégsem, úgy meg kell rendelnünk a tárhelyszolgáltatást.



Az első HTML-fájlunkat semmiképpen se Microsoft Wordben vagy más, HTML-képes szövegszerkesztőben készítsük el – ezek ugyanis „segíteni” próbálnak, és a maguk szája íze szerint átírják a kódunkat, ami csak ronthat a helyzeten. Hasonlóképpen nem tanácsoljuk az „azt kapod, amit láatsz” (WYSIWYG) típusú grafikus szerkesztők (amilyen például a Microsoft FrontPage vagy az Adobe Dreamweaver) használatát. A HTML nyelv elsajátításában sokkal nagyobb segítséget adnak az egyszerű szövegszerkesztők, hiszen így kénytelenek vagyunk a kóddal dolgozni. Később persze, ha már tudjuk, mi folyik a háttérben, használatba vehetjük a grafikus segédeszközöket (mint a FrontPage vagy a Dreamweaver).

Az egyszerű weboldalak készítésének alapjai

Az első órán megtanultuk, hogy a weboldal nem más, mint egy egyszerű szövegfájl, amelynek a tartalmát HTML-kódok „jelölik”, meghatározva ezzel a böngésző számára, hogy miként jelenítse meg a szöveget. Ezeknek a szövegfájloknak az elkészítéséhez egy *egyszerű szövegszerkesztőre* van szükségünk – ez lehet a Windows Jegyzettömbje (Notepad) vagy Macintosh rendszereken a TextEdit. Ne használjuk a WordPadet, a Microsoft Wordöt vagy más, hasonló fejlett szövegszerkesztőt, ezek ugyanis nem olyan egyszerű szövegfájlokat adnak végeredményként, amilyeneket a webes tartalom megjelenítése megkövetel.

Mielőtt nekikezdenénk a kódolásnak, készítsük el azt a szöveget, amelyet meg szeretnénk jeleníteni egy weboldalon:

1. Keressünk (vagy írjunk) néhány sort magunkról, a családunkról, a focicsapatunkról vagy más, minket érdeklő témáról.
2. Mentsük a szöveget egy egyszerű, szabványos ASCII fájlba. A Jegyzettömb és a hozzá hasonló egyszerű szövegszerkesztők automatikusan egyszerű szöveges formátumban mentenek, más programok esetében azonban előfordulhat, hogy ki kell választanunk ezt a fájltypust a Fájl, Mentés másként (File, Save as) menüponttal.

Ahogy haladunk az óránk anyagával, úgy gazdagítjuk a szöveget HTML-jelölésekkel (HTML-elemekkel), amelyek valódi webes tartalommal alakítják azt.

Ha HTML-elemeket tartalmazó fájlokat mentünk, mindig a `.html` kiterjesztést kell használnunk. Ez fontos, hiszen ha megfeledekezünk erről, a legtöbb egyszerű szövegszerkesztő valamilyen más kiterjesztést (például `.txt`) választ. Ha pedig így áll a helyzet, a fájlnkat nem biztos, hogy megtaláljuk a böngészővel, ha pedig igen, a megjelenítése nem lesz tökéletes. Röviden: a böngészők arra számítanak, hogy `.html` kiterjesztéssel jutnak hozzá a weboldalakhoz.



Ha Macintosh rendszeren a TextEdittel dolgozunk, a HTML-fájlok elkészítésének lépései némileg eltérnek a Windows Jegyzettömbnél látottaktól – itt ugyanis a fájl mentése előtt ki kell választanunk a Format (Formátum) menü Make Plain Text (Sima szöveg létrehozása) pontját, valamint a Saving (Mentés) beállításoknál ki kell kapcsolnunk az Append `'txt'` Extension to Plain Text Files (A `.txt` kiterjesztés hozzáfűzése a sima szövegfájlokhoz) jelölőnégyzetet. Emellett, az alapértelmezett beállítások szerint a szerkesztő úgy jeleníti meg a `.html` fájlokat, ahogy a böngészőben látnánk őket, ami nem teszi lehetővé a szerkesztésüket. Ennek megakadályozására kapcsoljuk be az Ignore Rich Text Commands in HTML Files (Szövegformázó parancsok figyelmen kívül hagyása a HTML-fájlokban) jelölőnégyzetet a beállítások Rich Text Processing (Formázott szövegek feldolgozása) részében.

Olyan weboldallal is találkozhatunk, amelyeknek a kiterjesztése `.htm`, ami szintén megengedett. Egyéb kiterjesztésekkel is találkozhatunk a Weben, mint a `.jsp` (Java Server Pages), az `.asp` (Microsoft Active Server Pages) vagy a `.php` (PHP: Hypertext Preprocessor), ezek a fájlok azonban kiszolgálóoldali eljárásokat használnak, amelyek túlmutatnak a HTML világán.

A 3.1. példában egy egyszerű weboldal HTML-kódját láthatjuk, amelyet akár be is írhatunk a szerkesztőnkbe. Ha megnyitjuk az oldalt a Firefoxban, a 3.1. ábrán látható kép tárul elénk a böngészőablakban. Minden weboldalon fel kell tüntetnünk a `<html></html>`, `<head></head>`, `<title></title>` és `<body></body>` címképarákat.

3.1. példa A `<html>`, `<head>`, `<title>` és `<body>` címkék

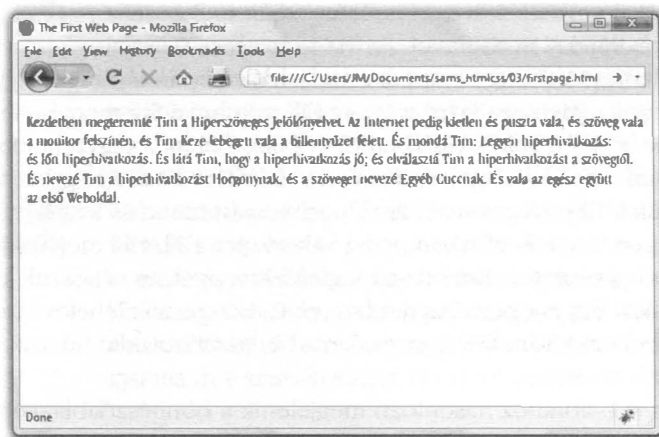
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>The First Web Page</title>
  </head>
```

```

<body>
  <p>
    Kezdetben megteremté Tim a Hiperszöveges Jelölőnyelvet. Az Internet
    pedig kietlen és pusztá vala, és szöveg vala a monitor felszínén,
    és Tim Keze lebegett vala a billentyűzet felett. s mondá Tim:
    Legyen hiperhivatkozás: és lőn hiperhivatkozás. s látá Tim, hogy
    a hiperhivatkozás jó; és elválasztá Tim a hiperhivatkozást a szö-
    vegtől. s nevezé Tim a hiperhivatkozást Horgonynak, és a szöveget
    nevezé Egyéb Cuccnak. s vala az egész együtt az első Weboldal.
  </p>
</body>
</html>

```



3.1. ábra

Ha a 3.1. példa tartalmát egy HTML-fájlba mentjük, majd megjelenítjük egy böngészőben, csak a cím és a törzs szövegét látjuk

A 3.1. példában, hasonlóan minden más HTML oldalhoz, a < jellel kezdődő és > jellel befejeződő szavak kódolt parancsok. Ezeket nevezzük HTML-címkéknek (tag), amelyek „felcímkézik” az egyes szövegrészeket, elárulva így a böngészőnek, hogy milyen fajta szövegről van szó – ez teszi lehetővé a megfelelő megjelenítést.



Elméletileg a következő webes szabványt a HTML 5 jelenti, amely azonban még korántsem terjedt el széles körben. A teljes váltásra nagyjából 2011-ben kerül majd sor. Mindazonáltal a HTML és az XHTML lehetőségeinek bemutatásánál indokolt esetben kitérünk arra, hogy miben különbözhet a HTML 5-ös változat.

Az első néhány kód sor általános bevezető, amelynek minden weboldalon szerepelnie kell. A jelentése egyszerűen annyi, hogy az oldal egy XHTML 1.1 típusú dokumentum, vagyis egy XHTML-oldal – hasonlóan a könyvünkben szereplő összes többi weboldallalhoz. Mivel az XHTML nem más, mint a HTML strukturáltabb változata, nem követünk el nagy hibát, ha HTML-oldalakként hivatkozunk ezekre az oldalakra. Az XHTML 1.1 megjelölésével olyan weboldalakat készítünk, amelyek követik a legfrissebb webes szabványokat, ami igen üdvös dolog.



Önálló feladat

Egyszerű weboldal létrehozása és megjelenítése

Mielőtt megtanulnánk a 3.1. példában alkalmazott HTML-elemek jelentését, érdemes pár szót szólni arról, hogyan is készült a fenti dokumentum, és miként jelenítettük meg. Kövessük hát az alábbi lépéseket:

1. Gépeljük be a 3.1. példa tartalmát a Windows Jegyzetömbbe (illetve a Macintosh TextEdit vagy más, számunkra kedves szövegszerkesztő ablakába).
2. Válasszuk a Fájl, Mentés másként menüpontot. Fontos, hogy fájltypusként az egyszerű szöveg (vagy ASCII szöveg) beállítást adjuk meg.
3. Adjuk a fájlnek a `firstpage.html` nevet.
4. Válasszunk a merevlemezünkön egy mappát a weboldalaink tárolására, és jegyezzük meg jól. Kattintsunk a Mentés (Save) vagy az OK gombra a fájl mentéséhez.
5. Indítsuk el a kedvenc böngészőnket. (Közben nyitva hagyhatjuk a Jegyzetömböt, így könnyen válthatunk az oldal megtekintése és szerkesztése között.) Az Internet Explorerben válasszuk a Fájl, Megnyitás (File, Open) menüpontot, és kattintsunk a Tallózás (Browse) gombra, a Firefoxban pedig válasszuk a Fájl, Fájl megnyitása (File, Open File) menüpontot. Keressük fel a megfelelő mappát, és válasszuk ki a `firstpage.html` fájlt. Egyes operációs rendszerek és böngészők lehetővé teszik azt is, hogy a megjeleníteni kívánt fájl egyszerűen a böngészőablakba húzzuk.

És tessék! A weboldalunk a 3.1. ábrához hasonlóan megjelenik a böngészőablakban.

Ha hozzájutottunk egy webtárhelyhez, az FTP segítségével feltölthetjük ide a `firstpage.html` fájlt. Ez a megjegyzés azért is lényeges, mert a könyv további része feltételezi, hogy rendelkezünk webtárhellyel, és nem okoz gondot a fájlok átvitele az FTP segítségével – ha mégis, hát lapozzunk vissza az első két óra anyagához, mielőtt továbblépnénk. Ha pedig tudatosan döntünk úgy, hogy helyben dolgozunk a fájlokkal (webtárhely nélkül), készüljünk fel arra, hogy a könyv útmutatásait a saját igényeinkhez igazítsuk (figyelman kívül hagyva például a „töltsük fel a fájlokat” és az „adjuk meg az URL-t” típusú utasításokat).



Ha egy a saját számítógépünkön tárolt weboldalt szeretnénk megnyitni, nem kell feltétlenül csatlakoznunk az Internethez. A böngésző alapértelmezésben persze indításkor megpróbálkozik ezzel, ami a legtöbb esetben teljesen ésszerű – ugyanakkor, ha a helyi merevlemezünkön szerkesztjük az oldalt, idegesítővé válhat, hogy állandóan nem elérhető oldalakról hallunk. Ha állandó internetkapcsolattal rendelkezünk LAN, kábelmodem vagy DSL szolgáltatás révén, mindez nem okoz gondot, hiszen a böngésző sohasem fog elérhetetlen oldalakra panaszkodni. Egyébként viszont meg kell rendszabályoznunk a böngészőt, amiben az Eszközök (Tools) menü lehet a segítségünkre.

Az XHTML-oldalakon kötelezően felhasználandó HTML-elemek

Elérkezett az idő, hogy megismerkedjünk a HTML-elemek „titkos nyelvével”. E tudás birtoklása olyan kreatív erőket szabadít fel bennünk, amelyekről az egyszerű emberi lények álmodni sem mernek. Ne áruljuk el nekik, hogy valójában igen egyszerű dologról van szó...



Ezen a ponton nem feltétlenül muszáj tisztában lennünk az olyan fogalmakkal, mint a karakterkódolás. A lényeg, hogy a weboldalainkon elhelyezzük a megfelelő azonosítókódot, amellyel igazodhatunk a legfrissebb webes szabványokhoz – könyvünk írásának idején ez az XHTML 1.1-et jelenti. A HTML 5 még nem tekinthető webes szabványnak, azt azonban érdemes tudnunk, hogy ha HTML 5 dokumentumot készítünk, ezekre a bevezető sorokra nincs szükség.



Az XML/XHTML azonosítókódra nincs feltétlenül szükség ahhoz, hogy működőképes weboldalt hozzunk létre. Erről könnyen meggyőződhetünk – csak töröljük a bevezető sorokat, amíg olyan kódot nem kapunk, amely a `<html>` címkével kezdődik – az eredményt továbbra is gond nélkül megnyithatjuk a böngészőben. A kihagyott kód arra szolgál, hogy megfeleljünk az érvényben levő webes szabványoknak – emellett lehetővé teszi, hogy az érvényesség ellenőrzésével megvizsgáljuk a kód igazodását a szabványokhoz, amint azt hamarosan láthatjuk is.

Mielőtt megismerkednénk a HTML-elemekkel, nézzük meg, mit is takar a 3.1. példa meglehetősen zavaros első sora. Nos, ez a sor azt jelzi, hogy a HTML-oldal valójában egy XML-dokumentum:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Az XML 1.0-s változatát alkalmazzuk, amely széles körben elfogadott, hasonlóan az UTF-8 karakterkódoláshoz. A 3.1. példa második és harmadik sora – ha lehet – még bonyolultabb:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Ezeknek a soroknak a pontos tartalma nem igazán lényeges a számunkra; mindössze arra kell ügyelnünk, hogy elhelyezzük őket minden weboldalunk elején. A kód mindössze arra utal, hogy a dokumentum XHTML 1.1 típusú, így a böngészők feltételezhetik, hogy megfelel az XHTML 1.1 szabvány követelményeinek.

A legtöbb HTML-elem két részből áll: a *nyitócímké* jelzi, hogy hol kezdődik az elem hatálya alatt álló szövegrészlet, a *zárócímké* pedig azt mutatja, hogy hol végződik. A zárócímkéket onnan ismerhetjük fel, hogy a `<` jel után egy `/` (perjel) áll a kódjukban.

Léteznek ezen felül *üres elemek* is, amelyek önmagukban állnak, nyitó- és zárócímke nélkül – ez esetben az elem kódját lezáró `>` előtt láthatjuk a `/` jelet. Az alábbiakban röviden összefoglaljuk e három címketípus szerepét:

- A *nyitócímke* egy HTML-utasítás kezdetét jelzi – a szöveg, amelyre az utasítás hat, a nyitócímke után áll. A nyitócímkek a `<` jellel kezdődnek, és a `>` jellel fejeződnek be (például: `<html>`).
- A *zárócímke* egy HTML-utasítás végét jelzi – a szöveg, amelyre az utasítás hat, a zárócímke előtt áll. A zárócímkek elején mindig a `</`, a végén pedig a `>` jel áll, (például: `</html>`).
- Az *üres címke* egy HTML-utasítást önmagában hajt végre, anélkül, hogy valamilyen szövegre hatna. Az üres elemek kezdetét a `<`, a végét pedig a `/>` jelöli, mint a `
` vagy az `` esetében.



Bizonyára észrevettük, hogy a 3.1. példában a `<html>` címkéhez némi további kód is kapcsolódik, nevezetesen két jellemző (az `xmlns` és az `xml:lang`), amelyek további, az elemhez kapcsolódó adatokat tartalmaznak. E két jellemző megadása szabványos követelmény minden XHTML-weboldalon – az előbbi az XML-névteret, míg az utóbbi a tartalom nyelvét adja meg. Ebben a könyvben a szabványos névteret használjuk, valamint az angol nyelvet. Ha más nyelven szeretnénk írni, cseréljük az `"en"` (English) kódot a megfelelő nyelv kódjára.



Érdekes lehet tárolnunk egy *oldalvázat*, amelyben csak a nyitó és záró `<html>`, `<head>`, `<title>` és `<body>` címkék szerepelnek – nagyjából úgy, mint a 3.1. példában. A későbbiekben ezt a dokumentumot kiindulópontként használhatjuk a weboldalaink elkészítéséhez, így megtakarítjuk a kötelező címkék beírására fordított időt.

A 3.1. példa `<body>` címkéje például arról árulkodik, hogy hol kezdődik a weboldal törzse, a `</body>` címke pedig ennek végét jelöli. Minden, ami a `<body>` és a `</body>` címkék között áll, megjelenik a böngészőablakban, amint az a 3.1. ábrán is látható.

A böngészőablak legtetején (lásd a 3.1. ábrát) találhatjuk a címet – ez pontosan megegyezik a `<title>` és a `</title>` címkék közötti szöveggel. A cím jelenik meg a weboldalhoz a böngésző Kedvencek (Favorites), illetve Könyvjelzők (Bookmarks) menüjében hozzárendelt bejegyzésében is (attól függően, hogy melyik böngészőt használjuk). Fontos, hogy az oldalainknak megfelelő címet adjunk, hiszen így a látogatóink a könyvjelzőik alapján később is emlékezni fognak a tartalmára.

A `<body>` és a `<title>` címke minden weboldal kódjában megjelenik, hiszen mindig szükség van törzsszövegre és címre. Hasonlóképpen, a `<html>` és a `<head>` címke megjelenése is általános (lásd a 3.1. példát). A `<html>` címkét egy oldal kódjának elején elhelyezve egyszerűen arra utalhatunk, hogy itt egy weboldaltól van szó, amely a `</html>` címkével ér véget.

Minden weboldal egy fejlécből és egy törzsből áll, amelyeket a `<head>`, illetve a `<body>` elemek határoznak meg. A fejléc alapvetően olyan adatok tárolására szolgál, amelyek meghatározzák az oldal megjelenését, ugyanakkor maguk nem válnak láthatóvá a böngészőablakban – ellentétben a `<body>` elem tartalmával. A `<head>` elem mindig a weboldal HTML-kódjának elején szerepel, rögtön a nyitó `<html>` címke után.

A címet azonosító `<title>` címkepár az oldal fejlécén belül jelenik meg, vagyis a nyitó `<head>` és a záró `</head>` címke között. (A későbbi órákon további elemekkel is megismerkedünk, amelyek a `<head>` és a `</head>` címkék közé kerülhetnek; ide tartoznak például a stílusszabályok.)

A 3.1. példában felbukkanó `<p>` elem egy szövegbekezdésre utal. Fontos, hogy amennyiben lehetséges, minden szövegrészletet valamilyen tárolóelemben helyezzünk el.

Oldalszerkezet kialakítása bekezdésekkel és sortörésekkel

A böngésző a HTML-oldalak megjelenítésekor nincs tekintettel a sorvégekre vagy a szóközök számára. Így például a 3.2. példában szereplő versnek a 3.2. ábrán látható, felső változatában a szavakat egy-egy szóköz választja el, jöllehet a kódban ezt nem így adtuk meg. A „felesleges” szóközöket a HTML-kód értelmezője lefaragja, ráadásul a sortörések akkor következnek be, ha a szöveg kiér a böngészőablak szélére – teljesen függetlenül a sajtó, eredeti sortöréseinktől.

3.2. példa *HTML-kód bekezdésekkel és sortörésekkel*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

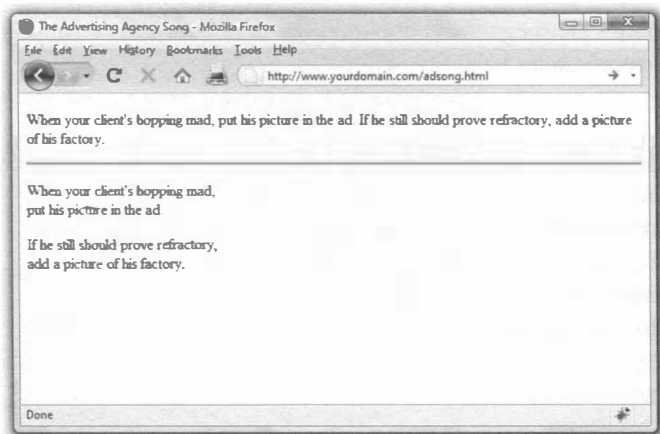
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>The Advertising Agency Song</title>
  </head>
  <body>
    <p>
      When your client's hopping mad,
      put his picture in the ad.

      If he still should prove refractory,
      add a picture of his factory.
    </p>
    <hr />
```

```

<p>
  When your client's hopping mad,<br />
  put his picture in the ad.
</p>
<p>
  If he still should prove refractory,<br />
  add a picture of his factory.
</p>
</body>
</html>

```



3.2. ábra

*Ha a 3.2. példa HTML-kódját weboldalként jelenítjük meg, bekezdéseket és sortöréseket csak a <p> és
 elemeknek megfelelően kapunk*



Rengeteg olyan weboldalt találhatunk, ahol a
 helyett csak
 áll, illetve a <p> címkének nincs záró </p> párja. Sose feledjük, hogy az Interneten rengeteg trehány munka kering, és attól, hogy látunk egy kódot, még nem kell feltétlenül helyesnek lennie. Ha a könyvben lefektetett kódolási elvekhez tartjuk magunkat, rengeteg jövőbeni munkától és bosszankodástól kímélhetjük meg magunkat. A helyes HTML-kódolási szokások kialakítása komoly részét képezi annak a folyamatnak, amelynek során sikeres webtervező válik belőlünk.

Ahhoz, hogy befolyásoljuk, hol törjenek a sorok és a bekezdések, HTML-elemeket kell használnunk. Ha egy szövegrészletet a <p></p> tárolócímkék között helyezünk el, a zárócímke után egy sortörésre számíthatunk. A következő órákon megtanuljuk majd, hogyan befolyásoljuk a sortörés magasságát a CSS segítségével. A
 elem egy sortörést kényszerít ki a bekezdésen belül. Az eddig látott elemektől eltérően a
 esetében nincs szükség záró </br> címkére – ez egyike a korábban említett üres elemeknek. Jóllehet a HTML 4 az üres elemek esetében nem követeli meg a / feltűnte-

tését, az XHTML és a jövőbeni szabványok igen. Fontos, hogy mindenben alkalmazkodjunk a legfrissebb szabványokhoz, és olyan weboldalat készítsünk, amelyeknek a kódja kifogástalan, így az üres elemeket minden esetben zárjuk a `</>` jelekkel.

A 3.2. példában és a 3.2. ábrán látható versben – amely egy reklámügynökség dalocskája – a `
` és a `<p>` elemekkel választottuk el a sorokat és a versszakokat. Észrevehettük ezen felül a `<hr />` elemet is, amely egy vízszintes vonalat helyez el az oldalon (lásd a 3.2. ábrát). Fontos tudnunk, hogy a `<hr />` elem egyúttal sortörést is okoz – még akkor is, ha nem tüntetünk fel mellette egy `
` elemet. A `
` elemhez hasonlóan a `<hr />` is üres, vagyis nem tartozik hozzá záró `<hr/>` címke.



Önálló feladat

Szöveg formázása HTML-kóddal

Vegyünk egy szövegrészletet, és próbáljuk HTML-ben formázni a frissen szerzett tudásunk segítségével:

1. Helyezzük el a szövegrészlet elején a `<html><head><title>My Title</title></head><body>` kódot (az általunk választott címet feltüntetve a *My Title* helyén). Adjuk meg emellett a megfelelő azonosítókat az oldal tetején, hogy megfeleljünk az XHTML szabvány követelményeinek.
2. Biggyesszük a `</body></html>` kódot a szöveg legvégére.
3. Helyezzünk egy-egy `<p>` címkét a bekezdések elejére és egy-egy `</p>` címkét a végükre.
4. Alkalmazzuk a `
` elemet, ahol csak egyszeres sorközü sortörést szeretnénk.
5. A `<hr />` segítségével válasszuk el egymástól vízszintes vonalakkal a szöveg nagyobb részeit.
6. Mentsük lemezre a fájlt `mypage.html` néven (a *mypage* helyett nekünk tetsző nevet adva neki).
7. Nyissuk meg a fájlt egy böngészőben a tartalma megjelenítéséhez. (Előbb töltsük fel FTP-vel a webtárhelyünkre, amennyiben rendelkezünk ilyennel.)
8. Ha valami nem úgy fest, ahogy szeretnénk, térjünk vissza a szövegszerkesztőhöz, és módosítsuk a fájl kódját (majd pedig, ha van webtárhelyünk, töltsük fel ismét). A változások megjelenítéséhez kattintsunk a böngésző Frissítés (Reload/Refresh) gombjára.



Ha komolyabb szövegszerkesztőt használunk a weboldal elkészítésére, győződjünk meg arról, hogy valóban egyszerű szöveges vagy ASCII formátumban mentjük a HTML-fájlt.

A tartalom tagolása címsorok segítségével

Ha weboldalakat nézegetünk az Interneten, feltűnhet, hogy sokuk esetében a lap tetején található címsor nagyobb és vastagabb betűkkel jelenik meg, mint a szöveg többi része. A 3.3. példa jól mutatja a címsor és a bekezdések szövegének eltérését. Ha egy szövegrészlet a `<h1>` és a `</h1>` címkék közé kerül, automatikusan nagy méretű címsorként jelenik meg. A `<h2>` és a `<h3>` elemek szintén címsorokat határoznak meg, de a betűméretük fokozatosan csökken, egészen a `<h6>` elemig.

3.3. példa Címsorelemek

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>My Widgets</title>
  </head>

  <body>
    <h1>My Widgets</h1>
    <p>My widgets are the best in the land. Continue reading to
      learn more about my widgets.</p>

    <h2>Widget Features</h2>
    <p>If I had any features to discuss, you can bet I'd do
      it here.</p>

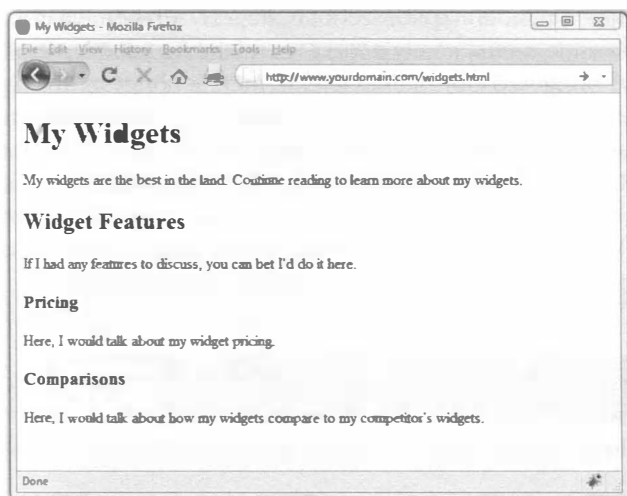
    <h3>Pricing</h3>
    <p>Here, I would talk about my widget pricing.</p>

    <h3>Comparisons</h3>
    <p>Here, I would talk about how my widgets compare to my
      competitor's widgets.</p>

  </body>
</html>
```



Az eddigi példák kapcsán bizonyára feltűnt, hogy a szerző a kód behúzásával kísérte meg érzékeltetni a HTML-dokumentum különféle részei közötti kapcsolatokat. A behúzások használata azonban egyáltalán nem kötelező – ha úgy tartja kedvünk, az elemeket össze is ömleszthetjük szóközök és sortörések nélkül; az eredmény a böngészőben így is megegyezik az előzőekben látottakkal. A behúzások kizárólag a kód olvasójának segítenek abban, hogy jobban átlássa az előtte álló weboldal szerkezetét. A megfelelő behúzások rendszeres használata tehát helyes gyakorlat, és sokat segít abban, hogy fenntartható kódhoz jussunk.



3.3. ábra

Termékbemutató weboldalunk tartalmát a címsorok három szintjével tagoljuk

Amint a 3.3. ábrán is láthatjuk, a HTML-címsorok kódja ennél egyszerűbb nemigen lehetne. Példánkban a „My Widgets” feliratot láthatóan a `<h1>` elemmel jelenítettük meg. A legnagyobb méretű (1. szintű) címsor létrehozásához nem kell mást tennünk, mint elhelyezni a cím elején a `<h1>`, a végén pedig a `</h1>` címkét. Ha némiképp kisebb (2. szintű) címsort szeretnénk – amely a főcímnél alacsonyabb rangú szerepet játszik –, alkalmazzuk a `<h2>` és `</h2>` címkéket a cím körül. Ha ennél is alacsonyabb szintű címsorra vágyunk, a `<h3>` és `</h3>` címkék segíthetnek. Fontos, hogy a címsoraink kövessék a tartalom hierarchiáját, vagyis egyetlen 1. szintű címsort tüntessünk fel, amely után egy vagy több 2. szintű címsor áll, ezeket kövessék a 3. szintű címsorok, és így tovább.

Elméletben a további, még alacsonyabb rangú címsorokhoz használhatjuk a `<h4>`, `<h5>` és `<h6>` elemeket is, de erre a legtöbbször nem kerül sor. A böngészőkben ritkán láthatunk számottevő különbséget a `<h3>` és az alacsonyabb rangú címsorok megjelenítésében, ráadásul a tartalom általában nem olyan alakban jelenik meg, hogy az elrendezéséhez hatféle címsorra lenne szükség.



Napjainkban rengeteg webhelyen láthatunk grafikusan kidolgozott betűkből álló, képként beágyazott szövegeket a hagyományos szöveges címsorok helyén. Érdemes azonban tudnunk, hogy a szöveges címsorok használata egyike a keresőoptimalizálási fogásoknak, amelyekről a 24. fejezetben olvashatunk majd. A keresők sorra veszik a weboldalak címsorelemeit, hogy képet kapjanak a tartalom felépítéséről, és nagyobb hangsúlyt adnak a fontosabb (például 1. szintű) címsoroknak, mint azoknak, amelyeket magunk is kisebb jelentőségűnek tartunk.

Fontos felhívunk a figyelmet a *cím* (title) és a *címsor* (heading) közötti különbségre. Ezeket könnyen összekeverhetjük a mindennapi beszédben, de a HTML-ben határozottan különbséget kell tennünk köztük: a *cím*, amely a `<title>` elemben jelenik meg, a teljes oldalt azonosítja –csak a böngésző címsorában jelenik meg, a böngészőablakban nem. A *címsorok* ugyanakkor arra alkalmasak, hogy a segítségükkel bizonyos szövegrészeket kiemeljünk a környezetükből. Egy weboldalon egyetlen `<title>` elem szerepelhet, amelynek a `<head>` és a `</head>` címke közé kell kerülnie. Ugyanakkor a `<h1>`, `<h2>` és `<h3>` elemekből annyit és olyan sorrendben használhatunk, ahogy csak szeretnénk. Mindazonáltal, ahogy a korábbiakban már említettük, fontos, hogy a címsorokat valóban a tartalom hierarchikus tagolására használjuk, ne pedig egyszerűen vizuális elemként – arra ott vannak a CSS-stílusok.

A szövegrészek megjelenésének részletes és teljes körű szabályozásáról könyvünk II. részében olvashatunk bővebben. Amíg nem vagyunk e tudás birtokában, a címsorok adják a legegyszerűbb és legerjedtebb módot arra, hogy a figyelmet bizonyos szövegrészekre irányítsuk.



Ne feledjük, hogy amit a weboldal fejlécében helyezünk el, azt nem megjelenítésre szánjuk, viszont minden, ami a weboldal törzsében áll, megjelenik a böngészőablakban.

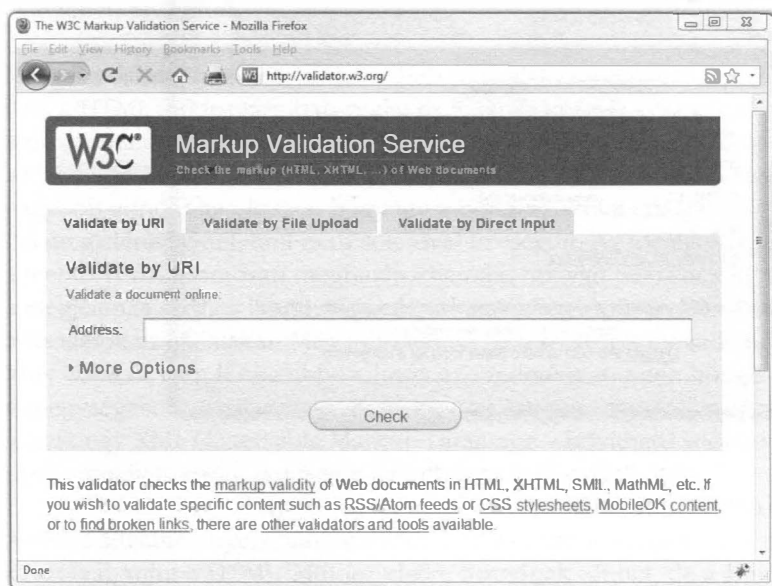
Lessük el a webtervezők fogásait!

Ha kicsit is körbenézünk napjaink csillogó-villogó, grafikával és olykor hanggal gazdagon felszerezett weboldalai körében, bizonyára ráébredünk, hogy az itt bemutatott egyszerű oldalak csak egy hatalmas HTML-jéghegy icipici csúcsát jelentik. Az alapok ismeretében azonban magunk is meglepődünk majd, milyen sok fogást leshetünk el mások munkáiból. Ha egy oldal HTML-kódjára vagyunk kíváncsiak, nem kell mást tennünk, mint a böngészőnk Nézet (View) menüjének Forrás (Source) pontját választani. Ne essünk kétségbe, ha ebben a pillanatban nem értjük minden HTML-elem jelentését, illetve nem boldogulunk a használatukkal – a következő pár órában minden világossá válik. A kukucskálásnak azonban már most is van értelme, hiszen így legalább látásból megismerünk néhány olyan elemet, amelyről a későbbiekben tanulunk majd, és képet kapunk arról, hogy mire is leszünk képesek az új tudásunk birtokában.

A webes tartalom érvényességének vizsgálata

A 2. órán a weboldalaink tesztelésének módjairól beszéltünk, amelyek közül az egyik legfontosabb az *érvényességvizsgálat* volt. Gondoljunk csak bele: lehet, hogy merész és szép házakat tervezünk, de ahhoz, hogy valóban megépüljenek, egy építésznek is rá kell bólintania a tervekre, igazolva, hogy a szerkezet valóban megépíthető. A weboldalak esetében az érvényességvizsgálat hasonló ehhez – az „építész” azonban ez esetben nem egy személy, hanem egy alkalmazás.

Az érvényességvizsgálat folyamata abból áll, hogy egy erre a célra készített alkalmazás töviről hegyire átvizsgálja az oldalainkat, és meggyőződik róla, hogy nem tartalmaznak hibákat, és megfelelnek az XHTML szabvány szigorú követelményeinek. A mi szemszögünkből ez egy igen egyszerű eljárás – a World Wide Web Consortium (W3C) ugyanis elérhetővé tett egy internetes érvényességvizsgáló eszközt, amelyet a <http://validator.w3.org/> címen érhetünk el. Ha beírjuk a címet a böngészőnkbe, a 3.4. ábrán látható W3C Markup Validation Service oldalra jutunk.

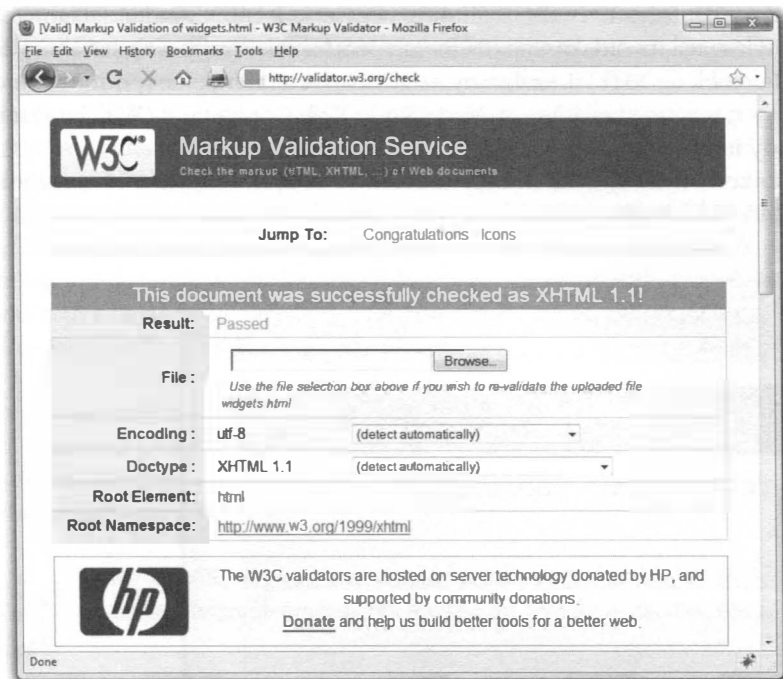


3.4. ábra

A W3C Markup Validation Service segítségével megvizsgálhatjuk a HTML- (XHTML-) dokumentumaink érvényességét, így meggyőződhetünk arról, hogy a kódjuk igazodik a szabványokhoz

Ha már közzétettük a weboldalunkat a hálózaton, válasszuk a Validate by URI (Ellenőrzés URI alapján) fület. A Validate by File Upload (Feltöltött fájl ellenőrzése) lapon a helyi fájlrendszerünkön tárolt weboldalak érvényességét vizsgálhatjuk. Végül, a Validate by Direct Input (Ellenőrzés közvetlen bemenetről) lapon a kérdéses fájl tartalmának részletét másolhatjuk be a vizsgálathoz. Ha minden rendben van a kóddal, az ellenőrzés végén a 3.5. ábrán látható pozitív visszajelzést kapjuk.

Amennyiben a W3C Markup Validation Service hibát talál egy weboldalon, részletes elemzéssel szolgál (pontosan megadva, hogy mely sorok sértik a szabályokat). Mindez nagyszerű lehetőséget ad arra, hogy felkutassuk a hibákat, és kigyomláljuk a nem megfelelő kódrészeket. Az érvényességvizsgálat tehát egy oldal szerkezetének ellenőrzése mellett segít kiszűrni és kijavítani a hibákat, mielőtt az oldalt közzétennénk az Interneten.



3.5. ábra

Ha a weboldalunk átment a W3C Markup Validation Service vizsgálatán, készen áll a közzétételre



Léteznek olyan webes fejlesztőeszközök, amelyek maguk is lehetőséget adnak az érvényességvizsgálatra, a W3C Markup Validation Service igénybe vétele nélkül. Vannak közöttük böngészőbővítmények, mint a Firebug (<http://getfirebug.com/>) vagy a HTML Validator (<http://users.skynet.be/mgueury/mozilla/>), de számos más program is szolgáltat hasonló lehetőségeket – ezekről az adott alkalmazás leírása tájékoztat.

HTML, XML, XHTML és HTML 5 – útmutató a szabványok dzsungeléhez

Az Internet hőskorában a HTML nagy találmánynak számított, hiszen lehetővé tette, hogy a tudósok kutatási anyagokat küldözgessenek egymásnak hatékony és viszonylag rendezett formában. Nem sokkal ezután azonban megjelentek a grafikus böngészők, és a HTML-t már kezdték a tudományos cikkeken túl más adatok megjelenítésére is használni. Így vált a HTML-ből a kutatók kellemes kis jelölőnyelvéből a hálózati közzététel

általánosan elfogadott nyelve. A böngészők készítői mindezen felbátorodtak, és rengeteg apró lehetőséggel egészítették ki a HTML-t. Ezek eleinte nagyszerűnek tűntek, de megtörték a HTML felépítésének egységét, így a böngészőkben ettől kezdve búcsút mondhattunk a weboldalak egységes megjelenítésének – az újdonságok egyes böngészőkben működtek, másutt nem, ha pedig „rossz” böngészőt választottunk, végül ott álltunk, ahol a part szakad. A HTML kezdett úgy kinézni, mint egy építkezés, amelyen megfelelő egységes terv nélkül rengeteg csapat dolgozik a saját szakállára. Végül egyes a böngészőkre jellemző lehetőségek bekerültek a szabványba, másokat pedig teljes egészében elvetettek.

Más forradalmakhoz hasonlóan a Web születése is meglehetősen kaotikus folyamat volt, a HTML módosításaiban pedig ez a káosz köszön vissza. Az évek során a programozók jelentős erőfeszítéseket tettek annak érdekében, hogy lenyesegezzék a HTML vadhajtasait, és kissé egységesebb képet adjanak a nyelvnek. A rendetlenségben a legnagyobb gondot az okozza, hogy így a böngészőknek találgatniuk kell egy-egy weboldal megjelenítésénél, ami nem sok jóval kecsegtet. Az ideális állapot az lenne, ha a webtervezők pontosan meghatározhatnák, hogyan nézzenek ki az oldalaik, és ez a megjelenés azonos lenne, függetlenül attól, hogy milyen böngészővel jelenítik meg a látogatók az oldalakat. Még jobb lenne, ha a tervezők megadhatnák az oldalak jelenítését is, és ez elég lenne ahhoz, hogy az eredmény minden böngészőben és rendszeren egységesen jelenjen meg. Ettől az utópisztikus állapottól még messze vagyunk, de létezik egy XML (Extensible Markup Language – bővíthető jelölőnyelv) nevű jelölőnyelv, amely komoly segítséget jelenthet az oda vezető úton.

Az XML általános nyelv, amely olyan, korlátozottabb rendeltetésű nyelvek létrehozására szolgál, mint a HTML. Mindez elsősre furcsának tűnhet, de a legkevésbé sem az, ha megértjük, hogy itt mindössze arról van szó, hogy az XML azt az alapszerkezetet és azokat az alapvető szabályokat rögzíti, amelyeket minden jelölő- vagy leírónyelvnek követnie kell. Mindennek ismeretében az XML segítségével magunk is létrehozhatunk egy új HTML-változatot. Akár egy BCCML (Bottle Cap Collection Markup Language – kupakgyűjtő jelölőnyelv) nevű jelölőnyelvet is kiötlölhetünk, amely segíthet a ritka kupakokból álló gyűjteményünk adatainak tárolásában és rendszerezésében. A lényeg az, hogy az XML megadja az alapokat az adatok egységes tárolásához – ezek az adatok pedig a kupakoktól a weboldalakig az élet bármely területéről származhatnak.

Azt gondolhatnánk, hogy a kupakoknak nincs sok közük a Webhez – de akkor miért említettük meg őket? Nos, ennek oka van, ugyanis az XML jelentősége túlmutat a weboldalakon: a segítségével tetszőleges típusú adatot leírhatunk bármilyen számítógépen. Ha felidézzük napjaink adatforrásainak gazdagságát a számítógépektől és mobiltelefonoktól a kézi számítógépeken át a rádió- és tévékészülékekig, világossá válhat, miért is szól többről az XML, mint weboldalak rendbe rakásáról. Mindazonáltal, az XML egyik első alkalmazása a Web kiemelését célozta a káoszból, ezért kap ez a nyelv fontos szerepet a HTML megértésében.

Ha viszont az XML alkalmasabb az adatok leírására, mint a HTML, miért nem használjuk webes jelölőnyelvként? Nos, az XML nem alkalmas a HTML helyettesítésére, és nem is versenytársa annak. Az XML hatása a HTML-re mindössze a rendezettség elérésében érvényesül – a HTML viszonylag laza szerkezetű nyelv, amely kiaknázza az XML szabályait. A két nyelv természetes egyesülése azt eredményezte, hogy a HTML alkalmazkodott az XML szerkezetéhez és szabályaihoz. Az egyesülés megvalósításaként született meg a HTML egy új változata, amely követi az XML szigorúbb szabályait – ez az XHTML. Ebben a könyvben szerencsére rögtön az XHTML-lel dolgozunk, hiszen ez valójában nem más, mint a HTML egy „tisztább” változata.

Végezetül, szólnunk kell pár szót a HTML 5-ről, amelyet a legújabb webes szabványként emlegetnek – valamikor talán valóban az lesz, de erre még éveket kell várunk. Mindazonáltal, ha ez bekövetkezik, az XHTML-t sem kell sutba dobunk, hiszen a HTML 5 – amely voltaképpen a HTML 4 egy átdolgozott változata – nem helyettesíti. Röviden, az XHTML és a HTML 5 együtt is létezhet majd a Weben, és azok a böngészők, amelyek ma támogatják az XHTML-t, egy szép napon majd a HTML 5-öt is elfogadják.

Könyvünk célkitűzése, hogy bevezetést adjon a webtervezés világába az XHTML és a CSS megismerésén keresztül. Mindazonáltal, ahol csak lehetséges, jelezzük, ha ezeknek a nyelveknek egy-egy eleme kimaradt a HTML 5 szabványból, így a távolabbi jövőre is könnyebben tervezhetünk. Ha az alapokkal tisztában vagyunk, és megtanultuk, miként használhatjuk a CSS-stílusokat a weboldal jelölőnyelvével (legyen az XHTML vagy HTML 5), pár év múlva könnyű dolgunk lesz, ha esetleg döntenünk kell, hogy álljunk-e az XHTML-ról a HTML 5-re.

Összefoglalás

Ezen az órán megismerkedtünk a weboldalak szerkezetének és működésének alapjaival. Megtudtuk, hogy a weboldalakat alkotó szövegfájlokat kódolt HTML-utasítások „keltik életre”, és arról is szó esett, hogy – főként a tanulás időszakában – érdemes egyszerű szövegszerkesztővel elkészítenünk az oldalaink kódját ahelyett, hogy grafikus szerkesztők szolgáltatásait vennénk igénybe. Megismerkedtünk a legegyszerűbb és legfontosabb HTML-elemekkel – ezeket alkalmazva egy egyszerű dokumentumból pillanatok alatt valódi weboldalt hozhatunk létre. Megtudtuk azt is, hogy egy weboldal elkészítésénél először el kell helyezni néhány kötelező elemet a kód elején és végén – ide tartozik a cím megadása is. Ezután kijelölhetjük a bekezdések és a sorok határait, majd vízszintes vonalakkal és címsorokkal tovább tagolhatjuk a tartalmat. A 3.1. táblázatban áttekintjük az órán megismert elemeket.

Az óra végén megismerkedtünk az XML és az XHTML fogalmával, valamint azzal, hogy miként köthetők a HTML-hez, végül pedig megtudtuk, mi is az a HTML 5, és hogyan kapcsolódik az itt tanultakhoz.

3.1. táblázat A 3. órán bemutatott HTML-elemek

Elem	Leírás
<code><html>...</html></code>	Magában foglalja a teljes HTML-dokumentumot.
<code><head>...</head></code>	Ez az elem tartalmazza a HTML-dokumentum fejlécét. A <code><html></code> címképáron belül kell használni.
<code><title>...</title></code>	Ez az elem tartalmazza a dokumentum címét. A <code><head></code> címképáron belül kell használni.
<code><body>...</body></code>	Ez az elem tartalmazza a HTML-dokumentum törzsét. A <code><html></code> címképáron belül kell használni.
<code><p>...</p></code>	Bekezdés. Az egyes bekezdések között kihagy egy sort.
<code>
</code>	Sortörés.
<code><hr /></code>	Vízszintes vonal.
<code><h1>...</h1></code>	Első szintű címsor.
<code><h2>...</h2></code>	Második szintű címsor.
<code><h3>...</h3></code>	Harmadik szintű címsor.
<code><h4>...</h4></code>	Negyedik szintű címsor (ritkán használt).
<code><h5>...</h5></code>	Ötödik szintű címsor (ritkán használt).
<code><h6>...</h6></code>	Hatodik szintű címsor (ritkán használt).

Kérdesz-felelek

- K: Készítettem egy *weboldalt*, de amikor megnyitom a fájlt a böngészőmben, nem a várt eredményt kapom, hanem szöveget HTML-kódokkal. Olykor ráadásul az oldal tetején mindenféle összevissza karakterek jelennek meg. Mi lehet a baj?
- V: Minden bizonnyal nem egyszerű szöveggént mentettük a fájlt. Próbáljuk meg újra a mentést, ezúttal azonban ügyeljünk arra, hogy biztosan a Plain Text vagy az ASCII Text fájltypust válasszuk. Ha nem találjuk ezt a beállítást, ne essünk kétségbe – egyszerűen írjuk be a HTML-kódot a Jegyzetömb vagy a TextEdit szerkesztőbe, és biztosak lehetünk benne, hogy minden megfelelőképpen fog működni. (Győződjünk meg arról is, hogy a fájl a `.html` vagy a `.htm` kiterjesztést kapta.)
- K: Számos olyan *weboldalt* találhatunk az Interneten, amelyeknek az elején nem szerepel a `<html>` címke, noha az előzőekben azt tanultuk, hogy a HTML-kódnak minden esetben így kell kezdődnie. Hogyan lehetséges ez?
- V: Számos böngésző elnézi nekünk, ha megfelelkezünk a `<html>` címkéről, és így is helyesen jeleníti meg a weboldalakat. Mindazonáltal, mégiscsak jobb, ha feltüntetjük ezt az elemet, hiszen egyes programok csak így fogadják el a fájlt érvényes HTML-oldalként. Már csak azért is így kell tennünk, mert azt szeretnénk, hogy a weboldalaink megfeleljenek az XHTML szabvány követelményeinek.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Melyik az a négy elem, amelynek minden HTML-oldalon meg kell jelennie?
2. Milyen HTML-elemekkel és szöveggel készíthetjük el a következőket?
 - Kis méretű címsor a **We are Proud to Present** szöveggel
 - Egy vízszintes vonal, amely átszeli az oldalt
 - Nagy méretű címsor az **Orbit** szöveggel
 - Közepes méretű címsor a **The Geometric Juggler** szöveggel
 - Egy újabb vízszintes vonal
3. Milyen kódra van szükség egy olyan HTML-oldal létrehozásához, amelynek a címe Foo Bar, az oldal tetején pedig egy címsor áll a Happy Hour at the Foo Bar szöveggel, amelyet a normál betűtípussal szedett Come on down! sor követ.

Válaszok

1. A `<html>`, a `<head>`, a `<title>` és a `<body>` elem (a hozzájuk tartozó `</html>`, `</head>`, `</title>` és `</body>` zárócímkékkel).
2. A feladatot az alábbi kóddal oldhatjuk meg:

```
<h3>We are Proud to Present</h3>
<hr />
<h1>Orbit</h1>
<h2>The Geometric Juggler</h2>
<hr />
```

3. A weboldal kódja a következő:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Foo Bar</title>
</head>
<body>
<h1>Happy Hour at the Foo Bar</h1>
<p>Come on Down!</p>
</body>
</html>
```

Gyakorlatok

- Lehetséges, hogy azért olvassuk ezt a könyvet, hogy a cégünk számára készítsünk webes tartalmakat, de ha másért nem, hát a gyakorlat miatt érdemes lehet egy saját weboldalt is létrehoznunk. Írjunk néhány bemutatkozó bekezdést, és készítsünk belőlük valódi weboldalt az órán tanultak szerint.
- Ahogy haladunk a könyv anyagában, a példák kapcsán saját weboldalakat is készítünk majd, ezért szánjunk most némi időt arra, hogy felépítsük a leendő weboldalaink sablonját – tüntessük fel az XML- és a dokumentumtípus-meghatározást, valamint a HTML-kód alapszerkezetének elemeit. Így bármikor lemásolhatjuk ezeket az adatokat, amikor csak szükségünk van rájuk.



4. ÓRA

A stíluslapok világa

A lecke tartalma:

- Egyszerű stíluslapok létrehozása
- Stílusosztályok használata
- Stílusazonosítók használata
- Belső stíluslapok és kódon belüli stílusok használata

Az előző órán megismerkedtünk a HTML és az XHTML alapjaival, és megtudtuk azt is, hogy miként készíthetünk HTML-sablonvázatot a webes tartalom számára. Ezen az órán a rangsorolt vagy *többszintű stíluslapok* (CSS, cascading style sheet) kerülnek terítékre, amelyek lehetővé teszik a tartalom megjelenítésének finomhangolását. A stíluslapok mögött meghúzódó elgondolás egyszerű: egy stíluslap-dokumentumot készítünk, amellyel megadjuk a webhelyünk betűtípusait, színeit, térközeit és más jellemzőit, amivel egyedi megjelenést biztosítunk a számára. Ha ezzel elkészültünk, ehhez a stíluslaphoz kapcsolhatunk minden olyan weblapot, amelynek hasonló megjelenést szánunk – nem kell tehát külön-külön megadnunk a stílusokat mindegyikük kódjában. Így, ha később módosítani szeretnénk a céges betűtípust vagy színösszeállítást, ezt megtehetjük a stíluslap egy-két

bejegyzésének megváltoztatásával, és nincs szükségünk arra, hogy sorban átírjuk az egyes statikus weblapok kódját. A *stíluslap* tehát formázási utasítások gyűjteménye, amelyek egyszerre több HTML-oldal megjelenését befolyásolják.

A stíluslapok számos formázási beállítást tesznek elérhetővé – megadhatjuk a betűtípusok különféle beállításait, a betű- és sorközöket, a margókat és szegélyeket, és még sok minden mást. A méreteket különféle egységekben határozhatjuk meg, így dönthetünk a hüvelyk, a milliméter, a pont vagy éppen a pica mellett. A stíluslapok segítségével pontosan elhelyezhetjük a kívánt szöveget és grafikákat a weboldalakon, akár koordinátákkal, akár az oldal más elemeihez képest.

Röviden szólva, a stíluslapok kifinomultabb megjelenést tesznek lehetővé a Weben, ráadásul – elnézést a kihagyhatatlan szójátékért – mindezt stílusosan teszik.



Ha van legalább három weboldalunk, amelyek hasonló formázással és betűtípussal rendelkeznek (vagy kellene, hogy rendelkezzenek), érdemes az óra során készíteni számukra egy közös stíluslapot. De ha erre jelenleg nincs is igényünk, mindenképpen érdemes stílusokat alkalmaznunk a weboldalaink egyes HTML-elemeire.

A CSS működése

A stíluslapokat leíró nyelv a CSS, amelynek az elemeivel betűtípusokat, színeket, illetve elhelyezést határozhatunk meg, így végeredményben megadhatjuk, hogy miként jelenjen meg egy oldal tartalma a böngészőablakban. A CSS-stílusokat tárolhatjuk közvetlenül az adott HTML-fájlban, de külön stíluslapfájlban is elhelyezhetjük azokat. Bárhogy is járunk el, a stíluslapok stíluszabályokat tartalmaznak, amelyek adott típusú elemek megjelenését befolyásolják. Ha külön fájlban tároljuk őket, a `.css` kiterjesztéssel jelöljük, hogy stíluslapfájlról van szó.

A *stíluszabályok* formázási utasítások, amelyeket a weboldalak elemeire, például a szövegbekezdésekre vagy hivatkozásokra alkalmazhatunk. A stíluszabályok *stílustulajdonságoknak* adott értékekből épülnek fel. A *belső stíluslapok* közvetlenül a weboldalak kódjában jelennek meg, míg a *külső stíluslapok* önálló fájlban kapnak helyet, amelyet egy erre a célra alkalmas címkével kapcsolunk a weboldalhoz – erről a címkéről hamarosan bővebben is szólunk.

A CSS nevében szereplő „cascading” (rangsorolt vagy többszintű) kifejezés arra utal, ahogyan a rendszer a stíluszabályokat a HTML-oldal elemeire alkalmazza. Arról van szó ugyanis, hogy a CSS stíluslap elemei egy hierarchiát alkotnak, amelyben a szűkebb stílusok felülírják a tágabb érvényűeket. A CSS-értelmezőre hárul a feladat, hogy a gyakorlatban is alkalmazza ezt a rangsort. Ha mindez kissé zavarosnak tűnik, gondoljunk a rangsorolás folyamatára úgy, mint a genetikai öröklésre, ahol a tulajdonságok nagyjá-

ban-egészében átöröklődnek a szülőkről a gyerekekre, de az utódok is rendelkeznek sajátos jellemzőkkel. Az általánosabb stílusszabályok tehát a teljes stíluslapra érvényesülnek, a szűkebb érvényű szabályok azonban felülírhatják azokat.

Egy rövid példa nyomán minden világossá válik majd. Pillantsunk az alábbi kódrészletre, és próbáljuk kiokoskodni, hogy mi történik a szöveg színével:

```
<div style="color:green">
  This text is green.
  <p style="color:blue">This text is blue.</p>
  <p>This text is still green.</p>
</div>
```



Bizonyára észrevettük, hogy az *elem* szó a szokásosnál többször szerepelt az óra eddigi anyagában (és gyakran megjelenik a könyv többi részében is), érdemes tehát tisztázni, hogy pontosan mit is takar ez a fogalom. Nos, *elemnek* nevezünk minden tartalom- vagy információegységet a weboldalakon – ez lehet például kép, bekezdés vagy hivatkozás. Az elemek kódolására a *címkék* szolgálnak, vagyis az elemre úgy tekinthetünk, mint egy leíró adatokkal (jellemzők, szöveg, képek és más egyebek) feltöltött címkére.

A fenti példában a zöld színt a `color` stílustulajdonságon keresztül alkalmaztuk a `<div>` elemre, így hát a `<div>` elemen belüli szöveg zöld lesz. Mindkét `<p>` elem a `<div>` gyermeke, így a zöld szövegstílus átöröklődik rájuk. Mindazonáltal az első `<p>` elem felülírja ezt a stílust, és kékre változtatja a szöveg színét. Végeredményképpen az első sor (amely a bekezdéscímkéken kívül esik) szövege zöld, az első valódi bekezdésben kék színű betűk szerepelnek, a második bekezdés szövege pedig megőrzi az örökölt zöld színt.

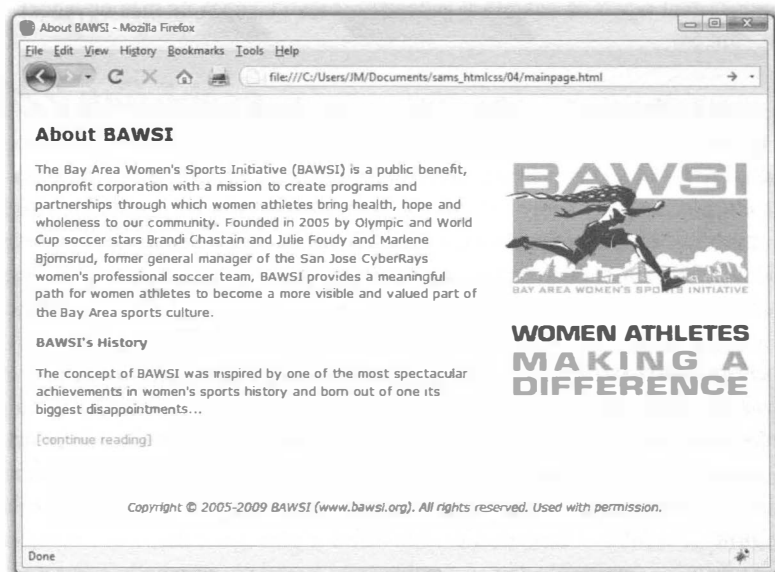
Más webes technológiákhoz hasonlóan a CSS is fejlődött az évek során. Eredeti változata, a *Cascading Style Sheets Level 1 (CSS1)* 1996-ban jelent meg, utóda, a máig használatban levő CSS 2 pedig 1998-ban. Napjaink böngészői kivétel nélkül képesek értelmezni a CSS 2 kódokat, így minden különösebb elővigyázatosság nélkül használhatjuk a CSS 2 stíluslapokat a munkánk során. Ebben a könyvben, ha a CSS-ről szólunk, a CSS 2-re gondolunk.

Nagyszerű referenciamunkát találhatunk a CSS használatáról a <http://www.w3.org/Style/CSS/> címen. Az óra további részében bemutatjuk, miként válhat a CSS megbecsült segítőtársunkká.

Egyszerű stíluslap létrehozása

A stíluslapok létrehozása – dacára lenyűgöző képességeiknek – igen egyszerű is lehet. Nézzük meg a 4.1. és a 4.2. ábrán látható weboldalakat, amelyek nyilvánvalóan számos közös képi jellemzővel rendelkeznek, amelyeket érdemes egy közös stíluslapban rögzítenünk:

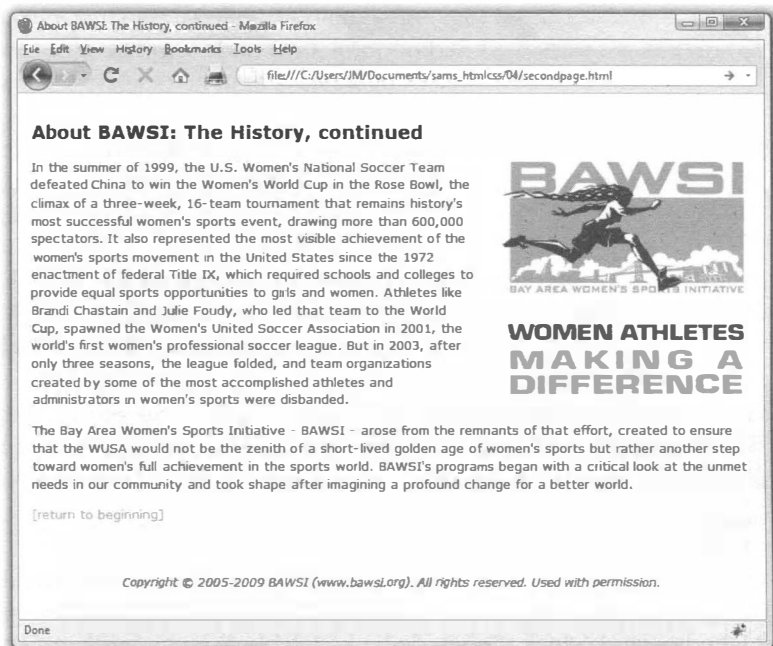
- Mindkét oldal címsorában nagy, félkövér Verdana betűket találunk, a törzsszövegben pedig rendes méretű és betűvastagságú szöveget ugyanezzel a betűtípussal.
- Mindkét oldalon találunk egy `logo.gif` nevezetű képet, amely a képernyő jobb oldalára kerül, a szöveg pedig körülfolymja.
- Minden szöveg fekete, kivéve az alcímeket, amelyek lila színnel jelennek meg.
- Mindkét oldal rendelkezik bal és felső margóval.
- A szövegsorok között azonos térközöket találunk.
- A lábjegyzetek középre igazítva, kis betűmérettel jelennek meg.



4.1. ábra

Ez a weboldal egy stíluslap segítségével határozza meg a szövegek és a képek megjelenését és térközzeit

A 4.1. példában látható stíluslap rögzíti a fenti tulajdonságokat.



4.2. ábra

Ez az oldal a 4.1. ábra stíluslapját használja, így képes megőrizni az egységes oldalképet

4.1. példa Külső stíluslap

```
body {
    font-size: 10pt;
    font-family: Verdana, Geneva, Arial, Helvetica,sans-serif;
    color: black;
    line-height: 14pt;
    padding-left: 5pt;
    padding-right: 5pt;
    padding-top: 5pt;
}

h1 {
    font: 14pt Verdana, Geneva, Arial, Helvetica,sans-serif;
    font-weight: bold;
    line-height: 20pt;
}

p.subheader {
    font-weight: bold;
    color: #593d87;
}
```

```
img {  
    padding: 3pt;  
    float: right;  
}  
  
a {  
    text-decoration: none;  
}  
  
a:link, a:visited {  
    color: #8094d6;  
}  
  
a:hover, a:active {  
    color: #FF9933;  
}  
  
div.footer {  
    font-size: 9pt;  
    font-style: italic;  
    line-height: 12pt;  
    text-align: center;  
    padding-top: 30pt;  
}
```

Első pillantásra ez igen hosszú kódnak tűnhet, de ha közelebbről megnézzük, azt látjuk, hogy egy-egy sor nem tartalmaz különösebben sok adatot. Megszokottnak nevezhető, hogy az egyes szabályokat külön sorokban helyezzük el a jobb átláthatóság érdekében. Ha már a kód értelmezhetőségéről beszélünk, nem mehetünk el szó nélkül amellett, hogy a stíluslapok kódja teljesen eltér a HTML-oldalakon láthatótól – ez nem véletlen, hiszen a CSS teljesen önálló nyelv.

Találhatunk persze itt ismerős HTML-címkéket is – ahogy magunk is sejthettük, a stíluslapon látható `body`, `h1`, `p`, `img`, `a` és `div` kódok a megfelelő HTML-elemekre utalnak, amelyekre az adott stílust alkalmazzuk. Az elemek neve után álló kapcsos zárójelek között szereplő kód pedig az adott elem megjelenésének meghatározására hivatott.

Esetünkben a `body` szövege 10 pontos Verdana betűkkel jelenik meg (amennyiben ez lehetséges), fekete színnel, és a sorok között 14 pont távolsággal. Ha a felhasználó nem rendelkezik a Verdana betűtípussal, a stíluslapon meghatároztuk, hogy mely betűtípusokkal és milyen sorrendben próbálja helyettesíteni a böngésző. Ha ezek egyike sincs jelen a rendszerben, a böngésző végül az alapértelmezett talp nélküli betűtípus-hoz nyúl. Mindemellett az oldal 5 pontos bal, jobb és felső margót kap.

A `<h1>` elemen belüli szöveg a stíluslap utasításai szerint 14 pontos félkövér Verdana betűkkel jelenik meg. Ha tovább haladunk, láthatjuk, hogy azok a bekezdések, amelyek mindössze a `<p>` elemet használják, a `body` elem összes stílusát öröklik. Ha azonban a subheader nevű osztályt is használatba veszik, a szöveg félkövéren és az `#593d87` színnel jelenik meg.

A 4.1. példa értékei után álló *pt* *pontot* jelent, ahol egy hüvelyket 72 pont tesz ki. Ha úgy tartja kedvünk, más mértékegységet is választhatunk, így mérhetünk hüvelykben (in), centiméterben (cm), képpontban (px) valamint emben (em) – ez utóbbi esetben az egység az *m* betű szélessége.

Talán észrevettük, hogy az egyes stíluszabályok után egy-egy pontosvessző (;) áll. Ezek választják el egymástól a szabályokat, így alapvető fontosságú, hogy az egyes szabályok után feltüntessük ezt a jelet.



A stíluslapokon tetszőlegesen nagy betűméretet megadhatunk, de érdemes tudnunk, hogy bizonyos megjelenítők és nyomtatók nem kezelik megfelelően a 200 pontnál nagyobb betűket.

Ahhoz, hogy ezt a stíluslapot a HTML-dokumentumainkban felhasználjuk, helyezzünk el a <head> részükben egy-egy <link /> elemet. A 4.2. példában a 4.1. ábrán látható weboldal HTML-kódját láthatjuk. Itt a következő <link /> elemet találjuk:

```
<link rel="stylesheet" type="text/css" href="styles.css" />
```

Mindez feltételezi, hogy a stíluslapot egy *styles.css* nevű fájlban tároljuk, még hozzá ugyanabban a könyvtárban, mint a HTML-dokumentumot. Ha a böngésző támogatja a stíluslapokat – ezt napjainkban mindegyikük megteszi –, az ott felsorolt tulajdonságok automatikusan fejtik ki a hatásukat, anélkül, hogy bármilyen HTML-formázási kódot kellene alkalmaznunk. Mindez jól igazodik az XHTML szabvány valódi célkitűzéséhez, amely szerint el kell választanunk a weboldalak tartalmát a megjelenítésükhöz szükséges adatoktól.

4.2. példa A 4.1. ábrán látható weboldal HTML-kódja

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>About BAWSI</title>
    <link rel="stylesheet" type="text/css" href="styles.css" />
  </head>
  <body>
    <h1>About BAWSI</h1>
    <p>The Bay Area Women's
    Sports Initiative (BAWSI) is a public benefit, nonprofit
    corporation with a mission to create programs and partnerships
    through which women athletes bring health, hope and wholeness to
    our community. Founded in 2005 by Olympic and World Cup soccer
    stars Brandi Chastain and Julie Foudy and Marlene Bjornsrud,
    former general manager of the San Jose CyberRays women's
    professional soccer team, BAWSI provides a meaningful path for
```

```
women athletes to become a more visible and valued part of the
Bay Area sports culture.</p>
<p class="subheader">BAWSI's History</p>
<p>The concept of BAWSI was inspired by one of the most
spectacular achievements in women's sports history and born out
of one its biggest disappointments... </p>
<p><a href="secondpage.html">[continue reading]</a></p>
<div class="footer">Copyright &copy; 2005-2009 BAWSI
(www.bawsi.org). All rights reserved. Used with permission.</div>
</body>
</html>
```



A legtöbb böngészőben megtekinthetjük a stíluslap tartalmát, ha megnyitjuk a megfelelő .css fájlt a Jegyzettömbben, vagy egy másik, dokumentumok megjelenítésére alkalmas szövegszerkesztő segédprogramban. (A .css fájl nevének megállapításához tanulmányozzuk a megfelelő HTML-fájl kódját.) A saját stíluslapjaink szerkesztéséhez használjunk valamilyen szövegszerkesztőt.



Jóllehet manapság a CSS-t széles körben támogatják a böngészők, a helyzet korántsem állt mindig így. Ráadásul ez a támogatás még ma sem minden esetben tökéletes. Ha meg szeretnénk tudni, miben térnek el egymástól a böngészők a CSS támogatása terén, látogassunk el

a [http://www.quirksmode.org/css/contents.html](http://www quirksmode.org/css/contents.html) címre.

A 4.2. példa tartalma azért érdekes, mert egyáltalán nem tartalmaz formázást, vagyis a HTML-kódban semmi nem utal arra, hogyan kellene megjeleníteni a szöveget vagy a képeket – nincsenek színek, betűtípusok vagy bármi más hasonló. Az oldal formázása azonban mégis tetszetős, köszönhetően a styles.css fájlban található külső stíluslapnak. Ennek a megközelítésnek az az igazi előnye, hogy így olyan, több oldalból álló webhelyet készíthetünk, amelynek az oldalai egységes képet mutatnak. Ráadásul egyetlen dokumentumban (ez a stíluslap) foglaljuk össze az oldalak képi megjelenésének kódját, vagyis egy módosítás az összes kapcsolódó oldalra hatással van.



Önálló feladat

Készítsünk saját stíluslapot!

Hozzunk létre egy mystyles.css nevű fájlt, és helyezzünk el benne néhány stílusszabályt a <body>, <p>, <h1> és <h2> elemek számára. Ha megvan a stíluslap, hozzunk létre egy új HTML-fájlt, amely tartalmazza ezeket az egyszerű elemeket. Próbáljunk ki különféle stílusszabályokat, és figyeljük meg, milyen könnyen módosíthatjuk bekezdések egész szövegtömbjeinek a megjelenését a stíluslap egyetlen apró módosításával.

A CSS-stílusok használatának alapjai

Megismerkedtünk a CSS-stíluslapokkal, és megtudtuk azt is, hogyan épülnek stíluszabályokra, amelyek a weboldalak tartalmának megjelenítését szabályozzák. A következőkben rövid áttekintést adunk a legfontosabb stílustulajdonságokról, lehetővé téve, hogy felépítsük a saját stíluslapjainkat.

A CSS számos stílustulajdonságot tartalmaz, amelyekkel szabályozhatjuk a betűtípusokat, az színeket, az igazítást, a margókat, és még sok egyebet. Ezek a tulajdonságok alapvetően két csoportra oszthatók:

- **Elrendezési tulajdonságok** – ide azok a tulajdonságok tartoznak, amelyek az oldal elemeinek elhelyezkedését, így a margót, a kitöltést, az igazítást és hasonlókat szabályozzák.
- **Formázási tulajdonságok** – ide tartoznak azok a tulajdonságok, amelyek az oldal elemeinek megjelenését befolyásolják, vagyis a betűtípust, a betűméretet, a színt, és így tovább.

Elrendezési tulajdonságok

A CSS elrendezési tulajdonságai segítségével meghatározhatjuk a tartalom elhelyezését egy weboldalon. Az egyik legfontosabb közülük a `display`, amely megadja, hogyan jelenjen meg egy adott elem a többihez képest. A tulajdonságnak négy értéke lehetséges:

- `block` – Az elem új sorban jelenik meg, mint az új bekezdések esetében.
- `list-item` – Az elem új sorban jelenik meg, és egy listacímke áll előtte.
- `inline` – Az elem az adott bekezdésen belül jelenik meg.
- `none` – Az elem nem jelenik meg, hanem rejtve marad.

A `display` tulajdonság a *relatív elhelyezésre* épít, amelynek során az elemeket a társaikhoz képest helyezzük el az oldalon. A CSS lehetővé teszi az *abszolút elhelyezést* is, amikor az adott elemet a többiektől függetlenül, pontosan meghatározott koordinátákkal jelenítjük meg. Minderről bővebben a könyv III. részében olvashatunk.



A `display` tulajdonságot könnyebben megérthetjük, ha a weboldalak elemeit úgy fogjuk fel, mint téglalap alakú területeket – ezek megjelenítését szabályozza a `display` tulajdonság. A `block` érték hatására az adott elem önállóan, külön sorba kerül, míg az `inline` érték alkalmazásakor az őt megelőző tartalom mellé helyezzük. A `display` egyike annak a kevés stílustulajdonságnak, amelyet a legtöbb stíluszabályban felhasználhatunk. Az alábbi példában bemutatjuk, hogyan állíthatjuk be a `display` értékét:

```
display: block;
```

Az adott elemhez tartozó téglalap alakú terület méretét a `width` (szélesség) és a `height` (magasság) tulajdonságokkal szabályozhatjuk. Számos más, méreteket meghatározó CSS-tulajdonsághoz hasonlóan itt is különböző mértékegységek közül választhatunk:

- `in` – hüvelyk
- `cm` – centiméter
- `mm` – milliméter
- `px` – képpont
- `pt` – pont

Egy stíluslapon belül különböző mértékegységeket is használhatunk, de érdemes a hasonló tulajdonságoknál hasonló mértékegységek mellett döntenünk. Így a betűtípusok tulajdonságainál például a pont, a méreteknél pedig a képpont nagyszerű választás lehet. Az alábbi példában az elem szélességét képpontban határozzuk meg:

```
width: 200px;
```

Formázási tulajdonságok

A CSS formázási tulajdonságai a tartalom megjelenését szabályozzák, nem pedig annak fizikai elhelyezését. Az egyik legismertebb közülük a `border` (szegély), amelynek a segítségével részben vagy teljesen láthatóvá tehetjük egy adott elem határait. Az alábbi tulajdonságok az elem szegélyének különféle jellemzőit írják le:

- `border-width` – A szegély vastagsága.
- `border-color` – A szegély színe.
- `border-style` – A szegély stílusa.
- `border-left` – A bal oldali szegély.
- `border-right` – A jobb oldali szegély.
- `border-top` – A felső szegély.
- `border-bottom` – Az alsó szegély.
- `border` – A szegély minden oldala.

A `border-width` tulajdonsággal a szegély vastagságát adhatjuk meg. Gyakran képpontban rögzítjük az értékét – így teszünk a következő példában is:

```
border-width: 5px;
```

A `border-color` és a `border-style` nem különösebben meglepő módon a szegély színét és stílusát adja meg, valahogy így:

```
border-color: blue;  
border-style: dotted;
```

A `border-style` tulajdonság az alábbi értékeket veheti fel:

- `solid` – Egyvonalas szegély.
- `double` – Kettőzött vonalas szegély.
- `dashed` – Szaggatott vonalas szegély.
- `dotted` – Pontozott szegély.
- `groove` – Bemetszett szegély.
- `ridge` – Peremes szegély.
- `inset` – Homorú megjelenést adó szegély.
- `outset` – Domború megjelenést adó szegély.
- `none` – Nincs szegély.

A `border-style` alapértelmezett értéke a `none` – ez az oka annak, hogy az elemek nem rendelkeznek szegéllyel, hacsak magunk nem teszünk valamit az ügy érdekében. A legelterjedtebb szegélystílusok a `solid` és a `double`.

A `border-left`, `border-right`, `border-top` és `border-bottom` tulajdonságok lehetővé teszik, hogy egyenként állítsuk be a különböző oldali szegélyeket. Ha azt szeretnénk, hogy a szegély egyforma legyen mind a négy oldalon, alkalmazhatjuk a `border` tulajdonságot, amelyben a `border-width`, `border-style` és `border-color` stílusok értékeit ebben a sorrendben, szóközzel elválasztva adhatjuk meg. Az alábbiakban a `border` tulajdonsággal egy két vörös vonalból álló, összesen 10 képpont vastagságú szegélyt hozunk létre:

```
border:10px double red;
```



A `border-style` egyetlen esetben nem a `none` értéket kapja alapértelmezés szerint – ha egy képet helyezünk el egy `<a>` elemen, vagyis egy hivatkozáson belül. Ilyenkor automatikusan folytonos szegélyt kapunk. Ez az oka annak, hogy olyan gyakran találkozunk a hivatkozásokban szereplő képeknél a `border-style:none` stílussal, amely eltünteteti ezt az automatikus szegélyt.

Megtudtuk tehát, hogy a szegély színét a `border-color` tulajdonsággal adhatjuk meg – a szegélyen belül álló tartalom színét azonban a `color` és a `background-color` tulajdonságok szabályozzák. Az előbbi az elemben szereplő szöveg (az előtér) színéért felel, míg az utóbbi a szöveg mögötti háttér színét adja meg. Az alábbiakban mindkét tulajdonságban névvel ellátott színt alkalmazunk:

```
color:black;
background-color:orange;
```

Ezekhez a tulajdonságokhoz egyéni színeket is rendelhetünk, csak adjuk meg a hexadecimális (lásd a 9. órát) vagy a decimális RGB (vörös-zöld-kék) kódjukat, mint a HTML-oldalak esetében:

```
background-color:#999999;
color:rgb(0,0,255);
```

A CSS segítségével könnyen szabályozhatjuk a webes tartalom igazítását és behúzásait is a `text-align` és a `text-indent` tulajdonságok révén, valahogy így:

```
text-align:center;
text-indent:12px;
```

Ha elkészültünk egy adott elem igazításával és behúzásával, hozzáláthatunk a betűtípusa beállításához. Ebben az alábbi tulajdonságok lehetnek a segítségünkre:

- `font-family` – A betűtípus családja.
- `font-size` – A betűméret.
- `font-style` – A betűstílus (*normal* vagy *italic*).
- `font-weight` – A betű súlya vagy vastagsága (*light*, *medium*, *bold* és így tovább).

A `font-family` tulajdonságban betűtípusok rangsorolt listáját adhatjuk meg – azért van szükség listára, mert így további lehetőségeket biztosítunk a böngészőnek arra az esetre, ha valamelyik betűtípus nem áll rendelkezésre az adott rendszeren.

A `font-size` tulajdonsággal a betűméretet határozhatjuk meg valamilyen mértékegységben, többnyire pontban mérve. Végül, a `font-style` tulajdonsággal a betű stílusát, a `font-weight` segítségével pedig a súlyát adhatjuk meg. A beállításaink összessége például így alakulhat:

```
font-family: Arial, sans-serif;
font-size: 36pt;
font-style: italic;
font-weight: medium;
```

Sokat tanultunk a stílusokról, így érdemes visszalapoznunk a 4.1. példához, hogy megnézzük, mennyit értünk belőle most. Az alábbiakban összefoglaljuk a listában szereplő stílustulajdonságok jellemzőit:

- `font` – Lehetővé teszi, hogy egyszerre határozzuk meg a betűk több tulajdonságát. Megadhatjuk az ajánlott betűtípusok listáját, így ha az első nem érhető el az adott rendszeren, a böngésző megpróbálkozhat a következővel, és így tovább. Feltüntethetjük a *bold* (félkövér), illetve az *italic* (dőlt) beállításokat, valamint a betűméretet. Mindazonáltal, ha úgy tartja kedvünk, ezeket a tulajdonságokat külön-külön is megadhatjuk a `font-family`, `font-size`, `font-weight` és `font-style` stílusokkal.
- `line-height` – A tördelők világában kitöltésnek (*leading*) is nevezett érték a sormagasságot adja meg, többnyire pontban.
- `color` – Ennek a tulajdonságnak a segítségével a szöveg színét határozhatjuk meg, akár szabványos nevekkal, akár hexadecimális színkódokkal (bővebben lásd a 9. fejezetben).

- `text-decoration` – Hasznos tulajdonság, ha a hivatkozások aláhúzását szeretnénk eltávolítani – ilyenkor adjuk neki a `none` értéket. Emellett választhatjuk az `underline`, az `italic` és a `line-through` beállítást is. A stílusok alkalmazását a hivatkozásokra bővebben a 8. órán tárgyaljuk.
- `text-align` – Ennek alkalmazásával balra, jobbra vagy középre (`left`, `right` vagy `center`) igazíthatjuk a szöveget, illetve sorkizárttá (`justify`) tehetjük.
- `padding` – Ezzel a tulajdonsággal kitöltést rendelhetünk az elem bal, jobb, felső és alsó oldalához egy ismert mértékegységben, illetve az oldal méretének százalékában. A jobb és bal oldal kitöltésének önálló meghatározására a `padding-right` és a `padding-left`, az alsó és felső oldalkitöltésére pedig a `padding-bottom` és a `padding-top` szolgál. Ezekről a tulajdonságokról bővebben a 14. és 15. órákon lesz szó.

Stílusosztályok használata

Ez egy önálló tanuláshoz készült könyv, így nem kell osztályba járnunk ahhoz, hogy a stílusokról tanuljunk, de a stílusosztályokkal mindenképpen meg kell ismerkednünk. Ha azt szeretnénk, hogy egy weboldal valamelyik szövegrészlete másképp jelenjen meg, mint az oldal többi része, bizonyos értelemben magunk is létrehozhatunk egyéni HTML-elemeket. Ezeket a különleges formázási utasításokat *stílusosztályokban* tárolhatjuk – a *stílusosztályok* tehát olyan formázási beállítások gyűjteményei, amelyek egy egységként alkalmazhatók egy weboldal bármely elemére.

Mielőtt azonban megismerkednénk az első stílusosztályunkkal, álljunk meg egy pillanatra, és tisztázzunk néhány fogalmat a CSS-stílusok témakörében. Először is, rögzítsük, hogy *stílustulajdonság* alatt olyan stílust értünk, amelyhez értéket rendelhetünk – ilyen például a `color` vagy a `font-size`. A stílustulajdonságot és a hozzá tartozó értéket egy *választó* vagy kijelölő (*selector*) segítségével alkalmazzuk a weboldal adott elemére. A választók segítségével azonosíthatjuk tehát az oldalak egyes elemeit. Lássunk most egy választót, egy tulajdonságot és egy értéket együtt egy egyszerű stílus-szabályban:

```
h1 { font: 36pt Courier; }
```

Itt a választó a `h1`, a stílustulajdonság a `font`, az érték pedig a `36 pt Courier`. A választó jelenléte fontos, hiszen így érjük el, hogy a stílus az oldal minden `h1` elemére hasson. No de mit tegyünk, ha különböző `h1` elemeket szeretnénk használni? Nos, ilyenkor lépnek színre a stílusosztályok.

Tegyük fel, hogy kétféle `<h1>` címsorra van szükségünk. Ilyenkor mindkettőjük számára létrehozhatunk egy osztályt, ha az alábbi kódot helyezzük el a stíluslapon:

```
h1.silly { font: 36pt Comic Sans; }
h1.serious { font: 36pt Arial; }
```

Vegyük észre, hogy a választókban itt a `h1` után pont áll, majd pedig az osztály neve. A két osztály között a HTML-kódban a `class` jellemzővel választhatunk, például így:

```
<h1 class="silly">Marvin's Munchies Inc. </h1>
<p>Text about Marvin's Munchies goes here. </p>
```

De használhatjuk a másik osztályt is:

```
<h1 class="serious">MMI Investor Information</h1>
<p>Text for business investors goes here.</p>
```

Ha egy osztályra szeretnénk hivatkozni a HTML-kódban, nem kell mást tennünk, mint megadni a nevét a megfelelő elem `class` jellemzőjében. Előző példánkban a Marvin's Munchies Inc. 36 pontos Comic Sans betűkkel jelenik meg, amennyiben a stíluslapra hivatkozó `<link />` elemet elhelyeztük az oldal elején, továbbá a Comic Sans betűtípus elérhető a felhasználó rendszerén. Az MMI Investor Information ugyanakkor 36 pontos Arial betűkkel jelenik meg. Az osztályok használatát a 4.2. példa is szemléltette: keressük a `<p>` elem subheader, illetve a `<div>` elem footer osztályát.

Mit tehetünk akkor, ha olyan stílusosztályt szeretnénk létrehozni, amelyet tetszőleges elemre alkalmazhatunk, nemcsak egy címsorra vagy más elemre. Nos, hozzárendelhetünk egy stílusosztályt a `<div>` elemhez – így tettünk a 4.2. példában –, ebbe azután belesomagolhatunk bármilyen szöveget, mintha csak egy bekezdést írnánk – a `<div>` igen hasznos tárolóelem.

Lényegében saját HTML-elemeket hozhatunk létre, ha a `div` választó után egy pontot (.) írunk, majd feltüntetjük mellette a kívánt stílusosztály nevét és a hozzá tartozó stílusmeghatározásokat. Ezzel az elemmel egyszerre tetszőleges számú betűtípus-, térköz- és margóbeállítást meghatározhatunk. Ezt követően, ahol csak használni szeretnénk ezt az egyéni elemet az oldalon, helyezzünk el egy `<div>` elemet, és a `class` jellemzőjében tüntessük fel a stílusosztály nevét.

Így például a 4.1. példa stíluslapján az alábbi stílusosztályt határoztuk meg:

```
div.footer {
  font-size: 9pt;
  font-style: italic;
  line-height: 12pt;
  text-align: center;
  padding-top: 30pt;
}
```

Ezt azután a következő elemmel alkalmaztuk a 4.2. példában:

```
<div class="footer">
```



Talán észrevettük, hogy megváltozott a kód alakja, amint több tulajdonságot soroltunk fel a stílus szabályokban. Ha egy stílus szabály egyetlen stílust tartalmaz, a tulajdonságnak értéket adó utasítást gyakran írják egy sorba a szabállyal, valahogy így:

```
div.footer { font-size: 9pt; }
```

Ha azonban több tulajdonság is szerepel a szabályban, nagyban könnyíti az olvashatóságot, ha soronként csak egyet tüntetünk fel, mint az alábbi kódban:

```
div.footer {
    font-size: 9pt;
    font-style: italic;
    line-height: 12pt;
    text-align: center;
    padding-top: 30pt;
}
```

Bármilyen áll a fenti elem és a záró `</div>` között a 4.2. példában, 9 pontos, dőlt, középre igazított betűkkel jelenik meg, 12 pontos sormagassággal és 30 pontos felső kitöltéssel.

A stílusosztályok igazi értékét az adja, hogy elválasztják a stílusok kódját az oldalak szerkezetétől, így egy oldal készítésénél a tartalomra összpontosíthatunk, és eközben nem kell arra figyelni, hogy miként jelenik majd meg a képernyőn. Ha pedig elkészültünk a tartalommal, nyugodtan összpontosíthatunk a megjelenítésre a stíluslap finombeállításainál. Meglepve tapasztalhatjuk majd, hogy a stíluslap apró módosításai milyen hatalmas változásokat okozhatnak a weboldal megjelenésében. Mindez könnyebbé teszi a webhelyünk fenntartását és módosítását.

Önálló feladat



Osztályok hozzáadása egy stíluslaphoz

Az óra korábbi részében elkészített stíluslapon helyezünk el néhány stílusosztályt. Hogy lássuk is munkánk gyümölcsét, alkalmazzuk ezeket az egyszerű HTML-oldalunkra. A gyakorlat kedvéért módosítsuk a `<h1>` és a `<p>` elemeket.

Stílusazonosítók használata

Az egyéni stílusosztályainkat annyiszor használjuk, ahányszor csak akarjuk – nem egyediek tehát. Előfordulhat azonban, hogy ez nem elég, és az elrendezés vagy a formázás érdekében egyes elemek megjelenését pontosan meg szeretnénk határozni. Ilyenkor osztályok helyett érdemes azonosítókat használnunk.

A *stílusazonosítók* olyan formázási utasítások gyűjteményei, amelyek egy weboldal egyetlen, pontosan meghatározott elemére alkalmazhatók. Egy azonosítót több oldalon is használhatunk, de egy adott weboldalon belül csak egyszer hivatkozhatunk rá.

Tegyük fel, hogy minden oldalunk törzsében található egy cím. Minden oldal egyetlen címmel rendelkezik tehát, de a cím minden oldalon ott van. A következő példában bemutatunk egy azonosítóval ellátott választót, amelyhez egyetlen tulajdonságot határozunk meg a hozzá tartozó értékkel:

```
p#title {font: 24pt Verdana, Geneva, Arial, sans-serif}
```

Figyeljük meg, hogy a választóban az azonosító neve egy kettőskereszt (#) után áll. Ha a HTML-kódban egy stílusazonosítóra szeretnénk hivatkozni, egyszerűen adjuk meg az azonosító nevét a kívánt elem id jellemzőjében:

```
<p id="title">Some Title Goes Here</p>
```

A nyitó és a záró `<p>` címke között álló teljes szöveg 24 pontos Verdana betűkkel jelenik meg – de bármely adott oldalon csak egyszer. Stílusazonosítókkal gyakran találkozunk majd az elrendezési elemeknél, így a fejléc, a lábléc, a törzs és hasonlók meghatározásánál. Mivel ezek az elemek csak egyszer fordulnak elő egy weboldalon, az azonosítók jobban alkalmazhatók az esetükben, mint az osztályok.

Belső stíluslapok és kódon belüli stílusok

Bizonyos esetekben olyan stílusokra is szükségünk lehet, amelyeket csak az adott weboldalon kívánunk alkalmazni. Semmi gond, egy stíluslapot elhelyezhetünk közvetlenül egy weboldalon, a `<style>` és a `</style>` címkék között is – ezt azonban kizárólag az oldal `<head>` részében tehetjük meg. A `<link />` elemre ezúttal nincs szükség, és erre a stíluslapra nem hivatkozhatunk más oldalakról (hacsak át nem másoljuk a kódját a másik dokumentum elejére). Ilyen esetekben belső stíluslapról beszélünk, amint az óra egy korábbi részében már említettük.

A 4.3. példában bemutatjuk, hogyan határozhatunk meg egy belső stíluslapot.

4.3. példa Weboldal belső stíluslappal

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Some Page</title>
```



```

<style type="text/css">
  div.footer {
    font-size: 9pt;
    line-height: 12pt;
    text-align: center;
  }
</style>
</head>
<body>
...
<div class="footer">
Copyright 2009 Acme Products, Inc.
</div>
</body>
</html>

```

A példában a `div.footer` stílusosztályt határozzuk meg az oldal fejlécében megjelenő belső stíluslapon. A stílusosztály ennek megfelelően rendelkezésünkre áll az oldal törzsében – fel is használjuk a szerzői jogi szöveg formázására.



A `` és a `` üres elemek, abban az értelemben, hogy önmagukban semmit nem tesznek, mindössze azonosítanak egy tartalomrészletet, amelyre érvényesíthetjük a `style` jellemzőben megadott beállításainkat. A `<div>` és a `` elemek között mindössze annyi a különbség, hogy az előbbi tömbelem, így a használata sortöréssel jár, míg a `` esetében nincs sortörés. Ha tehát egy folyó szöveg részletét szeretnénk egyedi módon formázni anélkül, hogy sortörést alkalmaznánk, a `` elemet kell választanunk.

Ellenőrizzük a stíluslapjaink érvényességét!

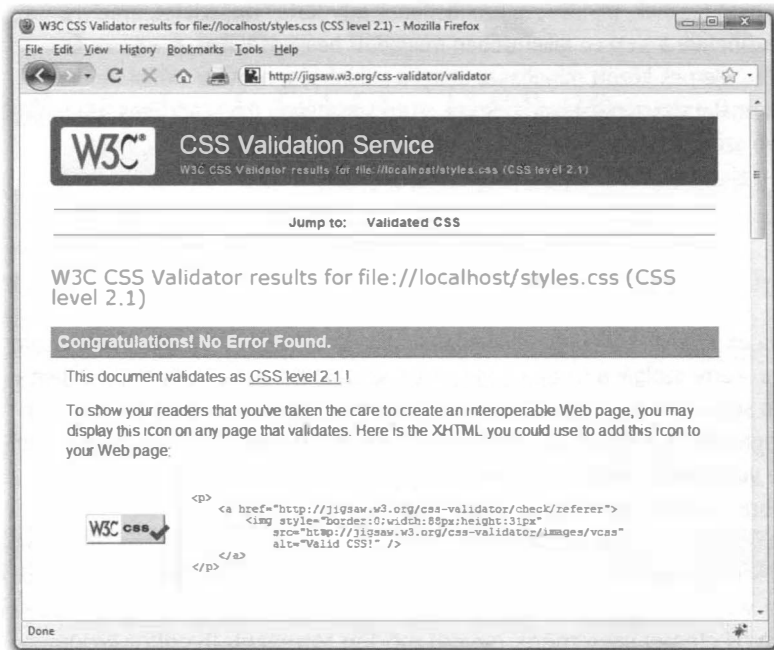
A HTML- és XHTML-kódokhoz hasonlóan a stíluslapok érvényességének vizsgálata is fontos – erre szolgál a <http://jigsaw.w3.org/cssvalidator/> címen található segédeszköz. A 3. órán látott érvényességellenőrző eszközhöz hasonlóan itt is megadhatjuk a kívánt webcímet, feltölthetjük a kérdéses fájlt, illetve bemásolhatjuk a vizsgálandó tartalmat a megfelelő szövegmezőbe. Végso célunk, hogy a 4.3. ábrán látható eredményt kapjuk – a stíluslapunk érvényes.

A belső stíluslapok nagyszerű szolgálatot tesznek, ha olyan stíluszabályt szeretnénk írni, amelyet egy adott weboldalon többször is felhasználunk. Adódnak azonban helyzetek, amikor egy bizonyos elemet szeretnénk egyedi módon formázni. Ilyenkor érdemes a kódon belüli stílusokhoz (inline style) folyamodnunk, amelyek lehetővé teszik, hogy az oldal kis részét – akár egyetlen elemét – formázzuk. Ilyen szabályokat létrehozhatunk a `<p>`, a `<div>` vagy a `` elemekben, csak vegyük használatba a `style` jellemző-jüket. Azért beszélünk kódon belüli stílusokról, mert a meghatározásuk ezúttal a HTML-kód belsejében található.

Lássuk most a `style` jellemző használatát egy rövid példán:

```
<p style="color:green">
  This text is green, but <span style="color:red">this text is
  red.</span>
  Back to green again, but...
</p>
<p>
  ...now the green is over, and we're back to the default color
  for this page.
</p>
```

A fenti példában láthatjuk, hogyan használhatjuk a `` elemet arra, hogy egy kódon belüli stílusszabállyal módosítsuk a szöveg színét. Ha jobban megfigyeljük a kódot, láthatjuk, hogy mind a `<p>`, mind a `` elem feltünteti a kódbeli stílusmeghatározásban a `color` tulajdonságot. Fontos megértenünk, hogy a `` és a `` címkék közötti szöveg esetében a `color:red` stílus felülírja a `color:green` beállítást. A második bekezdésre viszont ezeknek a `color` tulajdonságoknak egyike sincs hatással, ugyanis ez egy teljesen új bekezdés, amelynek a tartalma követi az oldal alapértelmezett színbeállítását.



4.3. ábra

A W3C CSS Validator meggyőz arról, hogy a 4.1. példa stíluslapja mentes mindenféle hibától

Összefoglalás

Ezen az órán megtanultuk, hogy a stíluslapokkal egyszerre több HTML-oldal megjelenését is befolyásolhatjuk. A segítségükkel hihetetlenül pontosan beállíthatjuk a HTML-elemek formátumát, térközeit és elhelyezését. Megtudtuk azt is, hogy a `style` jellemzőt szinte bármely HTML-elemre alkalmazhatjuk, meghatározva ezzel a HTML-oldal tetszőleges részének stílusát anélkül, hogy egy külön stíluslap-dokumentumra hivatkoznánk.

Megismerkedtünk a stíluslapok beépítésének három módszerével. Eszerint alkalmazhatunk önálló, `.css` kiterjesztéssel ellátott stíluslapfájlt, amelyet a weboldalaink `<head>` részében a `<link />` elemmel vehetünk használatba, megadhatjuk a stíluslapot a HTML-oldal fejlécében a `<style>` elemmel, továbbá a `style` jellemzővel közvetlenül is elhelyezhetjük a stílusszabályokat az egyes HTML-elemeknél. A 4.1. táblázatban összefoglaljuk az óra során megismert elemeket. A CSS 2 stíluslap-szabványról részletesebb tájékoztatást a <http://www.w3c.org> címen kaphatunk – itt megtudhatjuk, milyen formázási lehetőségeket kapunk a `<style>` elemen, illetve a `style` jellemzőn belül.

4.1. táblázat A 4. órán bemutatott HTML-elemek és jellemzőik

Elem/jellemzők	Leírás
<code><style>...</style></code>	Lehetővé teszi, hogy egy dokumentumba belső stíluslapot ágyazzunk be. Kizárólag a <code><head></code> és a <code></head></code> címkék között használható.
Jellemző	
<code>type="tartalomtípus"</code>	Az internetes tartalomtípus (a CSS-stíluslapok esetében ez mindig <code>"text/css"</code>).
<code><link /></code>	Külső stíluslapot (vagy más dokumentumot) kapcsol egy weboldalhoz. A dokumentum <code><head></code> részében használható.
Jellemző	
<code>href="url"</code>	A stíluslap címe.
<code>type="tartalomtípus"</code>	Az internetes tartalomtípus (a CSS-stíluslapok esetében ez mindig <code>"text/css"</code>).
<code>rel="stylesheet"</code>	A kapcsolat típusa. (A stíluslapok esetében mindig <code>"stylesheet"</code> .)
<code>...</code>	Önmagában semmiféle hatása nincs, mindössze lehetőséget ad a <code>style</code> és más jellemzők hatásának kifejtésére. (Hasonló a <code><div>...</div></code> elemhez, de nem okoz sortörést.)
Jellemző	
<code>style="stílus"</code>	Itt helyezhetjük el a kódon belüli stílusmeghatározásainkat. (A <code></code> , a <code><div></code> , a <code><body></code> és szinte bármely egyéb HTML-elemen belül alkalmazhatjuk.)

Kérdezz-felelek

- K: *Tegyük fel, hogy egy weboldalhoz egy olyan stíluslapot csatolunk, amelynek a beállításai szerint az oldalon megjelenő minden szöveg kék színt kap. Van azonban a kódban valahol egy `` elem. Milyen színnel jelenik meg az ebben szereplő szöveg – kékkel vagy vörössel?*
- V: Vörössel. A helyi, kódon belüli stílusok mindig felülírják a külső stíluslapok beállításait. Hasonlóképpen, a `<style>` és a `</style>` címkék között, az oldal elején feltüntetett stílusok szintén előnyt élveznek a külső stíluslapok beállításai-val szemben (a kódon belüli stílusok azonban ezeknél is erősebbek). Ez éppen a stíluslapok korábban említett rangsoroltságának köszönhető. A kép egyszerű: a külső stíluslapok beállításait felülírják a belső stíluslap szabályai, ezeket pedig a kódon belüli stílusok üthetik ki a nyeregből.
- K: *Hozzákapcsolhatunk egy weboldalhoz több stíluslapot is?*
- V: Persze. Készíthetünk egy külön stíluslapot a formázás (szöveg, betűtípusok, színek stb.), egy másikat pedig az elrendezés (margók, kitöltés, igazítás stb.) számára – majd mindkettőt csatolhatjuk a megfelelő `<link />` elemmel. A CSS szabvány valójában azt követeli meg a böngészőktől, hogy ha több stíluslap érhető el `<link />` elemek révén, tegyék lehetővé a választást a felhasználó számára – a gyakorlatban azonban a legtöbb böngésző egyszerűen az összes így megadott stíluslapot beépíti a kódba. Ha a szabvány szerint szeretnénk több stíluslapot használatba venni, alkalmazzuk az `@import` utasítást:

```
@import url(styles1.css);
@import url(styles2.css);
```

A `<link />` elemhez hasonlóan az `@import` utasítást is a weboldal `<head>` részében kell elhelyeznünk. Erről a hasznos utasításról bővebben a 20. órán lesz szó, amikor megtanuljuk, hogyan készítsünk stíluslapot kifejezetten weboldalak nyomtatására.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Milyen kóddal érhetjük el, hogy a címsoraink 30 pontos, kék Arial betűkkel jelenjenek meg, míg az oldal többi szövegrésze kétszeres közül, 10 pontos Times Roman betűtípussal (illetve a böngésző alapértelmezett betűtípusával) kerüljön a képernyőre?
2. Tegyük fel, hogy az előzőekben elkészített stíluslapot `corporate.css` néven tároltuk. Hogyan alkalmaznánk egy `intro.html` nevű weboldalra?
3. Hányféleképpen gondoskodhatunk arról, hogy a stílusszabályok kifejtsék a hatásukat a tartalomra?

Válaszok

1. Helyezzük el a stíluslapon az alábbi kódot:

```
h1 { font: 30pt blue Arial; }
body { font: 10pt blue; }
```

2. Írjuk be a következő kódsort az `intro.html` dokumentum `<head>` és `</head>` címkéi közé:

```
<link rel="stylesheet" type="text/css" href="corporate.css" />
```

3. Háromféleképpen: külső, belső és kódon belüli stílusokkal.

Gyakorlatok

- Készítsünk szabványos stíluslapot a webhelyünk számára, és csatoljuk az összes benne szereplő weboldalhoz. (Azoknak az oldalaknak az esetében, ahol el kell térnünk ettől, alkalmazzunk belső, illetve kódon belüli stílusokat.) Ha valamilyen cégnek dolgozunk, jó eséllyel már adottak a nyomtatott anyagok betűtípus- és formázási beállításai. Szerezzük meg ezeket a szabályokat, és alkalmazzuk a cég weboldalain is.
- Feltétlenül látogassunk el a stíluslapok hivatalos szabványoldalára (<http://www.w3.org/Style/CSS/>), és próbáljunk ki néhányat azok közül a különlegesebb stílustulajdonságok közül is, amelyeknek a bemutatására ezen az órán nem nyílt alkalom.



5. ÓRA

Szövegblokkok és listák

A lecke tartalma:

- Szöveg igazítása az oldalon belül
- A HTML három listatípusának használata
- Listák a listákban

A Web hőskorában a szöveget egyetlen betűtípussal és egyforma méretű betűkkel jelenítettük meg. Mostanra azonban a HTML és a CSS együttes használatával a betűk alakján felül a szöveg igazítását és megjelenését is megadhatjuk a weboldalainkon. Ezen az órán megismerkedünk a szövegigazítás alapvető műveleteivel, és bemutatunk pár olyan haladó fogást, mint a listák használata. Minthogy a listák igencsak elterjedtek, a HTML külön címkével biztosítja a listák automatikus behúzását, az egyes listaelemek számozását, illetve felsorolásjelekkel való ellátását. Végül, az oldalainkon lévő tartalom bemutatására szolgáló sok-sok módszer közül a különféle listatípusok formázásával ismerkedünk meg.



Önálló feladat

Készítsünk elő magunknak mintaszöveget!

Akkor sajátíthatjuk el a leghatékonyabban az óra folyamán bemutatott szövegformázási módszereket, ha előkészítünk magunknak egy mintaszöveget, amelyen aztán a behúzást, a középre igazítást és az egyéb műveleteket gyakorolhatjuk.

- Bármilyen vázlat, egy bemutatóból származó felsorolás, számozott lépések, vagy egy adatbázisból származó, szöveges információt tartalmazó lista jó gyakorlóanyag lesz.
- Mindegy, hogy milyen a szöveg, de érdemes olyat keresnünk, illetve begépelnünk, amely kikerülhet egy weboldalra. Valamilyen céges prospektus vagy az önéletrajzunk éppen megfelelő lesz.
- Ha a használni kívánt szöveg valamilyen szövegszerkesztő vagy adatbázis-kezelő programból származik, akkor mindenképpen mentsük új fájlba, egyszerű, formázás nélküli szöveggént, avagy ASCII formátumban. A mentett szöveget a lecke során majd ellátjuk a megfelelő HTML-címkékkel és stílusjellemzőkkel.
- A fejezetben megismert, a szövegtörzs tartalmának formázására szolgáló kód használatba vétele előtt lássuk el a szöveget az előző órákon használt, az oldal vázát kialakító HTML-címkékkel – a `<html>`, a `<head>`, a `<title>` és a `<body>` címkéről van szó.

Szöveg igazítása az oldalon belül

Megszoktuk, hogy a Weben olvasott szöveg nagy része balra van igazítva. Mindazonáltal gyakran találkozhatunk olyan helyzetekkel is, amikor a szöveget szévesen igazítanánk jobbra, vagy akár középre. A HTML lehetőséget ad arra, hogy a tömb- vagy blokkszintű elemeket – például a `<p>` és `</p>`, illetve a `<div>` és `</div>` címkék között lévő szöveget – külön-külön igazítsuk a megfelelő helyre. Mielőtt azonban a blokkelemek igazításának részleteit kezdenénk tárgyalni, egy gyors kitérőt teszünk, hogy megbeszéljük, miként működnek a HTML-jellemzők.

A jellemzők használata

A jellemzők tárolják a HTML-elemekhez kapcsolódó információkat. A *jellemzők* olyan különleges kódszavak, amelyeket egy HTML-elem belsejében használunk, és amelyek megadják, hogy az elem pontosan milyen hatású legyen. A legkisebb webes tartalomban is alapvető a szerepük, így igen fontos, hogy elsajátítsuk a használatuk módját.

A jellemzők használata együtt jár a bizonyos elemekre érvényes stílusok, osztályok, illetve azonosítók használatával. Ha egy stíluslapon a 4. órán tanultak szerint megadunk valamilyen osztályt vagy azonosítót, akkor az osztály, illetve az azonosító úgy érhető el, hogy magában a címkében a `class="osztály_neve"`, illetve az `id="azonosító_neve"` kódot szerepeltetjük. Miközben a böngésző a tartalom megjelenítését végzi, megnézi a stíluslapot, és az ott megadott stílusok ismeretében határozza meg a megjelenítés pontos módjáról. A fentiekhez hasonlóan a `style` jellemző használatával is megadhatók a különféle elemek formázására vonatkozó információk, anélkül, hogy az elemet egy stíluslaphoz kapcsolnánk. Amikor például a `<p>` elemmel megnyitunk egy bekezdést, a `style` jellemzővel adható meg, hogy a szöveg az adott bekezdésen belül a bal vagy a jobb margóhoz rendeződjön, vagy az oldal közepére igazodjon. Ha az adott bekezdést egy már létező osztályhoz, illetve azonosítóhoz kívánjuk rendelni, akkor a `class`, illetve az `id` jellemzőt kell beállítanunk.

Az alábbi példa mindhárom bekezdése balra van igazítva:

```
<p style="text-align: left;">Ide kerül a szöveg.</p>
<p class="leftAlignStyle">Ide kerül a szöveg.</p>
<p id="firstLeftAlign">Ide kerül a szöveg.</p>
```

Az első bekezdés esetében a formázást közvetlenül a `style` jellemzőben adjuk meg. A második bekezdés akkor lesz balra igazított, ha a stíluslap `leftAlignStyle` nevű bejegyzése tartalmazza a szöveg igazítására vonatkozó utasítást. Ehhez hasonló a harmadik bekezdés is, amennyiben az is akkor lesz balra igazítva, ha a stíluslap `firstLeftAlign` osztálya tartalmazza a szöveg igazítására vonatkozó utasítást.

Az előző példában, és azokban is, amelyeket az előző órák során mutattunk be, az Olvasónak esetleg feltűnt, hogy a címkéket, jellemzőket és stílusokat kisbetűvel írtuk. Az XHTML szabvány vonatkozó része megköveteli, hogy a jellemzőket kisbetűvel írjuk, csakúgy, mint a jellemzők értékét közrefogó idézőjelet.

A következő kódot például a legtöbb népszerű webböngésző megjelenítené:

```
<P STYLE=TEXT-ALIGN:CENTER>
```

A kód ugyanakkor nem elégíti ki az XHTML szabvány előírásait, mivel a címke nagybetűs, a `style` jellemző és az értéke (`text-align:center`) szintűgy, ráadásul az érték nincs idézőjelben. Ha meg szeretnénk felelni a legújabb szabványnak, illetve a legújabb alkalmazásoknak, akkor tanácsos a fenti alak szerint mindig az alábbi írti:

```
<p style="text-align:center">
```


A blokkszintű elemek igazítása

Ha az olyan blokkszintű elemeket, mint a `<p>`, anélkül szeretnénk a jobb margóhoz igazítani, hogy a stíluslapon külön osztályt, illetve azonosítót hoznánk létre, egyszerűen helyezzük el a bekezdés elején, a `<p>` címke belsejében a `style="text-align:right"` jellemzőt. Hasonló az elem középre igazítása is: ekkor a `style="text-align:center"` jellemzőt kell használnunk. Ha az oldalt balra kívánjuk igazítani, használjuk a `<p style="text-align:left">` kódot.



A HTML minden jellemzőjének és stílusszabályának van alapértelmezett értéke. Ha mi nem adunk meg értéket, a böngészők ezt az alapértelmezett értéket használják. A `<p>` elem `text-align` stílusszabálya esetében az alapértelmezett érték a `left` (balra). Így, ha a `<p>` elemen belül semmit nem állítunk be, akkor az eredmény ugyanaz lesz, mintha a `<p style="text-align:left">` kódot használtuk volna. Ha jó weboldalfelkészítők szeretnénk lenni, akkor fontos, hogy a stílusszabályok alapértelmezett értékét megtanuljuk.

A `text-align` stílusszabályt nem csupán a `<p>` elemekben használhatjuk. Sőt, az a helyzet, hogy a `text-align` stílusszabály bármely blokkszintű elemekben előfordulhat, azaz többek között találkozhatunk vele a `<h1>`, a `<h2>` és a többi címsort jelölő elemekben éppúgy, mint a `<div>` elemekben. A `<div>` elem különösen kézreálló, mivel alkalmas más, blokkszintű elemek körbezárására. Így lehetővé válik a webes tartalom nagyobb részleteinek egy egységként történő igazítása. A `<div>` elemekben lévő `div` a *division*, azaz *szakasz* szó rövidítése.

Az 5.1. példa bemutatja, hogy a `<p>` és a `<div>` elemekben miként használható a `style` jellemző és a `text-align` stílusszabály. Az eredményt az 5.1. ábrán láthatjuk. A `<div>` elem sok bonyolultabb, a könyv III. részében ismertetett helyzetben is használható.

5.1. példa A style jellemzőben használt text-align stílusszabály

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

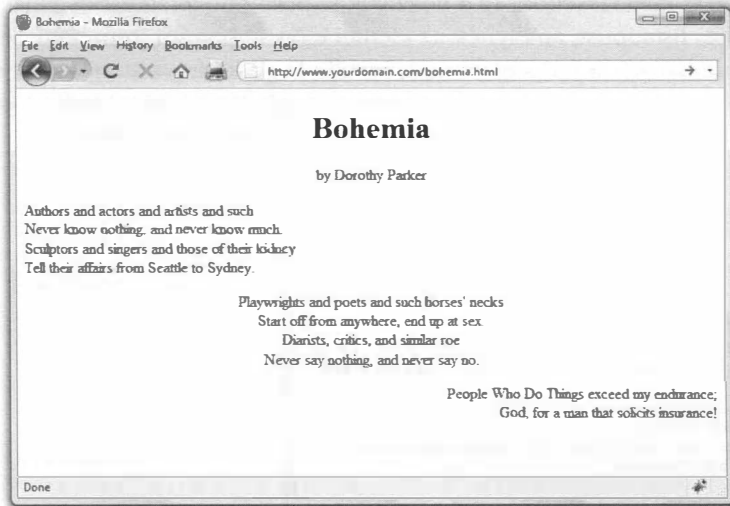
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Bohemia</title>
  </head>

  <body>
    <div style="text-align:center">
      <h1>Bohemia</h1>
      <h2>by Dorothy Parker</h2>
    </div>
```

```

<p style="text-align:left">
  Authors and actors and artists and such<br />
  Never know nothing, and never know much.<br />
  Sculptors and singers and those of their kidney<br />
  Tell their affairs from Seattle to Sydney.
</p>
<p style="text-align:center">
  Playwrights and poets and such horses' necks<br />
  Start off from anywhere, end up at sex.<br />
  Diarists, critics, and similar roe<br />
  Never say nothing, and never say no.
</p>
<p style="text-align:right">
  People Who Do Things exceed my endurance;<br />
  God, for a man that solicits insurance!
</p>
</body>
</html>

```



5.1. ábra

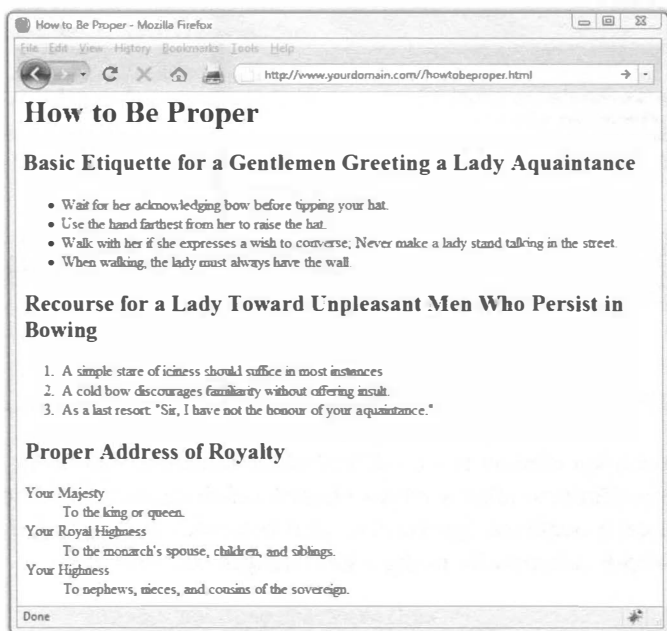
Az 5.1. példában használt szövegigazítások eredménye

Az oldal tartalma – beleértve a két címsort is – a `<div style="text-align:center">` elem hatására kerül középre. Mindazonáltal a `<div>` elemen belüli egyes bekezdések szövegigazítása felülbírálja ezt a beállítást. Így kerül az első bekezdés tartalma a bal margóhoz, a másodiké középre, a harmadik pedig a jobb margóhoz.

A HTML három listatípusa

A tisztább átláthatóság végett gyakran van szükség arra, hogy egy weboldalon az információt listaként jelenítsük meg. A HTML három alapvető listatípust ismer. Az 5.2. ábrán mindhármát bemutatjuk, az 5.2. példa pedig a kialakításukhoz szükséges HTML-kódot ismerteti.

- A rendezett listák olyan behúzott listák, amelyben az egyes listaelemek számozottak. A rendezett listák az `` címkével kezdődnek, és a `` címkével zárulnak. A lista elemeit a `` és a `` címkepár veszi közre. Minden `` címke automatikus sortörést eredményez. A teljes lista behúzott.
- A rendezetlen listák olyan behúzott listák, amelyben az egyes listaelemeket valamilyen felsorolásjel előzi meg. A rendezetlen listák az `` címkével kezdődnek, és a `` címkével zárulnak. A lista elemeit a rendezett listák elemeihez hasonlóan a `` és a `` címkepár veszi közre. Minden nyitó `` címke sortörést eredményez, illetve megjeleníti a felsorolásjelet is. A teljes lista behúzott.
- A meghatározáslisták kifejezésekből és a hozzájuk kapcsolódó jelentésből állnak. Ezt a különleges listatípust, amelynek az elemei előtt sem szám, sem betű, sem egyéb jel nem található, a `<dl>` és a `</dl>` címke fogja közre. Az egyes kifejezések a `<dt>` és a `</dt>` címkék közé kerülnek, a meghatározásokat pedig a `<dd>` és `</dd>` címkék között találjuk. A sortörés és a behúzás automatikus.



5.2. ábra

A HTML-listák három alaptípusa

5.2. példa *Rendezetlen listák, rendezett listák és meghatározáslisták*

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>How to Be Proper</title>
  </head>

  <body>
    <h1>How to Be Proper</h1>
    <h2>Basic Etiquette for a Gentlemen Greeting a Lady Acquaintance</h2>
    <ul>
      <li>Wait for her acknowledging bow before tipping your hat.</li>
      <li>Use the hand farthest from her to raise the hat.</li>
      <li>Walk with her if she expresses a wish to converse; Never
        make a lady stand talking in the street.</li>
      <li>When walking, the lady must always have the wall.</li>
    </ul>
    <h2>Recourse for a Lady Toward Unpleasant Men Who Persist in Bowing</h2>
    <ol>
      <li>A simple stare of iciness should suffice in most instances.</li>
      <li>A cold bow discourages familiarity without offering insult.</li>
      <li>As a last resort: "Sir, I have not the honour of your
        acquaintance."</li>
    </ol>
    <h2>Proper Address of Royalty</h2>
    <dl>
      <dt>Your Majesty</dt>
      <dd>To the king or queen.</dd>
      <dt>Your Royal Highness</dt>
      <dd>To the monarch's spouse, children, and siblings.</dd>
      <dt>Your Highness</dt>
      <dd>To nephews, nieces, and cousins of the sovereign.</dd>
    </dl>
  </body>
</html>

```



Ne feledjük, hogy az egyes webböngészők ugyanazt a tartalmat meglehetősen eltérően jelenítik meg. A HTML szabvány nem adja meg pontosan, hogy a webböngészőknek hogyan kell a listákat formázniuk. Így a régebbi böngészővel rendelkezők esetleg nem ugyanazt a behúzást látják, amit mi. A listaelemek elhelyezésének pontos beállítására a CSS-t használjuk – már a mai óra későbbi részében is.

Listák a listákban

Bár a meghatározáslisták elvileg arra valók, hogy kifejezések magyarázatát adjuk meg velük, sok weboldalkészítő minden olyan helyen meghatározáslistát használ, ahol behúzást szeretne látni. A gyakorlatban bármely szöveg behúzása gyorsan megoldható, ha elé `<dl>` és `<dd>`, mögé pedig `</dd>` és `</dl>` címkéket teszünk. A szöveg behúzására mindazonáltal megfelelőbb a `<blockquote></blockquote>` címkepár, így ugyanis anélkül történik meg a szöveg behúzása, hogy valamiféle meghatározás jelenlétére következtetnénk, ráadásul az oldal formázása is lényegesen áttekinthetőbb lesz. Megvannak ugyanis az elem területének szélességét, magasságát, háttérszínét, kerettípusát és színét, illetve egyéb tulajdonságait beállító jellemzők.

Minthogy a CSS használatával a listaelemek megjelenése remekül beállítható, a behúzás látványát úgy is elérhetjük, hogy nem használunk *egymásba ágyazott listákat*. Egymásba ágyazott listákat tehát ne használjunk, csak ha a tartalom kifejezetten ezt igényli. Más szóval, az egymásba ágyazott listákat akkor érdemes használni, ha az információ hierarchikusan tagolt – ilyet látunk az 5.3. példában.



Az „egymásba ágyazott”, illetve a „beágyazott” kifejezést olyankor használjuk, amikor egy elem teljes egészében egy másik elem belsejében jelenik meg. A beágyazott elemeket szokás az őket tartalmazó elem (a szülőelem) gyermekelemeinek is nevezni. Elterjedt – bár nem megkövetelt – webfejlesztői eljárás, hogy a beágyazott elemeket behúzással különítjük el, így könnyen áttekinthető, hogy milyen a gyermek- és a szülőelem viszonya.

5.3. példa *Listák használata vázlat készítésére*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Vertebrates</title>
  </head>

  <body>
    <h1>Vertebrates</h1>
    <ul>
      <li><span style="font-weight:bold">Fish</span>
        <ul>
          <li>Barramundi</li>
          <li>Kissing Gourami</li>
          <li>Mummichog</li>
        </ul>
      </li>
```

```

<li><span style="font-weight:bold">Amphibians</span>
  <ul>
    <li>Anura
      <ul>
        <li>Goliath Frog</li>
        <li>Poison Dart Frog</li>
        <li>Purple Frog</li>
      </ul>
    </li>
    <li>Caudata
      <ul>
        <li>Hellbender</li>
        <li>Mudpuppy</li>
      </ul>
    </li>
  </ul>
</li>
<li><span style="font-weight:bold">Reptiles</span>
  <ul>
    <li>Nile Crocodile</li>
    <li>King Cobra</li>
    <li>Common Snapping Turtle</li>
  </ul>
</li>
</ul>
</body>
</html>

```



5.3. ábra

A Firefox a többszintű rendezetlen listákat szépen behúzza, és a szinteket eltérő felsorolási jelekkel is megkülönbözteti. Az ilyen listákat könnyebb áttekinteni

Ahogy az 5.3. ábrán látjuk, alaphelyzetben rendszerint a korong az első szintű felsorolásjel, a második szintű a kör, a harmadik és a többi szintet pedig kitöltött négyzet jelzi. A felsorolásjel ugyanakkor akár szintenként is megadható, ha az egyszerű `` címke helyett az `<ul style="list-style-type:disc">`, az `<ul style="list-style-type:circle">`, illetve az `<ul style="list-style-type:square">` címkét használjuk, amelyek rendre korongot, kört, illetve négyzetet állítanak be felsorolásjelként.

A rendezetlen listák belsejében az egyes listaelemek felsorolásjele is megadható, mégpedig úgy, hogy a `` címke belsejében a `list-style-type` stílusszabályt használjuk. Az alábbi kód például az extra és a super szó előtt kört, a special szó előtt pedig négyzetet jelenít meg:

```
<ul style="list-style-type:circle">
  <li>extra</li>
  <li>super</li>
  <li style="list-style-type:square">special</li>
</ul>
```

A `list-style-type` stílusszabály a rendezett listák esetében is használatos, de ekkor nem a felsorolásjelet állítja be, hanem az egyes listaelemek elé kerülő szám, illetve betű típusát adja meg. Az 5.4. példa bemutatja, hogy egy többszintű listában miként használhatunk római számokat (`list-style-type:upper-roman`), nagybetűket (`list-style-type:upper-alpha`), kisbetűket (`list-style-type:lower-alpha`) és arab számokat. Az 5.4. ábrán a kód eredményeképp kialakult, szépen formázott vázlat látható.

Az 5.4. példában a `list-style-type` stílusszabályt csak az `` címkében használjuk, de használhatnánk a listák egyes `` elemei esetében is – még akkor is, ha hirtelen nem tudunk egyetlen olyan esetet sem, amikor ennek értelme lenne. Az alapértelmezés szerinti arab számok használatát be is állíthatjuk, mégpedig a `list-style-type:decimal` stílusszabállyal. A kisbetűs római számok használata a `list-style-type:lower-roman` stílusszabállyal állítható be.

5.4. példa *Többszintű lista készítése a list-style-type stílusszabály, illetve a style jellemző használatával*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Advice from the Golf Guru</title>
  </head>

  <body>
    <h1>How to Win at Golf</h1>
    <ol style="list-style-type:upper-roman">
```

```

<li>Training
  <ol>
    <li>Mental prep
      <ol style="list-style-type:upper-alpha">
        <li>Watch golf on TV religiously</li>
        <li>Get that computer game with Tiger whatsisname</li>
        <li>Rent "personal victory" subliminal tapes</li>
      </ol>
    </li>
    <li>Equipment
      <ol style="list-style-type:upper-alpha">
        <li>Make sure your putter has a pro autograph on it</li>
        <li>Pick up a bargain bag of tees-n-balls at Costco</li>
      </ol>
    </li>
    <li>Diet
      <ol style="list-style-type:upper-alpha">
        <li>Avoid junk food
          <ol style="list-style-type:lower-alpha">
            <li>No hotdogs</li>
          </ol>
        </li>
        <li>Drink wine and mixed drinks only, no beer</li>
      </ol>
    </li>
  </ol>
</li>
<li>Pre-game
  <ol>
    <li>Dress
      <ol style="list-style-type:upper-alpha">
        <li>Put on shorts, even if it's freezing</li>
        <li>Buy a new hat if you lost last time</li>
      </ol>
    </li>
    <li>Location and Scheduling
      <ol style="list-style-type:upper-alpha">
        <li>Select a course where your spouse or boss won't
          ➡ find you</li>
        <li>To save on fees, play where your buddy works</li>
      </ol>
    </li>
    <li>Opponent
      <ol style="list-style-type:upper-alpha">
        <li>Look for: overconfidence, inexperience</li>
        <li>Buy opponent as many pre-game drinks
          ➡ as possible</li>
      </ol>
    </li>
  </ol>
</li>
<li>On the Course

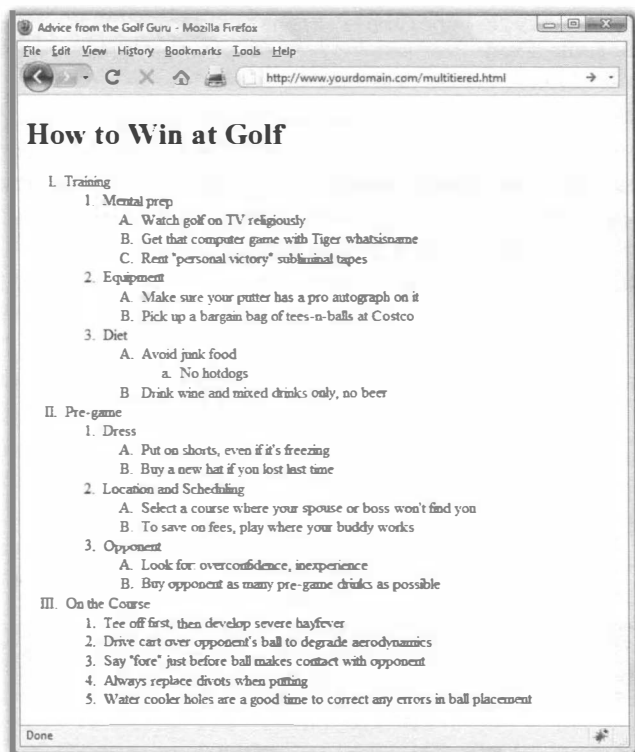
```



```

<ol>
  <li>Tee off first, then develop severe hayfever</li>
  <li>Drive cart over opponent's ball to degrade aerodynamics</li>
  <li>Say "fore" just before ball makes contact with opponent</li>
  <li>Always replace divots when putting</li>
  <li>Water cooler holes are a good time to correct any errors
    ➡ in ball
    placement</li>
</ol>
</li>
</ol>
</body>
</html>

```



5.4. ábra

Egy szépen formázott vázlat szinte minden kialakítást tetszetősebbé tehet

Összefoglalás

Ezen az órán kiderült, hogy a HTML-elemek beállításait és különféle viselkedésmódjait jellemzők segítségével adhatjuk meg. Megtanultunk szöveget igazítani a `style` jellemzőben tárolt CSS-stílusszabályok használatával, továbbá megtanultuk a HTML három alapvető listatípusának – rendezett lista, rendezetlen lista és meghatározáslista – kialakítását és együttes használatát. Vázlatok és egyéb összetett szövegelrendezések megjelenítésekor szükségessé válhat a listák más listákon belüli elhelyezése.

Az 5.1. táblázat a mai órán megismert elemeket és jellemzőket foglalja össze. Nem muszáj az összes címkét fejből tudnunk: azért van ez a könyv, hogy ha valamelyik elemre szükségünk van, kikereshessük belőle. Ne feledjük, hogy a „B” függelékben valamennyi HTML-elem szerepel.

11.1. táblázat Az ötödik órán tárgyalt HTML-elemek és -jellemzők

Elem/Jellemző	Leírás
<code><div>...</div></code>	A formázásra váró szövegrész.
<code><dl>...</dl></code>	Meghatározáslista.
<code><dt>...</dt></code>	A meghatározáslista részeként szereplő meghatározandó fogalom.
<code><dd>...</dd></code>	A meghatározáslista részeként szereplő, a meghatározandó fogalomhoz tartozó meghatározás.
<code>...</code>	Rendezett (számozott) lista.
<code>...</code>	Rendezetlen (felsorolásjeles) lista.
<code>...</code>	Az <code></code> és az <code></code> listában szereplő listaelemek.
Jellemzők	
<code>style="text-align: igazítás"</code>	A szöveget középre (center), balra (left) vagy jobbra (right) igazítja. (A <code><p></code> , a <code><h1></code> , a <code><h2></code> , a <code><h3></code> stb. címkékkel is használható.)
<code>style="list-style-type: számtípus"</code>	A listaelemek számozására szolgáló szám típusa. A lehetséges értékek: decimal (arab szám), lower-roman (római szám kisbetűvel), upper-roman (római szám nagybetűvel), lower-alpha (kisbetűk), upper-alpha (nagybetűk) és none (semmi).

11.1. táblázat Az ötödik órán tárgyalt HTML-elemek és -jellemzők

(folytatás)

Elem/Jellemző	Leírás
<code>style="list-style-type: felsorolásjel"</code>	A listaelemek felsorolásjelének típusa. A lehetséges értékek: <code>disc</code> (korong), <code>circle</code> (kör), <code>square</code> (négyzet) és <code>none</code> (semmi).
<code>style="list-style-type: típus"</code>	A listaelemet jelző felsorolásjel, illetve szám típusa. A lehetséges értékek: <code>disc</code> , <code>circle</code> , <code>square</code> , <code>decimal</code> , <code>lower-roman</code> , <code>upper-roman</code> , <code>lower-alpha</code> , <code>upper-alpha</code> és <code>none</code> .

Kérdezz-felelek

- K: Láttam olyan weboldalakat, amelyek háromdimenziós gömböket vagy egyéb különleges grafikát használnak. Hogy lehet ilyet készíteni?
- V: Ez a téma meghaladja ennek a fejezetnek a kereteit, de a 11. órán ezt is megtanuljuk.
- K: Hogyan lehet egy szöveget sorkizárttá tenni, azaz azt megoldani, hogy a bal margótól a jobb margóig érjen a szöveg?
- V: A formázás megadásakor használjuk a `text-align: justify` stílusszabályt.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

- Hogyan lehet egy oldalon belül mindent középre rendezni?
- Hogyan lehet egyetlen szót behúzni, illetve négyzetet tenni elé?
- Hogyan néz ki az a HTML-kód, amely megadja, hogy a „glunch” angol szó jelentése az, hogy „a look of disdain, anger, or displeasure”, illetve a „glumpy” angol szó annyit tesz, hogy „sullen, morose, or sulky”?

Válaszok

1. Ha arra gondoltunk, hogy a `<div style="text-align:center">` címkét közvetlenül az oldal tetején lévő `<body>` címke alá, a `</div>` címkét pedig közvetlenül az oldal alján lévő `</body>` címke fölé helyezzük, akkor okosan döntöttünk. A `text-align` stílusjellemzőt mindazonáltal maga a `<body>` elem is támogatja, azaz elhagyhatjuk a `<div>` elemet, és a `style="text-align:center"` stílusszabály kerülhet közvetlenül a `<body>` elembe – ezzel az egész oldal tartalmát középre igazítjuk.
2. Használjuk az alábbi kódot:

```
<ul style="list-style-type:square">
  <li>supercalifragilisticexpealidocious</li>
</ul>
```

(Ha a `style="list-style-type:square"` stílusszabályt magában a `` elemben helyeznénk el, az is ezzel az eredménnyel járna, mivel a lista egyetlen elemből áll.)

3. Használjuk az alábbi kódot:

```
<dl>
<dt>glunch</dt><dd>a look of disdain, anger, or displeasure</dd>
<dt>glumpy</dt><dd>sullen, morose, or sulky</dd>
</dl>
```

Gyakorlatok

- Helyezzünk el a weboldalunk különböző részein szövegblokkokat a szövegigazításra szolgáló stílusjellemzők segítségével. Próbálkozzunk a bekezdések és szakaszok (a `<p>` és a `<div>` elem) egymásba ágyazásával – így fogunk ráérezni arra, hogy a stílusok hatása hogyan érvényesül a tartalmi hierarchia egyes szintjein.
- Állítsunk elő számozott listát azokból a dolgokból, amelyeket a honlapunkon szerepeltetni kívánunk. Ezzel nemcsak a HTML-listák formázását gyakoroljuk, de felmerülnek azok a problémák is, amelyekkel a könyv későbbi fejezeteiben fogunk foglalkozni.



6. ÓRA

Betűformázás

A lecke tartalma:

- Hogyan állíthatunk be félkövér és dőlt betűket és egyéb betűalakokat?
- Haladó betűbeállítások
- Különleges karakterek használata

Az előző órán megismerkedtünk a szövegblokkok létrehozásának alapvető fogásaival, illetve a szöveg listává alakításával. Ezen az órán magukat a szöveget alkotó elemeket vesszük közelebből szemügyre, és megtanuljuk, hogyan módosítható a betűk kinézete – például a betűtípuscsalád, a méret és a betűvastagság. Megtudjuk, hogy miként használhatunk a weboldalainkon félkövér, dőlt, illetve áthúzott betűket, és azt, hogy miként alakíthatjuk a betűket felső, illetve alsó indexűvé. Megismerjük a betűtípus és a betűméret beállításának módját is.



Más webtervezők munkáit áttekintve a szövegformázásnak olyan módszereivel is találkozunk majd, amelyek eltérnek a könyvben ismertetett eljárásoktól. A „régimódi” szövegformázás eszközei közé tartozik a félkövér szövegrészek jelölésére szolgáló ` `, a dőlt betűvel írandó szöveget jelölő `<i> </i>`, valamint a betűtípus, a betűméret és egyéb jellemzők beállítására használható ` ` címkepár. Mostanra azonban ezeket semmi értelme megtanulnunk, hiszen lassan kipusztulnak a HTML-ből, a CSS pedig érezhetően hatékonyabb lehetőségeket kínál.



Önálló feladat

Újabb feladat, újabb mintaszöveg

Az előző órán készítettünk magunknak egy mintaszöveget tartalmazó munkafajlt. Készítsünk most is egyet, és a fejezet során próbáljuk ki a megismert módszereket – ezzel is elősegítve a hatékony tanulást.

Minthogy az ezen az órán előkerülő információ nagy része karakterszintű formázás, nem igazán van jelentősége annak, hogy milyen szöveget használunk. Olyan sok kipróbálható lehetőség van, hogy egy weboldalon belül semmiképp sem fordulhat elő valamennyi változat – hacsak meg nem akarjuk bolondítani a weboldal látogatóit. Használjuk ki az alkalmat, és próbáljunk ráérezni, milyen hatást gyakorolnak a szöveg-szintű változtatások a weboldalunk kinézetére.

Félkövér, dőlt és egyéb betűalakok

Réges-régen, amikor még az írógépek korát éltük, elégedettek voltunk az egyszerű szöveggel, illetve a nyomatékositást jelentő, itt-ott előforduló aláhúzással. Manapság minden papír alapú szövegben szinte kötelező a **félkövér** és a *dőlt* betűk használata. Természetesen a weboldalainkon is használhatunk félkövér és dőlt betűket; a szövegformázást számos címke és stílusszabály segíti.



A félkövér, illetve a dőlt szöveget eredményező stílusszabályok helyett használhatjuk a `` és a ``, illetve az `` és `` címkéket is. A `` címke hatása a legtöbb böngészőben a `` címke hatásával egyezik meg, az `` címke hatása pedig az `<i>` címkéjével – mindkét utóbbi elem dőlt betűket eredményez.

A `` és az `` címkét sokan tekintik előrelépésnek a `` és az `<i>` használatához képest, mivel az előbbiek csak annyit jeleznek, hogy a szöveg nyomatékositást igényel, de nem szabják meg pontosan, hogy azt hogyan kell megvalósítani. Más szóval, a böngészőnek nem kell a `` elemet szükségszerűen félkövér szöveggént értelmeznie, és az `` sem jelent feltétlenül dőlt betűt. Így a `` és az `` elem jobban beleillik az XHTML világába, hiszen jelentéstartalommal is gazdagítják a szöveget, nem csak a megjelenítését szabályozzák. Mind a négy elem része marad a HTML 5 szabványnak is, bár a szerepüket valamivel jobban megkülönböztetik.

A félkövér, illetve dőlt betűk kialakításának régimódi eszköze a `` `` és az `<i>` `</i>` címkepár. Ha félkövér szöveget szeretnénk írni, akkor a szöveg elején a ``, a szöveg végén pedig a `` címkét kell elhelyeznünk. Dőlt betűs szöveget hasonlóan

készíthetünk: a szöveget `<i>` és `</i>` címkékkel kell körbevennünk. Bár a módszer a böngészőkben remekül működik, és az XHTML is támogatja, nem olyan rugalmas és ügyes, mint a CSS stílusszabályaival megvalósított szövegformázás.

Bár a könyv III. részében lényegesen többet tudunk meg a CSS stílusszabályairól, érdekes kicsit izlelgetnünk a témát, mert ez elősegíti a szövegformázás megértését. A betűk vastagságának, illetve kövérségének szabályozása a `font-weight` stílusszabállyal válik lehetővé. A `font-weight` stílusszabály lehetséges értékei a `normal` (normál), a `bold` (félkövér), a `bolder` (kövérebb) és a `lighter` (vékonyabb); az alapértelmezett érték a `normal`. Dőlt betű a `font-style` stílusszabállyal állítható be. A szabály lehetséges értékei: `normal`, `italic` (dőlt) és `oblique` (ferde). Ha egyszerre több stílusszabályt is szeretnénk alkalmazni, akkor együttesen is megadhatjuk őket. Ezt mutatja az alábbi példa:

```
<p style="font-weight:bold; font-style:italic">Ez a bekezdés félkövér  
és dőlt!</p>
```

A fenti példában mindkét stílusszabályt a `<p>` elem `style` jellemzőjében adtuk meg. Több stílusszabály együttes használatakor figyelniünk kell az őket elválasztó pontosvesszőre (;) is.

A betűformázásokat nem csak a bekezdésekben használhatjuk. Az alábbi példa azt mutatja be, hogy miként használható dőlt betű egy felsorolásjeles listában:

```
<ul>  
  <li style="font-style:italic">Important Stuff</li>  
  <li style="font-style:italic">Critical Information</li>  
  <li style="font-style:italic">Highly Sensitive Material</li>  
  <li>Nothing All That Useful</li>  
</ul>
```

A `font-weight` stílusszabályt címsorokban is használhatjuk, de a vastagítás általában nem mutatkozik meg, mivel a címsorok alaphelyzetben is vastagítottak.

Bár a CSS-nek köszönhetően a formázási lehetőségek kibővülnek, van pár olyan HTML-elem, amely akkor is alkalmazható, ha eltekintenénk a CSS használatával együtt járó lehetőségek kiaknázásától. Az alábbiakban pár ilyen elemet ismertetünk. Az egyes elemek működését a 6.1. példa és a 6.1. ábra szemlélteti.

- `<small></small>` Kis betűméretű szöveg.
- `<big></big>` Nagy betűméretű szöveg – a HTML 5 nem tartalmazza, mivel a szöveg mérete hatékonyabban szabályozható CSS-sel.
- `` Felső indexben lévő szöveg.
- `` Alsó indexben lévő szöveg.
- ``, illetve `<i></i>` Nyomatékosított (dőlt betűs) szöveg.
- ``, illetve `` Vastagított (félkövér) szöveg.

- `<tt></tt>` Egyenlő szélességű karaktereket tartalmazó szöveg (írógépbetűk) – a HTML 5 nem tartalmazza, mivel a betűk mérete hatékonyabban szabályozható CSS-sel.
- `<pre></pre>` Egyenlő szélességű karaktereket tartalmazó szöveg, amely megőrzi a szóközöket és a sortöréseket.



Aláhúzott szöveget korábban az `<u>` címkével hoztunk létre, de ma már több okunk is van arra, hogy erről leszokjunk. Először is, a felhasználók arra számítanak, hogy az aláhúzott szöveg egy hivatkozás, ezért zavaró lehet, ha olyasmit húzunk alá, ami nem az. Másodszor, az `<u>` elem használata ellenjavallt, ami annyit jelent, hogy kikerül a HTML/XHTML nyelvből – ahogy a `<strike>` (áthúzás) elem is. A böngészők mindkét elemet megértik, és alighanem még sokáig meg is fogják, de aláhúzott vagy áthúzott szöveg létrehozásához a CSS használatát szorgalmazzák.

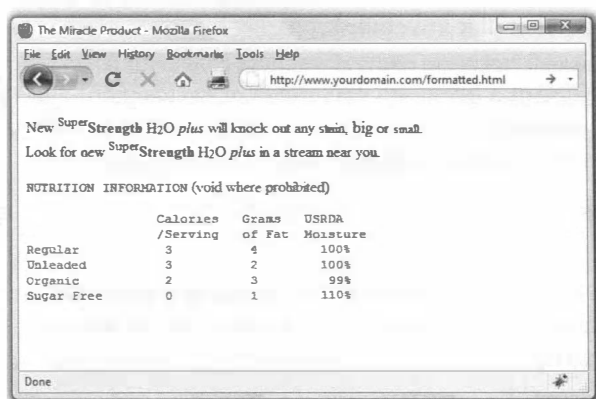
6.1. példa Különleges formázást megvalósító elemek

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>The Miracle Product</title>
  </head>

  <body>
    <p>
      New <sup>Super</sup><strong>Strength</strong> H<sub>2</sub>O
      <em>plus</em> will knock out any stain, <big>big</big> or
      <small>small</small>.<br /> Look for new
      <sup>Super</sup><b>Strength</b> H<sub>2</sub>O <i>plus</i>
      in a stream near you.
    </p>
    <p>
      <tt>NUTRITION INFORMATION</tt> (void where prohibited)
    </p>
    <pre>
      Calories      Grams      USRDA
      /Serving      of Fat      Moisture
Regular           3           4          100%
Unleaded          3           2          100%
Organic           2           3           99%
Sugar Free        0           1          110%
    </pre>
  </body>
</html>
```


A `<tt>` címke hatására a szöveg rendszerint Courier New betűtípussal jelenik meg. Ez egy olyan betűtípus, amelyben minden karakter egyenlő szélességű. Mindazonáltal a webböngészők a felhasználó számára lehetővé teszik, hogy a `<tt>` címke által használt, egyenlő szélességű karakterekből álló betűtípust egy másik, általuk választott betűtípusra cseréljék (nézzünk szét a böngészőnk beállításai között). Így elképzelhető, hogy az egyenlő szélességű karaktereket tartalmazó betűtípus nem is tartalmaz egyenlő szélességű karaktereket, bár a felhasználók túlnyomó része nem változtatja meg a böngésző alapértelmezett betűbeállításait.



6.1. ábra

A 6.1. példa karakterformázásainak eredménye

A `<pre>` címke hatására a böngésző szintén egyenlő szélességű karakterekkel írja a szöveget, de a címkének van egy másik, igen-igen hasznos hatása is. Ahogy a 3. órán megtanultuk, a böngésző rendszerint figyelmen kívül hagyja a HTML-fájlokban lévő sortöréseket és többszörös szóközöket. A `<pre>` címke hatására mind a szóközök, mind a sortörések megmaradnak. A `<pre>` címke nélkül a 6.1. ábra végén lévő szöveg például nagyjából így nézne ki:

```
calories grams usrda /serving of fat moisture regular 3 4 100% unleaded
3 2 100% organic 2 3 99% sugar free 0 1 110%
```

Hiába helyezünk el minden sor végén egy `
` címkét, az oszlopok akkor sem kerülnek egymás alá. Ha azonban a szövegrész elejére `<pre>`, a végére pedig `</pre>` címkét írunk, akkor az oszlopok megmaradnak oszlopnak, ugyanis nem maradnak el a szóközök. Sőt, így a `
` címkékre sincs szükség. A `<pre>` címke egyszerű és gyors módját kínálja annak, hogy az egyenlő szélességű karakterekből álló szövegfájljainkat az igazítást megőrizve, minimális erőfeszítéssel alakítsuk webes tartalomná.

A CSS a szöveg elrendezésére – és minden más szövegműveletre – hatékonyabb módszereket kínál, amelyekről a könyv III. részében tanulunk majd részletesen.

Haladó betűbeállítások

A weboldalainkon lévő szövegek méretét és kinézetét kezdetlegesen a `<big>`, a `<small>` és a `<tt>` címke használatával módosíthatjuk. Akadhatnak azonban olyan helyzetek, amikor jó lenne csak egy kicsit pontosabban megadni a szöveg méretét és kinézetét. Mielőtt azonban az XHTML-kód lehetőségeivel belefognánk a betűkkel való büttyölésbe, röviden tekintsük át, hogy miként mentek a dolgok a CSS megjelenése előtt – elképzelhető ugyanis, hogy más weboldalak forráskódját böngészve találunk még ilyen megoldásokat. Ne feledjük, hogy bár a régmódi eljárások még használatban vannak, nekünk már az új módszereket illik követnünk.

A stíluslapok színrelépését megelőzően a mostanra elavultnak számító `` elemet használták a weboldalakon lévő szövegek tulajdonságainak megadására. Az alábbi HTML-kód például a weboldalon lévő szöveg méretét és színét változtatja meg:

```
<font size="5" color="purple">Ez a szöveg nagy méretű és  
➡ lila lesz.</font>
```



A szövegszín beállításáról bővebben a 9. órán tanulunk, amikor is megismerjük az egyéni színek előállításának módját, valamint azt is, hogy miként adható meg a szöveges hivatkozások színe.

Ahogy látjuk, a `` elem `size` (méret) és `color` (szín) jellemzői teszik lehetővé azt, hogy a szöveg betűit különösebb erőfeszítés nélkül megváltoztassuk. Bár a módszer remekül működött, a CSS stílusszabályainak köszönhetően mára egy lényegesen jobb módszer váltotta fel. Az alábbiakban a betűk beállítását végző főbb stílusszabályok közül mutatunk be néhányat:

- `font-family`: A használt betűtípust állítja be.
- `font-size`: A betűk nagyságát adja meg.
- `color`: A betűk színét módosítja.

A `font-family` stílusszabály a szöveg megjelenítését végző betűtípus beállítását teszi lehetővé. Ennél a formázásnál több, vesszővel elválasztott érték is megadható, és érdemes is többet megadnunk. Így ha a felhasználó rendszerén nincs telepítve az elsőként megadott betűtípus, akkor a böngésző megpróbálkozhat egy másik használatával. A módszerrel az előző órák során már találkoztunk. A további betűtípusok megadása azért fontos, mert elvileg minden felhasználó gépén más-más betűtípusok vannak telepítve, legalábbis a szokásos alaptípusokon – Arial, Times New Roman stb. – felül. Ha más használható betűtípusokat is megadunk, akkor nagyobb az esély arra, hogy a szöveg végső soron valamilyen általunk is kipróbált betűtípussal jelenik meg azokban az esetekben, amikor a kedvencünk nincs telepítve. Az alábbi példa egy olyan `font-family` beállítást ismertet, amely a szöveg egy bekezdésében használni kívánt betűtípust adja meg:

```
<p style="font-family:arial, sans-serif, 'times roman'">
```

A példa számos érdekességet rejt magában. Először is, elsődleges betűtípusként az arial-t adtuk meg. A betűtípus-beállításban a nagybetűknek nincs szerepe, azaz az arial, az Arial és az ARIAL érték azonos hatású. A másik érdekesség, hogy a times roman névnél aposztrófokat használtunk, mégpedig azért, mert a névben szóköz található. Mindazonáltal, mivel a 'times roman' típus az általánosabb sans-serif (talp nélküli) értéket követi, nem valószínű, hogy a használatára sor kerül. A második helyen álló sans-serif érték ugyanis a böngésző számára ezt jelenti: „Ha a gépen nincs Arial betűtípus, akkor használd az alapértelmezett talp nélküli betűtípust.”



A tizenhatos számrendszerben megadott (hexadecimális) színekódokról a 9. órán fogunk tanulni. Most csak annyit kell megértenünk, hogy a green (zöld), a blue (kék), az orange (narancssárga) és hasonló színeknel a color stílusszabállyal pontosabban is beállítható a betűszín.

A font-size és a color stílusszabályt is elterjedten használják, mégpedig a betűk méretének és színének beállítására. A font-size beállítás kaphat előre meghatározott értéket – ilyen a small (kicsi), a medium (közepes) és a large (nagy) –, de pontosan is megadható, pontban kifejezve, azaz például 12pt vagy 14pt formában. A color formázás értéke lehet valamilyen előre megadott érték – például white (fehér), black (fekete), blue (kék), red (piros), avagy green (zöld) –, de megadható pontosan, a szín tizenhatos számrendszerű kódjával is, például így: #FFB499. Az alábbiakban az előző bekezdés-beállítást vesszük elő újra, de ezúttal a betűk méretét és színét is megadjuk:

```
<p style="font-family:arial, sans-serif, 'times roman'; font-size:14pt;
➡ color:green">
```

A 6.2. példa és a kódnak a 6.2. ábrán látható eredménye olyan betűtípus-beállításokat ismertet, amelyekkel már nekiláthatunk egy egyszerű internetes önéletrajz elkészítésének.

6.2. példa Egyszerű internetes önéletrajz betűtípus-beállításai

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>R&acute;sum&acute; for Jane Doe</title>

<style type="text/css">
  body {
    font-family: Verdana, sans-serif;
    font-size: 12px;
  }
```

```

h1 {
    font-family:Georgia, serif;
    font-size:28px;
    text-align:center;
}

p.contactinfo {
    font-size:14px;
    text-align:center;
}

p.categorylabel {
    font-size:12px;
    font-weight:bold;
    text-transform:uppercase;
}

div.indented {
    margin-left: 25px;
}
</style>
</head>
<body>
    <h1>Jane Doe</h1>
    <p class="contactinfo">1234 Main Street, Sometown,
    CA 93829<br/>
    tel: 555-555-1212, e-mail: jane@doe.com</p>

    <p class="categorylabel">Summary of Qualifications</p>
    <ul>
    <li>Highly skilled and dedicated professional offering a
    solid background in whatever it is you need.</li>
    <li>Provide comprehensive direction for whatever it is
    that will get me a job.</li>
    <li>Computer proficient in a wide range of industry-related
    computer programs and equipment. Any industry.</li>
    </ul>

    <p class="categorylabel">Professional Experience</p>
    <div class="indented">
        <p><span style="font-weight:bold;">Operations Manager,
        Super Awesome Company, Some City, CA [Sept 2002 -
        present]</span></p>
        <ul>
        <li>Direct all departmental operations</li>
        <li>Coordinate work with internal and external
        resources</li>
        <li>Generally in charge of everything</li>
        </ul>
        <p><span style="font-weight:bold;">Project Manager,
        Less Awesome Company, Some City, CA [May 2000 - Sept
        2002]</span></p>
        <ul>

```

```

        <li>Direct all departmental operations</li>
        <li>Coordinate work with internal and external
        resources</li>
        <li>Generally in charge of everything</li>
    </ul>
</div>

<p class="categorylabel">Education</p>
<ul>
<li>MBA, MyState University, May 2002</li>
<li>B.A, Business Administration, MyState University,
May 2000</li>
</ul>

<p class="categorylabel">References</p>
<ul>
<li>Available upon request.</li>
</ul>
</body>
</html>

```



6.2. ábra

A 6.2. példa eredményeként megjelenő weboldal

A példában CSS-t használunk – ami, amint azt a 4. órán megtanultuk, a formázásokat osztályokba rendezi –, és látjuk, hogy a tartalom különböző részein miként érvényesül a szövegformázás. Ha alaposabban megfigyeljük a `div.indented` osztályt, akkor felfigyelhetünk a `margin-left` beállításra. Ez a beállítás – amelyről a könyv II. részében fogunk tanulni – adott méretű térkört (példánkban 25 képpontnyit) helyez el az elem bal oldala mellett. Ez a térköz alakítja ki a 6.2. ábrán látható behúzásokat.

Különleges karakterek használata

Ma már a legtöbb betűtípus tartalmazza az európai nyelvek számára szükséges ékezetes betűket – például a *Café* szóban található *é* betűt (sokból kimarad azonban a magyar *ű* és *ő*). Van ezen felül még néhány matematikai jel, és pár, a szöveg központozása során használható jel, például a háromszög alakú felsorolásijel.

A 6.1. táblázatban ismertetett kódok használatával a HTML-dokumentum bármely részén elhelyezhetjük az ilyen különleges karaktereket. A http://www.webstandards.org/learn/reference/named_entities.html weboldalon ennél a táblázatnál kimerítőbb, több karakterkészletre vonatkozó listákat találunk.

A *café* szó leírásához például az alábbi módszerek közül mindkettő használható:

```
caf&eacute;
caf&#233;
```

6.1. táblázat Gyakran használt különleges karakterek az angol nyelvterületről

Karakter	Szám kód	Név kód	Leírás
"	"	"	idézőjel
&	&	&	& (és) jel
<	<	<	kisebb, mint
>	>	>	nagyobb, mint
¢	¢	¢	a cent (fizetőegység) jele
£	£	£	a font (fizetőegység) jele
	¦ vagy &brkbar;	¦	két részből álló függőleges vonal (cső)
§	§	§	paragrafus
©	©	©	copyright, a szerzői jog tulajdonosa
*	®	®	bejegyzett üzleti védjegy
°	°	°	a fok (mértékegység) jele
+–	±	±	plusz-mínusz
²	²	²	felső indexű kettes számjegy
³	³	³	felső indexű hármas számjegy

6.1. táblázat Gyakran használt különleges karakterek az angol nyelvterületről (folytatás)

Karakter	Szám kód	Név kód	Leírás
.	·	·	sort közepén lévő pont
¹	¹	¹	felső indexű egyes számjegy
$\frac{1}{4}$	¼	¼	egynegyed (törtszám)
$\frac{1}{2}$	½	½	egyketted, fél (törtszám)
$\frac{3}{4}$	¾	¾	háromnegyed (törtszám)
Æ	Æ	Æ	nagybetűs A-E kötés (ligatúra)
æ	æ	æ	kisbetűs a-e kötés (ligatúra)
É	É	É	ékezetes nagy É
é	é	é	ékezetes kis é
×	×	×	szorzásjel
÷	÷	÷	osztásjel

Bár a fenti karakterek számmal is megadhatók, mindegyiknek megvan a könnyebben fejből tartható, a nevéből származó kódja is.



A copyright jelét (©), vagy esetleg a bejegyzett üzleti védjegyét (®) keressük? A megfelelő kódok: © és ®. A bejegyzetlen védjegy (™) pedig az ™ kóddal csalogatható elő.

A HTML, illetve az XHTML nyelv egy különleges kódot, egy úgynevezett karakterkódot vagy karakteregységet (character entity) használ az olyan különleges karakterek megjelenítésére, mint a © és az ®. A karakterkódokat az & jellel kell kezdenünk, és pontosvesszővel kell zárunk. A 6.1. táblázatban megtaláljuk a leggyakrabban használt karakterkódokat, bár a HTML ennél lényegesen többet ismer.

A táblázatban szerepel a „kisebb, mint”, a „nagyobb, mint”, az idézőjel és az & karakter kódja is. Ha ezeket a karaktereket szeretnénk egy weboldalon használni, akkor a kódjaikkal kell őket helyettesítenünk, különben a böngésző HTML-parancsként fogja értelmezni őket.

A 6.3. példában és a 6.3. ábrán számos, a 6.1. táblázatban szereplő jelet mutatunk be használat közben.

6.3. példa A különleges karakterek kódolása

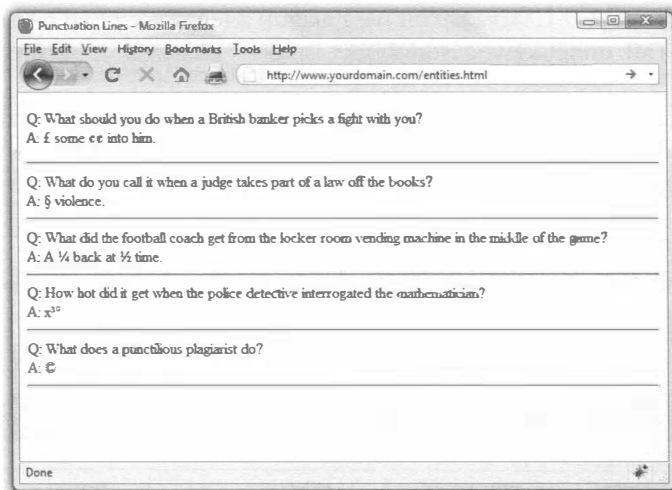
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Punctuation Lines</title>
  </head>

  <body>
    <p>
      Q: What should you do when a British banker picks a fight
      with you?<br />
      A: &pound; some &cent;&cent; into him.
    <hr />
    Q: What do you call it when a judge takes part of a law
    off the books?<br />
    A: &sect; violence.
    <hr />
    Q: What did the football coach get from the locker room
    vending machine in the middle of the game?<br />
    A: A &frac14; back at &frac12; time.
    <hr />
    Q: How hot did it get when the police detective interrogated
    the mathematician?<br />
    A: x&sup3;&deg;
    <hr />
    Q: What does a punctilious plagiarist do?<br />
    A: &copy;
    <hr />
  </p>
</body>
</html>

```



6.3. ábra

A 6.3. példa eredményeként megjelenő weboldal így néz majd ki a legtöbb böngészőben

Összefoglalás

Ezen az órán megtanultuk, hogyan írhatunk félkövér vagy dőlt betűs szöveget, hogyan helyezhetjük a szöveget felső vagy alsó indexbe, és miként szűrhatunk be különleges karaktereket és ékezetes betűket. Láttuk, hogy az előzetesen formázott, egyenlő szélességű karakterekből álló szövegek hasábjait hogyan rendezhetjük egymás alá, és azt is, hogy miként állítható be egy weboldal bármely szövegrészletének mérete, színe és betűtípusa.

A 6.2. táblázat az órán tárgyalt elemeket és jellemzőket foglalja össze. Ne feledjük, hogy minden HTML-elem szerepel a „B” függelékben is.

6.2. táblázat A 6. órán tárgyalt HTML-elemek és -jellemzők

Elem/Jellemző	Leírás
<code>...</code>	Nyomatékosítás (rendszerint dőlt betű).
<code>...</code>	Erősebb nyomatékosítás (rendszerint félkövér betű).
<code>...</code>	Félkövér szöveg.
<code><i>...</i></code>	Dőlt betűs szöveg.
<code><tt>...</tt></code>	Egyenlő szélességű karakterek (írógépbetűk).
<code><pre>...</pre></code>	Előformázott szöveg (megőrzi a sortöréseket és a szóközök számát; rendszerint egyenlő szélességű karakterekkel jelenik meg).
<code><big>...</big></code>	A szokásosnál némileg nagyobb szöveg.
<code><small>...</small></code>	A szokásosnál némileg kisebb szöveg.
<code><sub>...</sub></code>	Alsó index.
<code><sup>...</sup></code>	Felső index.
Jellemzők	
<code>style="font-family:betűtípus"</code>	A betűkészlet (betűtípus) neve, például Arial (használható a <code><p></code> , a <code><h1></code> , a <code><h2></code> , a <code><h3></code> stb. elemekben is).
<code>style="font-size:méret"</code>	A betű mérete. Megadható a <code>small</code> , a <code>medium</code> és a <code>large</code> , valamint az <code>x-small</code> (nagyon kicsi), és az <code>x-large</code> (nagyon nagy) stb. érték. Beállítható a pontban mért méret megadásával is (például <code>12pt</code>).
<code>style="color:szín"</code>	A szöveg színét változtatja meg.

Kérdezz-felelek

- K: *Hogyan tudhatom meg egy a gépemre telepített betűtípus pontos nevét?*
- V: Windows, illetve Macintosh operációs rendszeren nyissuk meg a Vezérlőpultot (Control Panel), és kattintsunk a betűtípusok mappájára (Fonts). A Windows Vista-felhasználóknak lehet, hogy a Vezérlőpult beállítását „Classic View”-ra (Klasszikus nézet) kell változtatniuk. Amikor a font-family stílusszabályban adjuk meg a betűtípus nevét, figyeljünk a név pontos beírására – bár megjegyeznénk, hogy a nagy kezdőbetűkre nem kell figyelniük, erre nem érzékeny a beállítás.
- K: *Miként írhatunk egy weboldalra az európaiától eltérő, például japán, arab vagy kínai betűkkel?*
- V: Először is, azoknak a felhasználóknak, akik el szeretnék olvasni az ilyen betűkkel írt szövegeket, telepíteniük kell a megfelelő nyelvi betűtípusokat, majd a böngészőjükben be kell állítaniuk ezeknek a betűknek a használatát. A jelekhez tartozó számkódokat az egyes nyelvi betűkészletekben a Windows Character Map (Karaktertábla) nevű programjával – más operációs rendszeren pedig valamilyen hasonló célú programmal – nézhetjük meg. A Character Map a Start, All Programs, Accessories, System Tools (Start, Minden program, Kellékek, Rendszereszközök) menüből érhető el. Ha például a 214-es számú karakterre van szükségünk, akkor a weboldal fájljában az &214; bejegyzést kell megadnunk. Ha nem találjuk a Character Map alkalmazást, nyissuk meg az operációs rendszer beépített súgóját, és ott keressük meg a program pontos helyét.

Ha csak rövid üzeneteket szeretnénk elhelyezni valamilyen ázsiai nyelven (mondjuk azt, hogy „Beszélünk tamil nyelven, hívjon bátran!), akkor a legokosabb, ha azt grafika-ként, egy képen helyezzük el. Így minden felhasználó elolvashatja a mondanivalónkat, még akkor is, ha a böngészője elsődleges nyelve az angol. Persze a különleges betűket tartalmazó grafika elkészítéséhez is valószínűleg le kell töltenünk az operációs rendszer megfelelő nyelvi csomagját. Ismét csak azt mondhatjuk, hogy a pontos teendőket a rendszerünk súgójában találjuk meg.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Hogyan hozható létre olyan bekezdés, amelyben az első három szó félkövér, ha a ``, illetve `` elem helyett inkább stíluszabályt használunk?
2. Hogyan ábrázolható a víz kémiai képlete?
3. Hogyan jeleníthető meg egy weboldalon a „© 2009, Webwonks Inc.” felirat?

Válaszok

1. Használjuk az alábbi kódot:

```
<p><span style="font-weight: bold">First three words</span>
are bold.</p>
```
2. Használjuk a `H₂O` formát.
3. Az alábbi jelölések valamelyikével:

```
&copy; 2009, Webwonks Inc.
&#169; 2009, Webwonks Inc.
```

Gyakorlatok

- Alkalmazzuk a fejezetben megismert karakterszintű formázási beállításokat blokkszintű elemekre (például: `<p>`, `<div>`, `` és ``) is. Próbálkozzunk az elemek egymásba ágyazásával – így fogunk ráérezni arra, hogy a stílusok hatása miként érvényesül a tartalmi hierarchia egyes szintjein.



7. ÓRA

Információk megjelenítése táblázatok segítségével

A lecke tartalma:

- Egyszerű táblázat létrehozása
- Táblázat méretének beállítása
- A tartalom igazítása és átívelése a táblázaton belül

Ezen az órán megtanuljuk, hogyan hozhatunk létre egy weboldalon a sorba rendezhető adatok térközeit, elrendezését és kinézetét szabályozó HTML-táblázat. Bár a CSS használatával is elérhető hasonló eredmény, vannak esetek, amikor az információ megjelenítésének határozottan egy táblázat a legmegfelelőbb módja. Megtapasztaljuk, hogy a táblázatok mennyire hasznosak tudnak lenni, ha sorokba, illetve oszlopokba rendezhető adatokat kell megjeleníteni. Elmagyarázzuk, hogy a weboldalak tervezői a múltban miként használtak táblázatokat az oldalak elrendezésének tárolására, és azt is, hogy ez sok esetben miért nem okos döntés. Mielőtt azonban belekezdzenénk az órába, annyit rögzítsünk magunkban, hogy a táblázat nem egyéb, mint függőleges oszlopokba és vízszintes sorokba rendezett adatok együttese.



Önálló feladat

A tartalom táblázatos megjelenítése

A fejezet olvasása során gondolkodjunk el arról, hogy a szöveges tartalom táblázattá alakítása miként válhat egy weboldal előnyére. Íme néhány megszívlelendő tanács:

- A táblázatok a legnyilvánvalóbban kimutatások – például nevek és számok több oszlopra kiterjedő listája – rendezett megjelenítésére alkalmasak.
- Minden olyan alkalommal használhatunk táblázatokat, amikor több szövegszlopra vagy egymás mellé kerülő képekre van szükség.

Az óra példái alapján készítsünk saját táblázatot egy weboldalunkra. A fejezet végén található gyakorlatok részletes tanácsokkal szolgálnak ehhez.

Egyszerű táblázat létrehozása



Van néhány stílustulajdonság, amellyel igen pontosan állítható be a táblázatok szegélye. Beállítható például a szegély vastagsága (a `border-width` tulajdonsággal), formája (a `border-style` tulajdonsággal) és a színe (a `border-color` tulajdonsággal). Ezek a tulajdonságok remekül működnek, de a táblázat minden egyes eleménél meg kell adnunk őket, és ez még akkor sem pihentető, ha a sorok, illetve cellák formázását CSS-osztályokkal végezzük.

A táblázatokban az információt sorokba rendezzük, és minden sorban több önálló cella is lehet. A táblázatok kialakítását a `<table>` (táblázat) címkével kezdjük, és a végükre természetesen a `</table>` címke kerül. Ha szegélyt is szeretnénk rajzoltatni a táblázatunknak, akkor a szegély szélességét a `border` (szegély) címke használatával adhatjuk meg, képpontban kifejezve. A `0`, illetve `none` érték azt jelzi, hogy a táblázat szegélye láthatatlan marad – akkor is ez a helyzet, ha a `border` jellemzőt elhagyjuk –, ami például akkor jöhet jól, ha az adott táblázattal az oldal elrendezésének kialakítását végezzük.

Ha a `<table>` címke a helyére került, a következő szükséges címke a `<tr>` lesz. A `<tr>` elem a táblázat egy sorát (table row) alakítja ki. Egy sorba legalább egy cella kerül, és az utolsó cellát követi a sort lezáró `</tr>` címke. Az egyes cellák létrehozására a `<td>` elemet (table data) használjuk. Maga az információ a `<td>` és a `</td>` címke közé kerül. A cella az a téglalap alakú terület, amely bármilyen szöveget, képet és HTML-elemet tartalmazhat. Ha egy sorban több cella is található, akkor kialakulnak a táblázat oszlopai is.

A táblázatok kialakítása során még egy alapvető elemet használunk. A `<th>` elem pontosan úgy működik, mint a `<td>`, annyi különbséggel, hogy a táblázat címsorát és -oszlopát jelezzük vele (*table heading* – táblázatcím). A legtöbb webböngésző a `<th>` elemmel kialakított cellák tartalmát félkövér, középre igazított betűkkel jeleníti meg.

Annai cellát hozunk létre, amennyit kedvünk tartja, de a táblázat minden sorában pontosan annyi cellának kell lennie, mint a többiben. A 7.1. példa HTML-kódja mindössze az eddig említett négy táblázatcímke használatával alakít ki egy egyszerű táblázatot. A 7.1. ábrán a kód alapján rajzolt táblázat látható, úgy, ahogy az egy webböngészőben megjelenik.

7.1. példa *Táblázat kialakítása a <table>, a <tr>, a <td> és a <th> elemmel*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Baseball Standings</title>
  </head>

  <body>
    <h1>Baseball Standings</h1>
    <table>
      <tr>
        <th>Team</th>
        <th>W</th>
        <th>L</th>
        <th>GB</th>
      </tr>
      <tr>
        <td>Los Angeles Dodgers</td>
        <td>62</td>
        <td>38</td>
        <td>-</td>
      </tr>
      <tr>
        <td>San Francisco Giants</td>
        <td>54</td>
        <td>46</td>
        <td>8.0</td>
      </tr>
      <tr>
        <td>Colorado Rockies</td>
        <td>54</td>
        <td>46</td>
        <td>8.0</td>
```

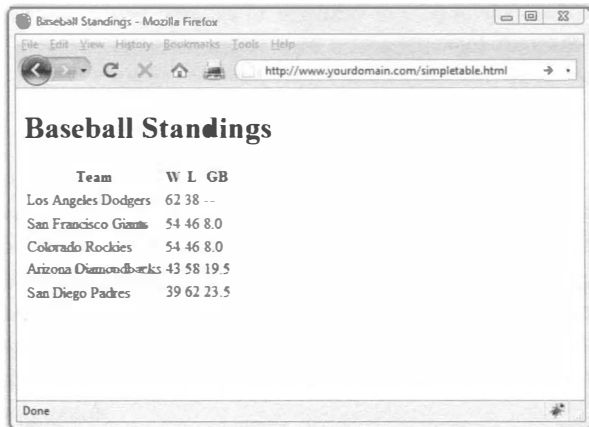
```

</tr>
<tr>
  <td>Arizona Diamondbacks</td>
  <td>43</td>
  <td>58</td>
  <td>19.5</td>
</tr>
<tr>
  <td>San Diego Padres</td>
  <td>39</td>
  <td>62</td>
  <td>23.5</td>
</tr>
</table>
</body>
</html>

```



Ha több szócímet teszünk a szavak és a címkék közé, a HTML figyelmen kívül hagyja azokat. Mindazonáltal könnyebben olvasható – és így kevesebb időpocsékoló hibát tartalmazó – táblázat készíthető, ha a 7.1. példában is látható módon behúzzuk a `<tr>` és a `<td>` címkéket.



7.1. ábra

A 7.1. példa alapján kialakuló táblázat hat sorból és négy oszlopból áll

A példatáblázatunk egy baseball-bajnokság állását tartalmazza. Az információ éppen táblázatba – sorokba és oszlopokba – kívánczik, bár az eredmény kicsit fapados. Az óra folyamán azonban megtudjuk, hogyan dobhatjuk fel egy kicsit a táblázatainkat. A táblázat címsorában a csapat nevét (Teams), a nyert (W – wins) és veszített (L – losses) találkozókat, illetve a vezető csapattól való lemaradást mutató jelzőszámot (GB, Games Behind) tüntettük fel.

Bár a 7.1. példában semmilyen formázást nem alkalmaztunk, az egyes cellák szövegét nyugodtan formázhatjuk. Megjegyeznénk azonban, hogy az egyes cellákban használt formázások és HTML-címkék a többi cellában nem érvényesülnek. A táblázaton kívül érvényes címkék nem fejtik ki a hatásukat a táblázat belsejében. Nézzük például a következő táblázatot:

```
<p style="font-weight:bold">
  <table>
    <tr>
      <td style="font-style:italic">hello</td>
      <td>there</td>
    </tr>
  </table>
</p>
```

A fenti példában a táblázat körül érvényes `<p>` címke segítségével mutattuk be, hogy a táblázatok belsejében nem jut érvényre a táblázaton kívül megadott címkék hatása. A „there” szó sem félkövér, sem dőlt nem lesz, ugyanis rá sem a táblázaton kívül megadott `font-weight:bold`, sem az előző cellába írt `font-style:italic` formázás nem hat. A példában lévő „hello” szó természetesen dőlt betűvel íródik ki.

Ha a „hello” és a „there” szót félkövérrel szeretnénk írni, az alábbiak szerint kell megváltoztatnunk a kódot:

```
<table style="font-weight:bold">
  <tr>
    <td style="font-style:italic">hello</td>
    <td>there</td>
  </tr>
</table>
```

A fenti példa mindkét szava félkövér, a „hello” szó ráadásul dőlt is. A formázásokat természetesen nem kötelező a `<table>` elembe, az egész táblázatra érvényesen megadni. A `font-weight:bold` formázást az egyes cellákra külön-külön is alkalmazhatjuk. Az, hogy minden kívánt cellánál megadjuk a `style="font-weight:bold"` bejegyzést, vagy a stíluslapon hozunk létre egy osztályt, és a cellákban csak a `class="osztálynév"` bejegyzést adjuk meg, már csak rajtunk múlik.

Táblázat méretének beállítása

Ha a táblázat szélességét nem adjuk meg, a táblázat, illetve az egyes cellák olyan szélesek lesznek, hogy a megadott tartalom elférjen bennük. Dönthetünk azonban a táblázat pontos méretének beállítása mellett is: ilyenkor a `<table>` elembe meg kell adnunk a `width` (szélesség), illetve a `height` (magasság) jellemzőt. Az egyes cellák szélessége és magassága is beállítható, ilyenkor a `width`, illetve a `height` jellemzőt

az adott `<td>` elembe kell elhelyeznünk. A `width` és a `height` jellemző értéke képpontban vagy százalékban adható meg. Az alábbi kód például egy ötszáz képpont széles és négyszáz képpont magas táblázatot hoz létre:

```
<table style="width:500px; height:400px">
```



Az a helyzet, hogy a HTML nyelvben is találunk `width` és `height` jellemzőt. Ezek mostanra, az XHTML megjelenésével elavultak, de más weboldaltervezők kódjában még találkozhatunk velük. A böngészők ezeket a jellemzőket is feldolgozzák, de helyettük érdemes inkább a `width` és a `height` stílustulajdonságot használnunk, ugyanis így teszünk eleget az XHTML kívánalmainak.

Ha azt szeretnénk, hogy a táblázatunk teljes szélességéből az első cella húsz, a második pedig nyolcvan százalékban részesüljön, az alábbi kódot kell írunk:

```
<table style="width:100%">
  <tr>
    <td style="width:20%">skinny cell</td>
    <td style="width:80%">fat cell</td>
  </tr>
</table>
```

Figyeljük meg, hogy a táblázat szélességét 100%-ban adtuk meg – így biztosítható, hogy a táblázat kitöltse a böngészőablak teljes szélességét. Ha a pontos, képpontban vett méret helyett százalékban adjuk meg az értékeket, a táblázat automatikusan akkora lesz, hogy kitöltse a böngészőablakot, ugyanakkor megmarad az esztétikai egyensúly is, esetünkben úgy, hogy az első cella a táblázat teljes szélességének húsz, a második pedig a nyolcvan százalékára terjed ki.

A 7.2. példában a 7.1. példából származó egyszerű táblázatot alakítottuk át, pontosan beállítva a cellák szélességét.

Team	W	L	GB
Los Angeles Dodgers	62	38	--
San Francisco Giants	54	46	8.0
Colorado Rockies	54	46	8.0
Arizona Diamondbacks	43	58	19.5
San Diego Padres	39	62	23.5

7.2. ábra

A 7.2. példa HTML-kódja hatsoros és ötoszlopos táblázatot hoz létre, pontosan megadva az egyes oszlopok szélességét

7.2. példa Táblázatcellák szélességének megadása

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Baseball Standings</title>
  </head>

  <body>
    <h1>Baseball Standings</h1>
    <table>
      <tr>
        <th style="width:35px;"></th>
        <th style="width:175px;">Team</th>
        <th style="width:25px;">W</th>
        <th style="width:25px;">L</th>
        <th style="width:25px;">GB</th>
      </tr>
      <tr>
        <td></td>
        <td>Los Angeles Dodgers</td>
        <td>62</td>
        <td>38</td>
        <td>-</td>
      </tr>
      <tr>
        <td></td>
        <td>San Francisco Giants</td>
        <td>54</td>
        <td>46</td>
        <td>8.0</td>
      </tr>
      <tr>
        <td></td>
        <td>Colorado Rockies</td>
        <td>54</td>
        <td>46</td>
        <td>8.0</td>
      </tr>
      <tr>
        <td></td>
        <td>Arizona Diamondbacks</td>
        <td>43</td>
        <td>58</td>
        <td>19.5</td>
      </tr>
    </table>
  </body>
</html>

```

```

        </tr>
        <tr>
            <td></td>
            <td>San Diego Padres</td>
            <td>39</td>
            <td>62</td>
            <td>23.5</td>
        </tr>
    </table>
</body>
</html>

```

A 7.1. és a 7.2. példa között két különbséget találunk. Az első, hogy a 7.2. példában elhelyeztünk egy újabb oszlopot, amelynek – bár az első sorában továbbra is a `<th></th>` címkepárt használjuk – nincs címsora. Az új oszlop a második sorától kezdve a hatodikig egy-egy képet tartalmaz, mégpedig az `` elemekben. A második különbség az, hogy a 7.2. példában az első sor minden `<th>` elemének szélességét egy-egy pontosan megadott értékű stílustulajdonság segítségével állítottuk be. Az első oszlop szélességének értéke 35 px (pixel, azaz képpont), a másodiké 175 px, a harmadik, negyedik és ötödik oszlopé pedig 25 px.

Figyeljük meg azt is, hogy a második sortól kezdve nem adjuk meg a cellák – a `<td>` elemek – szélességét. Az oszlopszélességet elég az első sorban megadnunk, a többi sor cellái ugyanis ugyanannak a táblázatnak a részei, ezért az első sorban megadott szélességet veszik fel. Más formázás megadása esetén – mondjuk a betűméret vagy betűszín beállításakor – azonban újra és újra meg kell adnunk a beállítást minden hasonlóan formázni kívánt résznél.

Tartalom igazítása és átívelése a táblázaton belül

Minden, amit egy táblázatcellába helyezünk, alapértelmezés szerint balra, függőlegesen pedig középre igazodik. A 7.1. és a 7.2. ábrán ezt az alapértelmezett igazítást figyelhetjük meg. Természetesen magunk is elvégezhetjük az igazítást, mind vízszintesen, mind függőlegesen – mégpedig a `text-align` és a `vertical-align` stílustulajdonságokkal.

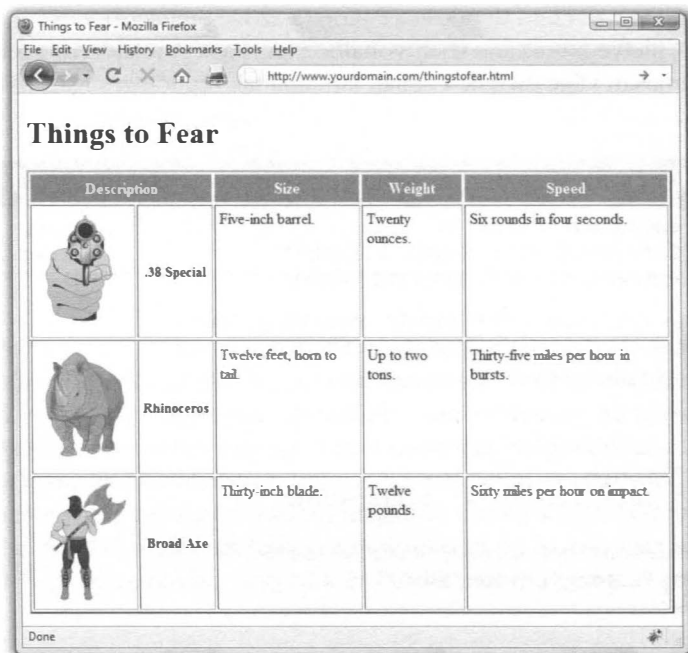
Ezeket az igazításért felelős tulajdonságokat minden `<tr>`, `<td>` és `<th>` elemben megadhatjuk. A `<tr>` elemben megadott igazítás az egész sorra érvényes. A táblázat méretétől függően jelentős időt takaríthatunk meg, ha a sorok, és nem a cellák szintjén adjuk meg az igazítást végző jellemzőket.

A 7.3. példában olvasható HTML-kódban kombináljuk a szövegigazítási lehetőségeket: az egyes sorokban megadjuk az alapértelmezett igazítást, de egy-egy cella esetében felülbíráljuk azt. A 7.3. ábrán a 7.3. példa kódjának eredményeként kialakuló weboldalt láthatjuk.

Íme néhány elterjedtebb a vertical-align stílustulajdonság lehetséges értékei közül: top (fent), middle (középen), bottom (lent), text-top (a szöveg tetejére), text-bottom (a szöveg aljára), illetve baseline (betűvonalhoz igazítás). A tulajdonság felsorolt értékeivel igen rugalmasan végezhetjük a cellák tartalmának függőleges igazítását.

7.3. példa *Igazítás, valamint térközök, szegélyek és háttérszínek beállítása a táblázatokban*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Things to Fear</title>
  </head>
  <body>
    <h1>Things to Fear</h1>
    <table border="2" cellpadding="4" cellspacing="2"
      width="100%">
      <tr style="background-color:red;color:white">
        <th colspan="2">Description</th>
        <th>Size</th>
        <th>Weight</th>
        <th>Speed</th>
      </tr>
      <tr style="vertical-align:top">
        <td></td>
        <td style="font-size: 14px;font-weight:bold;
          vertical-align:middle;text-align:center">.38 Special</td>
        <td>Five-inch barrel.</td>
        <td>Twenty ounces.</td>
        <td>Six rounds in four seconds.</td>
      </tr>
      <tr style="vertical-align:top">
        <td></td>
        <td style="font-size: 14px;font-weight:bold;
          vertical-align:middle;text-align:center">Rhinoceros</td>
        <td>Twelve feet, horn to tail.</td>
        <td>Up to two tons.</td>
        <td>Thirty-five miles per hour in bursts.</td>
      </tr>
      <tr style="vertical-align:top">
        <td></td>
        <td style="font-size: 14px;font-weight:bold;
          vertical-align:middle;text-align:center">Broad Axe</td>
        <td>Thirty-inch blade.</td>
        <td>Twelve pounds.</td>
        <td>Sixty miles per hour on impact.</td>
      </tr>
    </table>
  </body>
</html>
```



7.3. ábra

A 7.3. példa HTML-kódja az igazítási stílusok és a colspan jellemző használatát szemlélteti

A 7.3. ábra tetején egyetlen cella – a „Description” szót tartalmazó – két oszlopot ível át. Ez a cellához tartozó `<th>` elembe elhelyezett `colspan` (oszlopátívelés) jellemző hatás. Ahogy már bizonyára rájöttünk, a `rowspan` (sorátívelés) jellemző is létezik – ezzel a jellemzővel az adható meg, ha egy cellának több sornyi magasságot kell átfognia.



Amikor több oszlopon, illetve soron átívelő celláink vannak, néha nehéz megoldani, hogy a sorok és oszlopok ne keveredjenek össze. Gyakran a legkisebb hiba felboríthatja az egész táblázat szerkezetét. Sok időt és fejfájást spórolhatunk meg, ha előbb papíron felvázoljuk a táblázatot, és csak utána kezdünk hozzá a HTML nyelvű megvalósításához.

Az *átívelés* (spanning) annyit tesz, hogy egy cellát több sor, illetve oszlop nagyságúra nyújtunk. A `colspan` jellemző hatására a cella több oszlop szélességét, míg a `rowspan` jellemző hatására több sor magasságát veszi fel.

Az átívelésen felül a „Description” rész második oszlopának celláiban olyan tulajdonságokat is megadtunk, amelyeknek a hatására a szöveg félkövér betűkkel íródik ki, valamint függőlegesen és vízszintesen is a cella közepére kerül.

Van pár olyan fogás a 7.3. példában, amit még nem magyaráztunk el. A teljes táblázat, illetve a táblázat egyes cellái számára is megadható saját, a weboldal háttérétől különböző háttér. Ha háttérrel szeretnénk megadni, akkor a `<table>`, a `<tr>`, a `<td>`, illetve a `<th>` elemekben a `background-color` (háttérszín), illetve a `background-image` (háttérkép) stílustulajdonságot kell megadnunk, pontosan úgy, ahogy a `<body>` elem esetében (lásd a 9. órát). Például, ha az egész táblázat háttérszínét sárgára kívánjuk állítani, akkor a `<table style="background-color:yellow">` vagy a vele egyértékű `<table style="background-color:#FFFF00">` elemet kell megadnunk.

A 7.3. példában háttérszínt csak a felső sor kapott: a sor cellái a vörös háttérüket a `<tr style="background-color:red;color:white">` elemnek köszönhetik.

A `color` (szín) stílustulajdonság hatására pedig a sorban lévő szöveg fehér lesz.

A `background-color` tulajdonsághoz hasonló a példában nem szereplő `background-image` tulajdonság. Ez utóbbit a táblázat háttérképének megadására használjuk. Ha a `leaves.gif` képet szeretnénk háttérként beállítani, akkor a `<table style="background-image:url(leaves.gif)">` elemet kell használnunk. Figyeljük meg, hogy a képfájl neve zárójelbe kerül, elé pedig az `url` szót írjuk – ebből tudja a tulajdonság, hogy most a képfájl helye következik.



Bár az XHTML engedélyezi a `cellpadding` és a `cellspacing` jellemző használatát, `padding`, illetve `border-spacing` néven megtaláljuk a két tulajdonság CSS-beli megfelelőjét is. A használatukról bővebb információkat és példákat egyaránt a „B” függelékben olvashatunk.

A táblázatok profi kezelése nem merül ki a stílustulajdonságok használatában. Ahogy a 7.3. példából kiderül, a `cellpadding` és a `cellspacing` jellemzővel a táblázat szegélye mentén lévő üres hely nagysága szabályozható. A `cellspacing` jellemző a szegélyek és a cellák közötti helyet, a `cellpadding` pedig a cella tartalmát körülvevő üres hely méretét adja meg képpontban kifejezve. Ha a `cellpadding` jellemző értékét 0-ra állítjuk, akkor a cellák tartalma olyan közel kerül a szegélyhez, amennyire csak lehetséges – akár hozzá is érhet. A `cellpadding` és a `cellspacing` jellemzővel jól szabályozható a táblázat egészének kinézete.

Oldalelrendezés kialakítása táblázatokkal

Az óra elején már jeleztük, hogy a weboldaltervezők nem csak sorokba rendezhető információk megjelenítésére használják a táblázatokat, hanem a weboldalak elrendezését is szokás – volt – táblázatokkal kialakítani. Más webtervezők forráskódjait tanulmányozva mind a mai napig sok táblázatos elrendezést találunk. A módszer elterjedésének oka az volt, hogy az 1990-es évek közepe és a 2000-es évek eleje között a böngészők meglehetősen eltérően támogatták a CSS-t, a táblázatokat viszont minden böngésző lényegében azonos módon jelenítette meg, így aztán a weboldaltervezők inkább egy táblázatra

alapozták az oldal elrendezését, mert így tudták elérni, hogy minden böngésző egyformán jelenítse meg az oldalt. Mindazonáltal most, hogy a CSS-t minden jelentősebb böngésző viszonylag egyformán értelmezi, a webtervezőket már semmi nem tántoríthatja el attól, hogy a szabványokon alapuló ajánlásokat követve *ne* használjanak táblázatokat az oldalelrendezések kialakításához.

A World Wide Web Consortium (W3C) – a Világháló jövőjét átlátni hivatott szabványalkotó testület – nem a táblázatokat, hanem a stíluslapokat tekinti az oldalelrendezés eszközeinek. A stíluslapok lényegesen sokrétűbben használhatók a táblázatoknál – ezért foglalkozik ez a könyv is lényegében csak a CSS-sel megvalósított oldalelrendezéssel.

Lássunk pár komolyabb indokot arra nézve, hogy miért ne használjunk táblázatokat a weboldalak elrendezésének megvalósításakor:

- A táblázatok összemossák a megjelenítést a tartalommal. – A CSS és a szabványoknak megfelelő weboldaltervek egyik előnye, hogy a megjelenés elválasztható a tartalomtól.
- A táblázatok szükségtelenül bonyolítják a weboldal későbbi átalakítását. – Ha egy táblázaton alapuló webhely elrendezésén változtatni szeretnénk, akkor a webhely minden oldalát át kell írunk (kivéve ha egy bonyolult dinamikus kialakítású oldalról beszélünk – ilyenkor viszont a dinamikus tartalmat előállító részben kell mindent átírunk).
- A táblázatok megnehezítik az akadálymentesítést. – A képernyőolvasó programok a táblázatban táblázatba kívánczoló tartalomra számítanak, és megpróbálják a weboldalt ennek megfelelően felolvasni.
- A táblázatok bajossá teszik az oldalak mobileszközön történő megjelenítését. – A táblázaton alapuló elrendezések általában nem elég rugalmasak ahhoz, hogy a kis képernyőkhöz igazodhassanak (lásd a 19. órát).

Ezzel még csak néhányat említettünk a táblázatok használatán alapuló weboldaltervezés átkai közül. A felmerülő problémák részletesebb elemzése a *Miért butaság a táblázatos szerkezet?* című, a <http://www.hotdesign.com/seibold/hungarian/> weboldalon megtekinthető népszerű bemutatóban olvasható.

Összefoglalás

Ezen az órán megtanultuk, hogyan szervezhetünk szöveget és képeket sorokból és oszlopokból álló táblázatokká. Megismertük a táblázatok létrehozását szolgáló három alapvető elemet, valamint jónéhány, ezekben az elemekben használható, a táblázat igazítását, térközait és kinézetét szabályozó jellemzőt és stílust. Megtudtuk azt is, hogy a táblázatok együtt, egymásba ágyazva is használhatók, ami tovább bővíti az elrendezési lehetőségek körét.

A 7.1. táblázat a mai órán tárgyalt elemeket és jellemzőket foglalja össze.

7.1. táblázat A 7. órán tárgyalt HTML-elemek és -jellemzők

Elem/Jellemző	Leírás
<code><table>...</table></code>	Táblázatot hoz létre. A táblázat belsejében korlátlan számú sor (<code><tr></code> elem) kaphat helyet.
Jellemzők	
<code>border="szélesség"</code>	A táblázat szegélyének vastagságát adja meg képpontban kifejezve. A <code>border="0"</code> , vagy a <code>border</code> jellemző elhagyása a szegélyeket láthatatlanná teszi.
<code>cellspacing="térköz"</code>	A táblázat cellái között lévő térközt adja meg, képpontban.
<code>cellpadding="belső margó"</code>	A táblázat celláinak határa és a cellában lévő tartalom közötti térközt adja meg, képpontban.
<code>style="width:szélesség"</code>	A táblázat szélességét adja meg az oldalon. Képpontban és az oldal százalékos arányában kifejezve egyaránt megadható.
<code>style="height:magasság"</code>	A táblázat magasságát adja meg az oldalon. Képpontban és az oldal százalékos arányában kifejezve egyaránt megadható.
<code>style="background-color:szín"</code>	A táblázat és a külön beállított háttérszínrel nem bíró cellák háttérszínét adja meg.
<code>style="background-image: url(kép_elérési_útja)"</code>	A táblázat és a külön beállított háttérképpel nem bíró cellák háttérképét adja meg (ha háttérszín is megadtunk, akkor a kép átlátszó részei a háttérszín látszódik).
Elem	
<code><tr>...</tr></code>	A táblázat egy sorát adja meg; legalább egy cellát (<code><td></code> elemet) tartalmaz.
Jellemzők	
<code>style="text-align:igazítás"</code>	A cellák tartalmának vízszintes igazítását adja meg az adott soron belül. Lehetséges értékei: <code>left</code> (balra), <code>right</code> (jobbra) és <code>center</code> (középre).
<code>style="vertical-align:igazítás"</code>	A cellák tartalmának függőleges igazítását adja meg az adott soron belül. Elterjedtebb értékei: <code>top</code> (fent), <code>middle</code> (középen) és <code>bottom</code> (lent).

7.1. táblázat A 7. órán tárgyalt HTML-elemek és -jellemzők

(folytatás)

Elem/Jellemző	Leírás
<code>style="background-color: szín"</code>	A sor cellái közül a külön beállított háttérszínnel nem bíró cellák háttérszínét adja meg.
<code>style="background-image: url(kép_elérési_útja) "</code>	A sor cellái közül a külön beállított, saját háttérképpel nem bíró cellák háttérképét adja meg.
Elem	
<code><td>...</td></code>	Egy táblázatcellát határoz meg.
<code><th>...</th></code>	Egy táblázatfejléc-cellát ad meg.
Jellemzők	
<code>style="text-align: igazítás"</code>	A cellák tartalmának vízszintes igazítását határozza meg. Lehetséges értékei: left (balra), right (jobbra) és center (középre).
<code>style="vertical-align: igazítás"</code>	A cellák tartalmának függőleges igazítását adja meg. Elterjedtebb értékei: top (fent), middle (középen) és bottom (lent).
<code>rowspan="sorok_száma"</code>	Megadja, hogy hány soron íveljen át a cella.
<code>colspan="oszlopok_száma"</code>	Megadja, hogy hány oszlopon íveljen át a cella.
<code>style="width: szélesség"</code>	A cellaoszlop szélességét adja meg. Képpontban és az oldal százalékos arányában kifejezve egyaránt megadható.
<code>style="height: magasság"</code>	A cellasor magasságát adja meg. Képpontban és az oldal százalékos arányában kifejezve egyaránt megadható.
<code>style="background-color: szín"</code>	A cella háttérszínét adja meg.
<code>style="background-image: url(kép_elérési_útja) "</code>	A háttérkép, amit a cellában meg kívánunk jeleníteni.

Kérdezz-felelek

- K:** Készítettem egy nagy táblázatot. Amikor betöltöm az oldalt, sokáig üres marad az oldal, és semmi nem látszik. Miért kell várakozni?
- V:** Az összetett táblázatoknál beletelik némi időbe, amíg megjelennek a képernyőn, mert a böngészőnek még az előtt kell mindent kiszabizálnia, hogy a táblázatot akár részben megjelenítené. A sebesség némileg növelhető, ha a táblázat minden képét ellátjuk width és height jellemzővel. A width jellemző használata a `<table>` és a `<td>` elemekben is jótékony hatású.

K: *Lehetséges táblázatokat egymásba ágyazni?*

V: Igen, egy táblázat celláiban elhelyezhetők további táblázatok. Megemlítenénk azonban, hogy az egymásba ágyazott táblázatok – főleg a nagyobbak – lassan töltődnek be, és a megjelenítésük is időigényes. Mielőtt egymásba ágyazott táblázatot készítenénk, gondoljuk végig, hogy az oldalon megjeleníteni kívánt tartalom nem formázható-e CSS használatával. Lehet, hogy a könyv végigolvasását megelőzően nem tudunk minden kétséget kizáróan válaszolni erre a kérdésre, de íme egy kis segítség: a kérdésre a legtöbb esetben igenlő válasz adható.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Hogyan hozhatunk létre egy egyszerű, két sorból és két oszlopból álló, alapértelmezett szegéllyel határolt táblázatot?
2. Az előző kérdést folytatva: hogyan választhatjuk el a táblázat celláit 30 képpontos térközzel a szegélytől?
3. Hogyan festhetjük az előző két kérdés táblázatának bal felső celláját zöldre, a jobb felsőt pirosra, a bal alsót sárgára, a jobb alsót pedig kékre?

Válaszok

1. Használjuk az alábbi HTML-kódot:

```
<table border="1">
  <tr>
    <td>Top left...</td>
    <td>Top right...</td>
  </tr>
  <tr>
    <td>Bottom left...</td>
    <td>Bottom left...</td>
  </tr>
</table>
```

2. A `<table>` elemet kell a `cellspacing="30"` jellemzővel bővítenünk.
3. A bal felső `<td>` elembe írjuk be, hogy `style="background-color:green"`, a jobb felsőbe azt, hogy `style="background-color:red"`, a bal alsóba, hogy `style="background-color:yellow"`, végül a jobb alsóba azt, hogy `style="background-color:blue"`.

Gyakorlatok

- Van a webhelyünknek olyan oldala, ahol a látogatók esetleg szívesen olvasnák lista, illetve táblázat formájában az információkat? Hozzunk létre egy táblázatot a sorokba rendezhető információk megjelenítésére. Minden oszlopnak legyen saját címsora – vagy akár saját grafikája. A mai órán megismert lehetőségek közül kísérletezzünk az igazítások különféle típusaival, illetve a térközökkel.
- Gyakran találkozunk váltakozó színű sorokból álló táblázatokkal, ahol az egyik sor mondjuk szürke, a következő pedig fehér háttérű. Egy táblázat sorait azért lehet érdemes váltakozó színűvé tenni, mert így első pillantásra is könnyebben azonosítja a szemünk az egyes sorokat. Készítsünk egy váltakozó színű sorokból álló táblázatot – szükség esetén a szöveg színe is váltakozzon. Bár a színekről szóló 9. óra még előttünk áll, már elegendő tudással rendelkezünk ahhoz, hogy bele merjünk fogni a feladatba.



8. ÓRA

Külső és belső hivatkozások használata

A lecke tartalma:

- Horgonyhivatkozások használata
- Hivatkozás a webhely egyik oldaláról egy másikra
- Hivatkozás külső tartalomra
- Hivatkozás elektronikus levélcímekre
- Ablakok megcélzása hivatkozásokkal
- A hivatkozások formázása CSS-stílusokkal

Az eddigiekben megtanultuk, hogyan készíthetünk a HTML-elemek segítségével egyszerű weboldalakat. Ebben a pillanatban azonban ezek a tartalomdarabkák szigetekként úsznak a semmiben, nem kapcsolódva egyéb oldalakhoz (megjegyeznénk, hogy ez sem teljesen igaz, hiszen a 4. órán becsempésztünk a kódba néhány oldalhivatkozást). Ahhoz, hogy munkánkból „valódi” webes tartalom váljék, valahogy össze kell kapcsolnunk az Internet többi részével – de legalábbis a saját vagy céges webhelyünk egyéb oldalaival.

Ezen az órán bemutatjuk, hogyan hozhatunk létre hivatkozásokat a saját dokumentumainkon belül, és hogyan hivatkozhatunk más dokumentumokra. Emellett megtanuljuk azt is, hogyan lássuk el stílusokkal a hivatkozásainkat – így az alapértelmezett kék színű, aláhúzott szöveg helyett nekünk tetsző megjelenést kaphatunk.

Webcímek használata

A webhelyünk tartalmának tárolására a legegyszerűbb módszer, ha minden fájlt egyetlen mappában gyűjtünk össze. Ha így áll a helyzet, az egyes fájlokra könnyen hivatkozhatunk, csak tüntessük fel a nevüket az `<a>` elem `href` jellemzőjében.



Mielőtt nekikezdenénk a munkának, érdemes felfrissítenünk arra vonatkozó ismereteinket, hogy hová helyezzük a fájljainkat a kiszolgálón, és miként kezeljük őket, ha több könyvtárat is alkalmazunk. Minderre nagy szükségünk lesz, amikor a webdokumentumainkban hivatkozásokat adunk meg. Lapozunk hát vissza a 2. órához, ott is *A fájlok helye a webkiszolgálón* című részhez.

Jellemzőnek (attribute) azokat az elemekhez mellékelte kiegészítő adatokat nevezzük, amelyek tovább árnyalják az elem feladatát. Így például az `<a>` elem `href` jellemzője annak az oldalnak a címét adja meg, amelyre hivatkozunk.

Ha egy maroknyinál több oldalunk van, illetve szeretnénk némi szerkezetet adni a webhelyünknek, érdemes könyvtárakba (vagy ha jobban tetszik, mappákba) rendezni azokat, amelyeknek a neve tükrözi a tartalmukat. Így például a képek egy `images` (képek) nevű könyvtárba kerülhetnek, a cég adatai az `about` (rólunk) nevezetűbe, és így tovább.

Akárhogy is rendezzük el a dokumentumainkat a webkiszolgálón, relatív címeket mindig használhatunk, amelyek éppen elegendőek ahhoz, hogy egy weboldaltól elérjünk egy másikat. A *relatív cím* (viszonyított cím) egy weboldaltól egy másikig vezető útvonalat ad meg, ellentétben a teljes (vagy *abszolút*) internetcímmel.

Ha visszaemlékszünk a 2. órán tanultakra, tudhatjuk, hogy a webkiszolgáló dokumentumgyökere a webes tartalom legfelső szintű könyvtárát jelenti. A webcímekben ezt a könyvtárat a perjel (`/`) azonosítja, és ez a jel szolgál az alacsonyabb szintű könyvtárak elválasztására is az útvonalon, valahogy így:

```
/könyvtár/alkönyvtár/al-alkönyvtár
```



A HTML-kódban mindig az egyszerű perjelet (`/`) használjuk a könyvtárak neveinek elválasztására, így feledkezzünk el a Windowsban megszokott fordított perjel (`\`) alkalmazásáról. Igaz tehát, hogy a Weben minden előre mutat – még a perjelek is!

Tételezzük fel, hogy egy `zoo.html` nevű oldalt hozunk létre a dokumentumgyökérben, és a kódjában el szeretnénk helyezni egy-egy hivatkozást az `elephants` alkönyvtár `african.html` és `asian.html` nevű fájljára. A hivatkozások kódja így fest:

```
<a href="/elephants/african.html">Learn about African elephants.</a>
<a href="/elephants/asian.html">Learn about Asian elephants.</a>
```

Az itt szereplő címek a gyökérhez viszonyított címek, ugyanis nem szerepel bennük a teljes tartománynév, ugyanakkor a viszonyítási alapjuk a perjellel megjelölt dokumentumgyökér.

A hagyományos relatív címek használatánál elhagyhatjuk a kezdő perjelet. Így a hivatkozások címét a böngésző az adott fájl könyvtárához képest értelmezi – ez lehet a dokumentumgyökér, de lehet ennek bármilyen szintű alkönyvtára is:

```
<a href="elephants/african.html">Learn about African elephants.</a>
<a href="elephants/asian.html">Learn about Asian elephants.</a>
```

Az `elephant` alkönyvtár `african.html` és `asian.html` fájljai az alábbi módon hivatkozhatnak viszont a `zoo.html` főoldalra:

```
<a href="http://www.yourdomain.com/zoo.html">Return to the zoo.</a>
<a href="/zoo.html">Return to the zoo.</a>
<a href="../zoo.html">Return to the zoo.</a>
```

Az első kódsor abszolút hivatkozást takar. Ez esetben *abszolút* semmi kételyünk nem lehet arra nézve, hogy hová mutat a hivatkozás, hiszen a teljes URL előttünk áll, beleértve a tartománynevet is. A második esetben azonban egy gyökérhez viszonyított hivatkozást látunk. Itt az útvonal az éppen meglátogatott tartományhoz viszonyítva értendő, így nincs szükség arra, hogy megadjuk a protokoll típusát (például `http://`), illetve a tartomány nevét (mint a `www.tartomany.com`) – a kezdeti perjel azonban ott van, jelezve, hogy a címet a dokumentumgyökérhez viszonyítva kell értelmeznünk.



A relatív (`elephants/african.html`) és az abszolút címezés

(`http://www.takeme2thezoo.com/elephants/african.html`) közötti döntésben a következő szabályhoz érdemes tartanunk magunkat: akkor használjunk relatív címeket, ha együtt tárolt fájlokra, például egy webhely összetevőire hivatkozunk, míg az abszolút címezést akkor kell igénybe vennünk, ha olyan fájlokra hivatkozunk, amelyek valahol másutt – egy másik számítógépen vagy lemezmeghajtón, illetve leggyakrabban az Internet egy másik helyén – találhatók.

A harmadik hivatkozásban *két pontot* (`..`) láthatunk – ez az adott könyvtárat tartalmazó könyvtárra, vagyis a szülőkönyvtárra utal. Ha tehát két ponttal találkozunk egy elérési úton belül, gondolatban mindig lépünk „egy szinttel feljebb”. Ha a relatív hivatkozásokat egységes módon használjuk az oldalainkon, a hivatkozásokhoz hozzá sem kell nyúlnunk, ha a webhelyet egy másik könyvtárba, lemezmeghajtóra vagy webkiszolgálóra költöztetjük.

A relatív címek szükség esetén meglehetősen bonyolult könyvtárszerkezeteket is átfoghatnak. A kiterjedt webhelyek oldalainak rendszerezéséről és összekapcsolásáról bővebben a 23. órán lesz szó.



Önálló feladat

Remélhetőleg már magunk is készítettünk pár weboldalt az eddigi feladatok során. Az alábbiakban továbbmegyünk – létrehozunk néhány újabb oldalt, és össze is kapcsoljuk azokat:

1. Határozzunk meg egy kezdőoldalt, amely a webhely bejárásának kiindulópontjaként működik majd – valamilyen formában az összes oldalunknak kapcsolódnia kell hozzá. Ha a korábbiakban már készítettünk magunkról, illetve a cégünkéről szóló oldalt, lehet ez a nyitólap, de újat is létrehozhatunk, ha úgy látjuk jónak.
2. Helyezzük el a kezdőoldalon az általunk korábban elkészített HTML-oldalak hivatkozásait (illetve helykitöltő sorokat azok számára, amelyeknek a létrehozását a későbbiekben tervezzük). Győződjünk meg arról, hogy a fájlnevek – beleértve a kis- és nagybetűk használatát – pontosan szerepelnek minden hivatkozásban.
3. A kezdőoldalon kívül minden más oldal tetején (vagy alján) helyezzünk el egy-egy hivatkozást, amely visszavisz a kezdőoldalra. Így a webhely bejárása jelentősen könnyebbé válik.
4. Érdemes lehet hivatkoznunk – akár a kezdőoldalról, akár egy külön ennek a célnak szentelt oldalról – a témánkhoz kapcsolódó, illetve érdekesnek tartott webhelyekre. Gyakran előfordul, hogy a webhelyek tulajdonosai a barátai webhelyeit is feltüntetik e hivatkozások sorában. A cégeknek azonban körültekintően kell eljárniuk ezekkel a hivatkozásokkal, nehogy túl gyorsan más webhelyekre irányítsák a reménybeli ügyfeleiket.

Oldalon belüli hivatkozások horgonyokkal

Az `<a>` *elem* (tag) – amely a webes hivatkozások megjelenítéséért felel – az „anchor” (horgony) szóból kapta a nevét, ami arra utal, hogy az oldal egy meghatározott helyét jelölhetjük meg vele. A könyvünkben eddig látott példákban mindössze annyit mutatunk be, hogy miként használhatjuk az `<a>` elemet arra, hogy valahová máshová vigyük a látogatót. Ezzel azonban még nem fedtük fel a benne rejlő lehetőségek teljes körét. Lássuk hát, miként alkalmazhatunk horgonyhivatkozásokat, amelyek egyazon oldal különböző részeit kapcsolják össze.

Oldalhelyek megjelölése horgonyokkal

Az `<a>` elem lehetővé teszi, hogy egy oldal pontjait horgonyként jelöljük meg, és így olyan hivatkozásokat hozunk létre, amelyek ezekre a pontokra mutatnak. A 8.1. példában, amelyet egy kicsit később mutatunk be, a gyakorlatban is láthatjuk mindezt. Ahhoz, hogy megértsük, hogyan is működnek ezek a hivatkozások, kukkantsunk bele a kódba, és vizsgáljuk meg a sorrendben első `<a>` elemet:

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

Amikor az `<a>` elemet hivatkozásként használjuk, a `href` jellemzővel adjuk meg a kívánt címet. Vagyis az `<a href>` elemre rákattinthatunk, míg az `<a id>` elem egy ilyen kattintás célállomásaként szerepel. Példánkban az `<a>` elem meghatároz egy célpontot, de ezúttal nem jön létre hivatkozás – itt mindössze nevet adunk annak a pontnak, ahol ez a címke megjelenik. A záró `` címkére szükség van, mint ahogy arra is, hogy az `id` jellemzővel megadott név egyedi legyen, az elem belsejében azonban nem feltétlenül kell tartalmat elhelyeznünk.



A HTML régebbi változataiban az `id` helyett a `name` jellemző volt használatos. A HTML újabb változatai, valamint az XHTML lemondta a `name` használatáról, így itt az `id` jellemző szerepel.

Hivatkozás horgonyokra

A 8.1. példában egy webhelyet láthatunk különféle horgonyokkal egyetlen oldalon belül. Vegyük szemügyre az utolsó `<a>` elemet:

```
<a href="#top">Return to Index.</a>
```

A `#` (kettőskereszt) jel azt jelöli, hogy a `top` szó egy névvel ellátott horgonyra hivatkozik a dokumentumon belül, nem pedig egy másik oldalra. Ha a felhasználó a Return to Index (Vissza a tárgymutatóhoz) hivatkozásra kattint, a böngésző azt a részét jeleníti meg az oldalnak, amelyik az `` elemmel kezdődik.

8.1. példa Horgonyok beállítása az `<a>` elem `id` jellemzőjével

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Alphabetical Shakespeare</title>
  </head>
```



```

<body>
  <h1><a id="top"></a>First Lines of Shakespearean Sonnets</h1>
  <p>Don't you just hate when you go a-courting, and you're down
    on one knee about to rattle off a totally romantic Shakespearean
    sonnet, and zap! You space it. <em>"Um... It was, uh... I think it
    started with a B..."</em></p>
  <p>Well, appearest thou no longer the dork. Simply refer to this page,
    click on the first letter of the sonnet you want, and get an instant
    reminder of the first line to get you started. <em>"Beshrew that
    heart that makes my heart to groan..."</em></p>
  <h2 style="text-align:center">Alphabetical Index</h2>
  <h3 style="text-align:center">
    <a href="#A">A</a> <a href="#B">B</a> <a href="#C">C</a>
    <a href="#D">D</a> <a href="#E">E</a> <a href="#F">F</a>
    <a href="#G">G</a> <a href="#H">H</a> <a href="#I">I</a>
    <a href="#J">J</a> <a href="#K">K</a> <a href="#L">L</a>
    <a href="#M">M</a> <a href="#N">N</a> <a href="#O">O</a>
    <a href="#P">P</a> <a href="#Q">Q</a> <a href="#R">R</a>
    <a href="#S">S</a> <a href="#T">T</a> <a href="#U">U</a>
    <a href="#V">V</a> <a href="#W">W</a> <a href="#X">X</a>
    <a href="#Y">Y</a> <a href="#Z">Z</a>
  </h3>
  <hr />
  <h3><a id="A"></a>A</h3>
  <ul>
    <li>A woman's face with nature's own hand painted,</li>
    <li>Accuse me thus, that I have scanted all, </li>
    <li>Against my love shall be as I am now</li>
    <li>Against that time (if ever that time come) </li>
    <li>Ah wherefore with infection should he live, </li>
    <li>Alack what poverty my muse brings forth, </li>
    <li>Alas 'tis true, I have gone here and there, </li>
    <li>As a decrepit father takes delight, </li>
    <li>As an unperfect actor on the stage, </li>
    <li>As fast as thou shalt wane so fast thou grow'st, </li>
  </ul>
  <p><a href="#top"><em>Return to Index.</em></a></p>
  <hr />
  <!-- continue with the alphabet -->
  <h3><a id="Z"></a>Z</h3>
  <p>(No sonnets start with Z.)</p>
  <p><a href="#top"><em>Return to Index.</em></a></p>
</body>
</html>

```



A 8.1. példa vége felé láthatunk egy sort az alábbi tartalommal:

```
<!-- continue with the alphabet -->
```

Ez a szöveg (amely egy HTML-megjegyzés) megjelenik ugyan a forráskódban, a böngészőablakban azonban nem. A megjegyzések használatáról bővebben a 23. órán lesz szó.

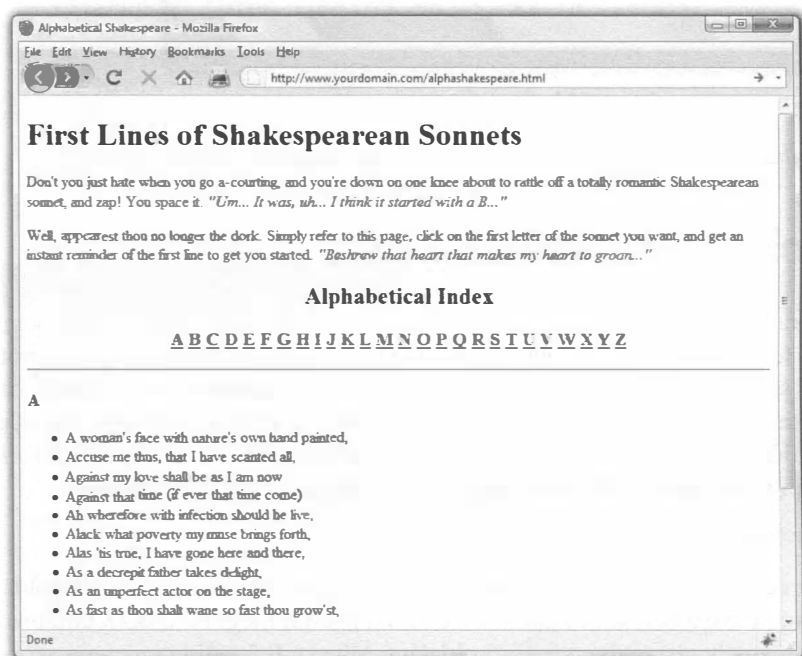


Az `<a>` elem `id` jellemzőjében megadott horgonyneveket alfanumerikus karakterrel kell kezdenünk, de ha egyszerűen csak meg szeretnénk számozni a horgonyok azonosítóit, akkor is írunk a sorszámok elé némi szöveget (például `photo1`, `photo2` stb.), mert a böngészők elfogadják ugyan a csak számból álló azonosítókat, de ez az XHTML-szabvány szerint már nem érvényes kód.

A 8.1. példa minden `<a href>` eleme egy-egy aláhúzott hivatkozást ad a megfelelő `<a id>` horgonyhoz – azaz adna, ha mindegyik horgonyt meghatároztuk volna.

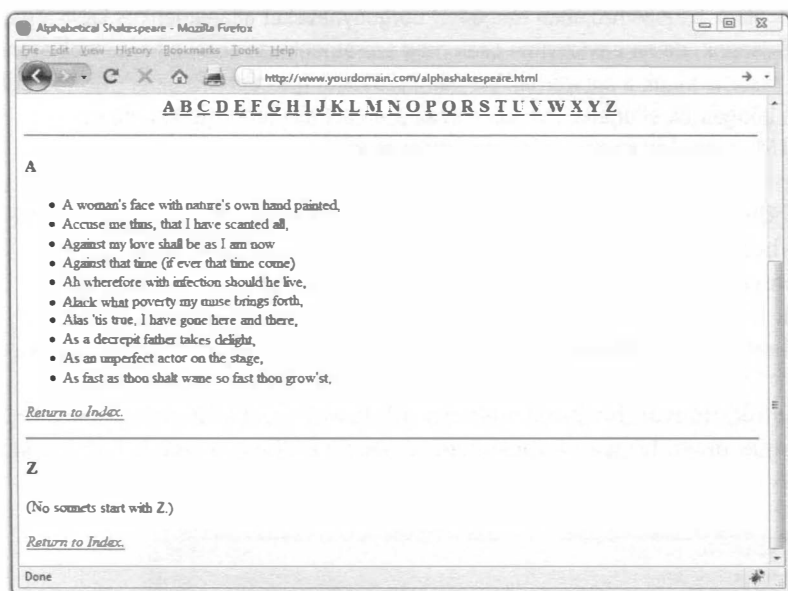
A példában ezért csak az A és a Z hivatkozás működik, de ha úgy tartja kedvünk, a többi magunk is kitölthetjük. Ha a 8.1. ábra Alphabetical Index (Betűrendes tárgymutató) felirata alatt a Z betűre kattintunk, a 8.2. ábrán látható oldalra jutunk.

Miután megtanultuk, hogyan hivatkozzunk egy adott weboldal különböző részeire, most azt is láthatjuk majd, hogyan kapcsolhatunk össze különálló webes tartalmakat.



8.1. ábra

A 8.1. példában látható `<a id>` elemek nem jelennek meg a képernyőn, míg az `<a href>` elemeket aláhúzott hivatkozásként látjuk



8.2. ábra

A 8.1. ábrán látható oldalon a Z betűre kattintva az oldal megfelelő részéhez jutunk

Hivatkozások a saját oldalaink között

Amint azt az óra korábbi részében megtudhattuk, a href jellemző megadásakor nem kell feltétlenül feltüntetnünk a `http://` karakterláncot a címben, amennyiben a hivatkozott tartalom a saját tartományunkon belül található (illetve ugyanazon a számítógépen, ha helyben tekintjük meg). Továbbá, ha egyazon tartomány vagy számítógép két fájlja között hozunk létre hivatkozást, a teljes webcím megadására sincs szükség. Sőt ha a két fájl ugyanabban a mappában található, elég csak a HTML-fájl nevét feltüntetni:

```
<a href="pagetwo.html">Go to Page 2.</a>
```

A 8.2. példában és a 8.3. ábrán egy fejtörő oldalt mutatjuk be, amelynek a megoldásait a 8.3. példában, illetve a 8.4. ábrán látható weboldal jeleníti meg. Ez utóbbi tartalmaz egy hivatkozást, amely visszavisz a fejtörő oldalára. Mivel a 8.2. példa egy vele azonos könyvtárban található fájlra hivatkozik, a teljes cím helyett használhatjuk magát a fájlnevet.

8.2. példa *A historyquiz.html fájl*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>History Quiz</title>
  </head>

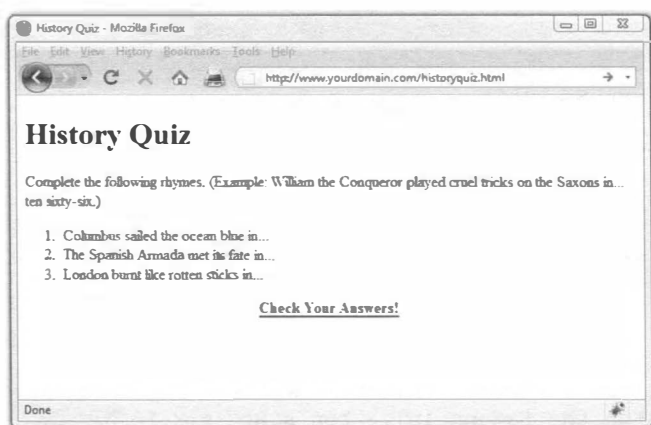
  <body>
    <h1>History Quiz</h1>
    <p>Complete the following rhymes. (Example: William the Conqueror
    Played cruel tricks on the Saxons in... ten sixty-six.)</p>
    <ol>
      <li>Columbus sailed the ocean blue in...</li>
      <li>The Spanish Armada met its fate in...</li>
      <li>London burnt like rotten sticks in...</li>
    </ol>
    <p style="text-align: center;font-weight: bold;">
      <a href="historyanswers.html">Check Your Answers!</a>
    </p>
  </body>
</html>
```

8.3. példa *A historyanswers.html oldal, amelyre a historyquiz.html hivatkozik*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

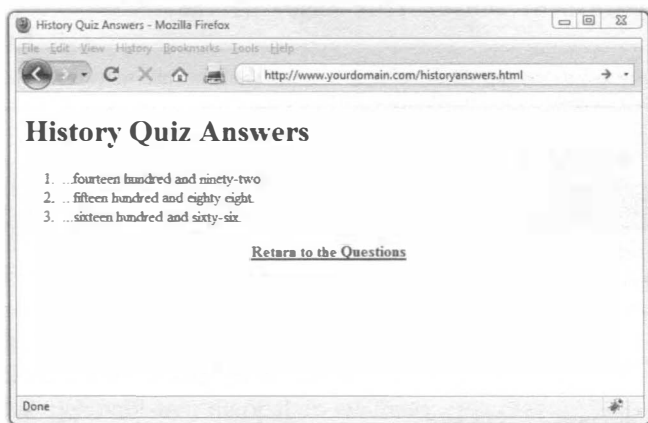
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>History Quiz Answers</title>
  </head>

  <body>
    <h1>History Quiz Answers</h1>
    <ol>
      <li>...fourteen hundred and ninety-two.</li>
      <li>...fifteen hundred and eighty eight.</li>
      <li>...sixteen hundred and sixty-six.</li>
    </ol>
    <p style="text-align: center;font-weight: bold;">
      <a href="historyquiz.html">Return to the Questions</a>
    </p>
  </body>
</html>
```



8.3. ábra

A *historyquiz.html*, amelynek a kódját a 8.2. példában láttuk, és amelyre a 8.3. példa hivatkozása utalt



8.4. ábra

A 8.3. ábrán látható *Check Your Answers!* hivatkozás erre a válaszokat tartalmazó oldalra vezet, a *Return to the Questions* hivatkozás pedig visszavisz a 8.3. ábra oldalára

Azzal, hogy teljes webcímek helyett fájlneveket használunk, rengeteg időt takaríthatunk meg. Ennél is fontosabb azonban, hogy az oldalaink közötti hivatkozások mindig működőképesek maradnak, bárhová is helyezzük át a webhelyünket. A hivatkozásokat így ellenőrizhetjük még a számítógépünk merevlemezén, majd ezután átmásolhatjuk a fájlokat egy webkiszolgálóra, CD-re, DVD-re vagy memóriakártyára, miközben nem kell aggódnunk amiatt, hogy a hivatkozások esetleg megszakadnak. Nincs ebben semmi varázslat – minden a weboldalak címeinek megfelelő használatán múlik.

Hivatkozás külső webes tartalomra

A saját webhelyünkön belüli oldalakra és a külső webes tartalomra mutató hivatkozások között mindössze annyi a különbség, hogy az utóbbi esetben a teljes címet fel kell tüntetnünk, beleértve a `http://` előtagot, a tartománynevet, valamint a fájl teljes elérési útját (legyen szó HTML-, kép-, illetve multimédiafájlról, vagy bármi másról).

Így ha valamelyik oldalunkról a Google keresőjére szeretnénk hivatkozni, ilyesfajta abszolút címzést kell használnunk az `<a>` elemben:

```
<a href="http://www.google.com/">Go to Google</a>
```



Ismeretes, hogy a legtöbb böngészőben nem okoz gondot, ha elhagyjuk a webcím elejéről a `http://` előtagot. Ez a lazaság azonban a HTML-kódban nem elfogadható – az `<a href>` hivatkozásokban a teljes alakot kell megadnunk.

Mindemellett az előzőekben tanultakat alkalmazva más oldalak névvel ellátott horgonyaira is hivatkozhatunk – ehhez nem kell feltétlenül ugyanazon az oldalon lennünk. Mindössze annyit kell tennünk, hogy a cím, illetve a fájlnev után egy `#` jelet követően feltüntetjük a horgony nevét. A következő hivatkozás például egy `photos` nevű horgonyhoz visz az `african.html` nevezetű oldalon, amely a `www.takeme2thezoo.com` tartomány `elephants` könyvtárában található:

```
<a href="http://www.takemetothetoo.com/elephants/african.html#photos">
Check out the African Elephant Photos!</a>
```

Ha egy olyan oldalon helyezzük el ezt a hivatkozást, amely maga is a `www.takeme2thezoo.com` tartományban található (hiszen mi vagyunk a webhely fenntartói), a hivatkozás ilyen egyszerű alakot ölt:

```
<a href="/elephants/african.html#photos">Check out the African Elephant
Photos!</a>
```

Amint azt a korábbiakban megtanultuk, a `http://` előtag és a tartománynév ilyen esetben elhagyható.



Fontos, hogy a kettőskereszt jelet csak az `<a href>` elemekben használjuk – ha az `<a id>` elemekbe kerülnek, a hivatkozások elromlanak.

Hivatkozás elektronikus levélcímre

Amellett, hogy weboldalakra és azok részeire hivatkozhatunk, az `<a>` elem lehetővé teszi azt is, hogy egy elektronikus levélcímet adjunk meg a `href` jellemzőben, ami egyszerű módot ad a látogatói visszajelzésre. Természetesen magunk is feltüntethetjük a levélcímet valahol, arra számítva, hogy a lelkesebb látogatók majd begépelik, de ez növeli a hibák lehetőségét. Ha ugyanakkor egyetlen kattintással lehetővé tesszük a levélküldést, még ettől a fáradságtól is megkíméljük a felhasználókat, és teljesen kiküszöböljük az elgépelés eshetőségét.

Egy elektronikus levélcímre mutató HTML-hivatkozás valahogy így fest:

```
<a href="mailto:yourusername@yourdomain.com">Send me an email message.</a>
```

A Send me an email message (Küldjön nekem elektronikus levelet) szöveg pontosan olyan alakban jelenik meg, mint más `<a>` hivatkozások.

Ha azt szeretnénk, hogy a látogatóink számára láthatóvá is váljon az elektronikus levélcímünk (így feljegyezhetik, vagy az alapértelmezettől eltérő levelezőprogramban is felhasználhatják), a `href` jellemző mellett tüntessük fel az `<a>` és az `` címke közötti szövegben is:

```
<a href="mailto:yourusername@yourdomain.com">yourusername@yourdomain.com</a>
```

Ha a felhasználó egy ilyen hivatkozásra kattint, a legtöbb böngészőben egy ablakot kap, amelyben nyomban megírhatja a nekünk szánt üzenetét – ehhez a rendszer automatikusan azt a levelezőprogramot nyitja meg, amelyet az illető használ. A hivatkozásban azonban mi is elhelyezhetünk előre megírt tartalmat mind a levél tárgyában, mind pedig a törzsében – ehhez csak használatba kell vennünk a `mailto` hivatkozásban a `subject` és a `body` változókat. A változókat a levélcímtől egy kérdőjellel (?) választhatjuk el, az egyes változókat az értéküktől egy egyenlőségjellel (=), végül pedig a változó-érték párokat az „és” jel (&) határolja. A „változó” és az „érték” fogalmát jelenleg nem szükséges megértenünk, elég, ha látjuk, hogyan helyezhetünk el alapértelmezett tárgysort és törzsszöveget az előbbi példa hivatkozásában:

```
<a href="mailto:author@somedomain.com?subject=Book Question&body=When is the next edition coming out?">author@somedomain.com</a>
```

Ha a felhasználó erre a hivatkozásra kattint, a böngésző létrehoz egy elektronikus levelet, amelynek a címzettje az `author@somedomain.com` (*szerző@törtélem.com*) a tárgyában a Book Question (Kérdés a könyvvel kapcsolatban), a törzsében pedig a When is the next edition coming out? (Mikor jelenik meg az új kiadás?) szöveg áll.



Ha a levélben alapértelmezés szerint csak a tárgyat szeretnénk megadni, a törzset nem, egyszerűen hagyjuk ki az `& jelet`, a `body` változót, az egyenlőségjelet és a megadott szöveget, valahogy így:

```
<a href="mailto:author@somedomain.com?subject=
BookQuestion">author@somedomain.com</a>
```

Mielőtt kitapétáznánk a weboldalainkat az e-mail címünkkel, álljon itt egy halk figyelmeztetés és egy apró fogás. Bizonyára nem ismeretlen előttünk a levélszemét (spam) fogalma, és talán az sem, hogy akik ezt a nyavalyát a felhasználók nyakába zúditják, komoly címadatbázisokat építenek fel erre a célra, az egyik gyűjtési módszerük pedig éppen a weboldalak `mailto` hivatkozásainak „learatása”.

Szerencsére van egy apró fogás, amellyel ezeknek a támadásoknak a nagy többségét kivédhetjük. A módszer alapja, hogy a karakterek helyett karakterkódokat (karakter-egyedeket, `character entity`) használunk a címekben, amelyek becsapják a gyűjtögető programokat. Vegyük például a `jcmeloni@gmail.com` címet. Ha a betűket a karakterkód-megfelelőikre cseréljük, a legtöbb levélcímgyűjtő programot lerázhatjuk. Az ASCII-kisbetűk kódjai `a` -nél kezdődnek (ez az „a” betű kódja), és ábécésorrendben emelkednek. Így a „j” betű kódja `j`, a „c” betűé `c`, és így tovább. Ha az összes karaktert az ASCII-megfelelőjére cseréljük, ezt kapjuk:

```
<a
href="mailto:&#106;&#099;&#109;&#101;&#108;&#111;&#110;&#105;&#064;&#103;
&#109;&#097;&#105;&#108;&#046;&#099;&#111;&#109;">Send me an email
message.</a>
```

Mivel a böngésző a karakterkódokat karakterként értelmezi, a végeredmény a megjelenítés szempontjából az eredetivel megegyező lesz. Az automatikus levélcímgyűjtő programok ugyanakkor az oldal HTML-kódját nézik, ami ez esetben meglehetősen kusza címet ad. Ha nem akarunk magunk bajlódni a webcímünk átalakításával, írjuk be egy keresőbe az „email address encoder” (levélcímkódoló) kifejezést, és biztosan találunk néhány, az Interneten elérhető szolgáltatást, amely elvégzi ezt helyettünk.



Bevett szokás a szerző e-mail címét minden oldal alján feltüntetni. Ezzel egyrészt megkönnyítjük a látogatóknak, hogy kapcsolatba lépjenek velünk, másrészt így az oldalak esetleges hibáiról is nagyobb eséllyel kapunk visszajelzést. Csak a karakterkódos fogás alkalmazására kell ügyelnünk, hiszen így kívül maradhatunk a címgyűjtögetők látókörén.

Hivatkozás megnyitása új böngészőablakban

Megtanultuk hát, hogyan adjunk meg hivatkozásokat – legyen szó akár a webhelyünkön belülre, akár kívülre mutató hivatkozásokról –, de kihagytunk még egy hivatkozásfajtát, amely lehetővé teszi, hogy a kívánt weboldalt egy új ablakban nyissuk meg.

Bizonyára hallottunk már az *előugró ablakokról* (pop-up window), amelyek többnyire valamilyen reklámot jelenítenek meg teljesen önműködően, a felhasználó hozzájárulása nélkül. Vannak azonban helyzetek, amikor egy másik ablak megnyitása teljességgel indokolható. Ilyen lehet, ha egy kisebb, másodrendű böngészőablakban szeretnénk kiegészítő tájékoztatást adni úgy, hogy közben a felhasználó továbbra is láthassa az eredeti ablakot. Ez gyakran előfordul, ha egy animált bemutatóra, egy klipre vagy más multimédiás elemre kattintunk. Új ablak megnyitására akkor is szükség lehet, ha a webhelyünkön kívüli tartalmat szeretnénk megjeleníteni.

Az új böngészőablak megnyitása – különösen ha teljes képernyős ablakról van szó – még akkor is szembemegy bizonyos használhatósági és hozzáférhetőségi irányelvekkel, ha ezt a felhasználó tudtával tesszük. Korábban az új ablakok megnyitása többnyire az `<a>` elem `target` jellemzőjével történt – ez azonban már nem szerepel a szigorú XHTML 1.1 szabványban.

Ugyanezt az eredményt az irányelvek követése mellett is elérhetjük, de ehhez szükség van némi JavaScript-kódra és más fejlettebb programozástechnikai módszerekre. Ezekről a 18. órán lesz szó, ahol megtanuljuk, hogyan jelenítsünk meg ablakokat a külső hivatkozásaink számára úgy, hogy a szabványoknak és a hozzáférhetőségi követelményeknek is megfeleljünk.

Hivatkozások formázása CSS-stílusokkal

Az oldalaink szöveges hivatkozásai alapértelmezés szerint kék betűkkel, aláhúzva jelennek meg. Talán észrevettük azt is, hogy a már meglátogatott hivatkozások színe lilára változik – ez is alapértelmezés. Ha azonban akár egy kis időt is eltöltöttünk az Interneten, láthattuk, hogy nem minden hivatkozás kék és lila – és ezért, úgy vélem, mindannyian hálásak lehetünk. Szerencsére némi CSS-kód és az `<a>` elem álosztályainak ismeretében mi is kiléphetünk az alapértelmezések unalmas világából.



Szöveg helyett képeket is használhatunk hivatkozásként, csak helyezzük el a megfelelő `` elemet a nyitó `<a>` és a záró `` címke között. A hivatkozásként használt képekről bővebben a 11. órán lesz szó.

Az *álosztály* olyan „osztály”, amely az elemek stílusát bizonyos körülmények között írja le, amelyek megfelelhetnek például a felhasználó műveleteinek. Így például az `<a>` elem ismert álosztályai a `link`, a `visited`, a `hover` és az `active`.

- Az `a:link` azt az állapotot írja le, amelyben a hivatkozást akkor látjuk, ha még nem kattintottunk rá.
- Az `a:visited` arra az állapotra utal, amikor a hivatkozást már meglátogatták, és benne van a böngésző memóriájában.
- Az `a:hover` arra az állapotra utal, amikor a felhasználó éppen a hivatkozás felett áll az egérmutatóval (de még nem kattintott rá).
- Az `a:active` azt határozza meg, hogy miként kell megjeleníteni a hivatkozást, ha a felhasználó rákattintott a hivatkozásra, de még nem engedte fel az egérgombot.

Tegyük fel, hogy az alábbi stílusokkal szeretnénk egy hivatkozást létrehozni:

- Félkövér, Verdana betűtípussal (aláhúzás, vagyis „díszítés” nélkül)
- Világoskék alapszínnel
- A hivatkozás színe változzon vörösre, ha a felhasználó fölé viszi az egérmutatót, vagy rákattint
- A meglátogatott hivatkozás betűszíne legyen szürke

Mindennek a következő stíluslap-bejegyzések felelnek meg:

```
a {
    font-family: Verdana, sans-serif;
    font-weight: bold;
    text-decoration: none;
}
a:link {
    color: #6479A0;
}
a:visited {
    color: #CCCCCC;
}
a:hover {
    color: #E03A3E;
}
a:active {
    color: #E03A3E;
}
```

A színeket a példában tizenhatos számrendszerű (hexadecimális) értékekkel adtuk meg – erről bővebben a 9. órán lesz szó.



Mivel a példában szereplő hivatkozás az állapotától függetlenül félkövér (és nem aláhúzott) Verdana betűkkel jelenik meg, ezt a három tulajdonságot feltüntethetjük az a választó bejegyzésében. Az álosztályok azonban a betűszín tekintetében eltérnek egymástól, ezért ezt a szabályt külön-külön fel kell tüntetnünk mindegyikükénél. Az álosztály örökli a szülője szabályait – hacsak felül nem írja azokat. Következésképpen a példa összes álosztályában félkövér, nem aláhúzott Verdana betűtípus szerepel. Ha azonban az alábbi szabályt alkalmaztuk volna a `hover` álosztálynál, a szöveg Comic Sans betűtípussal jelenne meg, amikor a felhasználó fölé viszi az egérmutatót (persze csak akkor, ha a betűtípus elérhető a számítógépen):

```
a:hover {
    font-family: "Comic Sans MS";
    color: #E03A3E;
}
```

Emellett, mivel az `active` és a `hover` álosztályok ugyanazt a betűszínt alkalmazzák, egyesíthetjük a stíluslap-bejegyzéseiket:

```
a:hover, a:active {
    color: #E03A3E;
}
```

A 8.4. példában összerakjuk az eddigi kódrészleteket – az így kapott oldalon, amelyet a 8.5. ábrán láthatunk, az álosztályok az általunk választott stílusokkal jelennek meg.

8.4. példa *Hivatkozások álosztályainak formázása stílusokkal*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Link Style</title>

    <style type="text/css">
      a {
        font-family: Verdana, sans-serif;
        font-weight: bold;
        text-decoration: none;
      }
      a:link {
        color: #6479A0;
      }
      a:visited {
        color: #CCCCCC;
      }
      a:hover, a:active {
        color: #FF0000;
      }
    </style>
  </head>
  <body>
    <a href="#">Sample Link</a>
  </body>
</html>
```

```

</style>
</head>
<body>
  <h1>Sample Link Style</h1>
  <p><a href="simplelinkstyle.html">The first time you see me, I
    should be a light blue, bold, non-underlined link in the Verdana
    font</a>.</p>
</body>
</html>

```



8.5. ábra

A hivatkozások megjelenését stílusokkal szabályozhatjuk

Ha megtekintjük a kapott oldalt egy böngészőben, láthatjuk, hogy a hivatkozás valóban világoskék, félkövér, nem aláhúzott Verdana betűkkel jelenik meg. Ha fölé visszük az egérmutatót, majd rákattintunk, és nem engedjük fel az egérgombot, a szöveg színe vörösre vált. Végül, ha a kattintás után felengedjük az egérgombot, a böngésző egyszerre újra betölti az oldalt, hiszen a hivatkozás ugyanerre a fájlra mutat. A cím azonban bekerül a böngésző memóriájába, így már látogatott hivatkozásként jelenik meg – a kék helyett szürke betűkkel.

A CSS segítségével széles körben módosíthatjuk a hivatkozásaink szövegtulajdonságait. Megváltoztathatjuk a betűtípust, a betűméretet, a betűvastagságot, a díszítést, és így tovább. Az is előfordulhat, hogy többféle hivatkozásstílust is szeretnénk használni ugyanazon az oldalon belül. Ilyenkor osztályokat hozhatunk létre – nem kell magunkat az `<a>` elem egyetlen stíluskészletére korlátoznunk. Az alábbiakban egy `footerlink` nevű osztály stílusait mutatjuk be, amelyek a lábléc hivatkozásainak csinosítására születtek:

```

a.footerlink {
  font-family: Verdana, sans-serif;
  font-weight: bold;
  font-size: 75%;
  text-decoration: none;
}

```

```
a.footerlink:link, a.footerlink:visited {
    color: #6479A0;
}
a.footerlink:hover, a.footerlink:active {
    color: #E03A3E;
}
```

Amint a példában is láthatjuk, ilyenkor a választó neve (a) után egy pont áll, ezt követi az osztály neve (footerlink), egy kettőspont, végül pedig az álosztály neve:

```
választó.osztály:álosztály
a.footerlink:hover
```

Tekintsük át a B függelékét, így képet kaphatunk arról, hogy milyen stílusokat is alkalmazhatunk a hivatkozásokra.

Összefoglalás

Az <a> elem teszi a hiperszöveget „hiperré” – a segítségével teremtünk kapcsolatot más weboldallal vagy az adott oldal horgonyaival. Ezen az órán megtanultuk, hogyan hozhatunk létre egyszerű hivatkozásokat más oldalakra a relatív vagy abszolút címük megadásával.

Ha a webhelyünkön kívüli oldalakra hivatkozunk, fontos, hogy a kívánt oldal teljes címét feltüntessük az <a href> elemben. Ha ugyanakkor a saját oldalaink között teremtünk kapcsolatot, elég a fájlnev és a megfelelő elérési út feltüntetése.

Megtanultuk azt is, hogyan határozhatunk meg névvel ellátott horgonyokat egy weboldal szerkezetében, és hogyan hivatkozhatunk rájuk. Megtudtuk, hogy a hivatkozásokban elektronikus levélcímet is elhelyezhetünk, amivel megkönnyíthetjük, hogy a látogatóink visszajelzést küldjenek nekünk. Kitértünk arra is, miként rejthetjük el az e-mail címünket a levélszemét-küldők elől, az óra végén pedig szót ejtettünk arról, hogyan határozhatjuk meg a hivatkozások megjelenését CSS-stílusok segítségével.

Az <a> elemről tanultakat a 8.1. táblázatban foglaljuk össze.

8.1. táblázat A 8. órán megismert HTML-elemek és jellemzőik

Elem/jellemző	Leírás
<code><a>...</code>	A href jellemzővel alkalmazva hivatkozást kapunk a megadott weboldalra vagy horgonyra, míg az id jellemzővel egy horgonyt kapunk, amelyre hivatkozhatunk.
Jellemzők	
href="cím"	A hivatkozott dokumentum vagy horgony címe.
id="név"	A dokumentumban található horgony neve (azonosítója).

Kérdezz-felelek

- K: *Mi történik, ha egy olyan weboldalra hivatkozunk valahol az Interneten, amelyet a gazdája később töröl vagy áthelyez?*
- V: Nos, ez attól függ, hogyan állította be az illető a webkiszolgálóját. Többnyire a „Page not found” (Az oldal nem található) üzenetet vagy valami hasonlót kapunk. Persze ilyenkor sincs komolyabb gond, hiszen a böngésző Vissza gombjával visszajuthatunk a kiindulási oldalra. Webhelyünk fenntartójaként komoly felelősségünk van abban, hogy rendszeresen futtassunk valamilyen hivatkozásellenőrző programot, hogy meggyőződjünk róla, hogy mind a belső, mind a külső hivatkozásaink rendben vannak. Ilyen szolgáltatás például a <http://validator.w3.org/checklink> címen található Link Checker.
- K: *A webhelyem egyik belső hivatkozása az otthoni számítógépemen tökéletesen működik, de ha felteszem az oldalakat az Internetre, hibát észlelek. Mi történhetett?*
- V: A leggyakrabban a következő problémák valamelyike áll a háttérben:
- *A kis- és nagybetűk hibás használata.* Windows rendszerű gépeken, ha a MyFile.html fájlra az `` kóddal hivatkozunk, semmilyen probléma nem jelentkezik, a legtöbb webkiszolgálón azonban a hivatkozást az `` alakban kell megadnunk (de módosíthatjuk a MyFile.html nevét is). A problémát súlyosbítja, hogy egyes szövegszerkesztők és fájltviteli programok a megkérdezésünk nélkül módosítják a kis- és nagybetűket. A legjobb, ha megmaradunk a csupa kisbetűs fájlneveknél.
 - *Szóközők a fájlnevekben.* A legtöbb webkiszolgáló nem engedi meg a szóközőket tartalmazó fájlnevek használatát. Soha ne adjuk egy oldalnak a my page.html nevet – használjuk helyette a mypage.html vagy a my_page.html változatot (utóbbi esetben aláhúzást alkalmazva a szóköző helyett).

- *Helyi abszolút címek.* Ha valamilyen rejtélyes okból helyi abszolút címmel (például: C:\mywebsite\news.html) hivatkozunk egy fájlra, a hivatkozás az Interneten nem fog működni. A helyi abszolút címek használata minden körülmények között ellenjavallt; a megjelenésük többnyire annak tudható be, hogy egy az oldal tesztelése közben használt átmeneti hivatkozás véletlenül bent maradt a kódban. Ügyeljünk tehát arra, hogy az ilyen hivatkozásokat kigyomláljuk, mielőtt feltöltenénk a webhelyünket az Internetre.

K: *Alkalmazhatjuk egyetlen hivatkozáson belül a href és az id jellemzőt?*

Ha igen, miért tennénk?

V: Igen, és ezzel némi gépeléstől kímélhetjük meg magunkat, ha amúgy is egymás mellett állna egy hivatkozás és egy horgony. Általában azonban a zavar elkerülése végett érdemes az `<a href>` és az `<a id>` elemeket külön kezelni, hiszen igen eltérő szerepet játszanak a HTML-dokumentumokban.

K: *Mi történik, ha véletlenül elírjuk egy horgony nevét, vagy elfelejtjük kitenni elé a # jelet?*

V: Ha olyan horgonyra hivatkozunk, amely nincs jelen az oldalon, illetve elírjuk a nevét, a hivatkozás az oldal tetejére visz.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Általános iskolai legjobb barátunk ráakadt a webhelyünkre az Interneten, és szeretne velünk hivatkozást cserélni. Hogyan helyeznénk el valamelyik oldalunkon hivatkozást a `www.supercheapsuits.com/~billybob/` címen található webhelyére?
2. Milyen HTML-kóddal érhetnénk el, hogy ha egy látogatónk a lap tetején az „About the Authors” (A szerzőkről) szavakra kattint, átugorjon a szerzők bemutatására, amely az oldal egy másik részén található?
3. Ha az e-mail címünk a `bon@soir.com`, hogy alakíthatjuk át a „goodnight greeting” szöveget hivatkozássá, amelyre kattintva a látogatóink üzenetet küldhetnek nekünk?

Válaszok

1. Helyezzük el az alábbi kódot az oldalon:

```
<a href="http://www.supercheapsuits.com/~billybob/">Billy Bob's
site</a>
```

2. Írjuk a következő kódsort az oldal tetejére:

```
<a href="#credits">About the Authors</a>
```

Ezt pedig a szerzők bemutatása elé:

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

3. Helyezzük el ezt a kódot az oldalunkon:

```
Send me a <a href="mailto:bon@soir.com">goodnight greeting</a>!
```

Gyakorlatok

- Készítsünk HTML-oldalt, amely a kedvenc webhelyeink formázott listáját tartalmazza. Ha egyesek megvannak a könyvjelzőink között is, látogassuk meg őket, és másoljuk ki a címüket a böngésző címmezőjéből.
- Ha korábban készítettünk már oldalakat egy webhely számára, most nézzük át őket újra, és keressük meg azokat a helyeket, ahol érdemes lenne megadnunk a lehetőséget a látogatóknak, hogy kapcsolatba lépjenek velünk. Tüntessük fel ezeken a helyeken az elektronikus levélcímünket. Ne feledjük, a kapcsolatfelvétel és a visszajelzés lehetőségeiből sosem elég – különösen ha üzleti vállalkozást működtetünk.



9. ÓRA

A színek használata

A lecke tartalma:

- Hogyan válasszunk színeket a webhelyünkhöz?
- A színek viselkedése a Weben
- Hexadecimális színkódok használata
- A háttér, a szöveg és a szegélyek színének beállítása CSS-stílusok segítségével

Az eddigiekben bemutatott weboldalak mindegyikében fehér háttérrel és fekete betűket láttunk. Ez persze nem alapkövetelmény, annyit azonban biztos megállapíthatunk, hogy az Interneten a leggyakrabban olyan oldalakat találunk, amelyek világos háttéren sötét színnel jelenítik meg a szöveget. A színválasztás fogásainak bemutatása után megtanuljuk, mikor használhatunk színeket, miként válasszuk ki őket, és hogyan adjuk meg egy oldal különböző elemeinek színét.

A színválasztás fogásai

Arra semmiképpen sem vállalkozhatunk, hogy megmondjuk, melyik webhelyhez milyen színek illenek, abban azonban segíthetünk, hogy milyen elveket tartsunk szem előtt a választásnál. A használt színek komoly hatással lehetnek a látogatóinkra, így ha például egy e-kereskedelmi oldalt üzemeltetünk, érdemes olyan színösszeállítást alkalmazni, amely valósággal csábítja a látogatókat a termékínálatunk megtekintésére és így a vásárlásra is. Fontos, hogy a színeket körültekintően és kellő empátiával használjuk. Hogy jön ide az empátia? Nos, ne feledjük, hogy a Világháló nemzetközi közösség, amelyben a színek értelmezése változó lehet. Japánban például nagyon népszerű a rózsaszín, ugyanakkor Kelet-Európában komoly elutasítással viszonyulnak hozzá. Ehhez hasonlóan lehet, hogy Amerikában egyértelműen „zöldhasúak” a bankók, de a legtöbb ország bankjegyei a szívránc minden színében pompáznak, így ott a „pénz színe” kifejezés értelmetlen.

A kulturálisan érzékeny színválasztás mellett érdemes az alábbi tanácsokat is megfogadnunk:

- *Használjunk természetes színválasztéket.* Ez nem azt jelenti, hogy a földszínekhez kell ragaszkodnunk, mindössze arra utal, hogy célszerű olyan színeket alkalmaznunk, amelyekkel az ember séta közben is találkozik – tartózkodjunk a rikító színektől, amelyek fárasztják a szemet.
- *Alkalmazzunk szűkebb színskészletet.* Nincs szükség 15 különböző színre ahhoz, hogy elérjük, amit szeretnénk. Sőt ha a weboldalunk szövege és ábrái 15 különböző színben jelennek meg, érdemes lehet újra átgondolni, hogy mit is szeretnénk valójában üzeni a látogatóknak. Összpontosítsunk három-négy főbb színre és néhány kiegészítő árnyalatra – többet semmiképp se használjunk.
- *Vegyük figyelembe a látogatók koreloszlását.* Mivel ezt magunk nem tudjuk szabályozni, jobb, ha egy mindenki számára elfogadható középútat keresünk. A fiatalok körében népszerű élénk színek lehet, hogy az idősebbek között már nem aratnak akkora tetszést, de hasonló eltérések lehetnek a férfiak és nők, valamint a különböző kultúrákhoz tartozók ízlése között.

Úgy érezhetjük, hogy a fenti útmutatások követése mellett már nem sok választási lehetőségünk marad. Ez azonban egyáltalán nincs így – itt egyszerűen csak arra hívtuk fel a figyelmet, hogy kicsit gondolkodjunk el, mielőtt döntéseket hoznánk. Ha pedig rákérünk az Interneten a „color theory” (színelmélet) kifejezésre, a gondolataink újabb lendületet kaphatnak, akár csak a színkerék megismerése után.

A *színkerék* egy diagram, amely a színeket kör alakban elhelyezve mutatja be. Ez az ábrázolásmód alkalmas arra, hogy élénk tárja az alapszíneket, valamint a kevert és kiegészítő színeket. Színsémákat a színkerék használatával kaphatunk, ezekből pedig kikristályosodhat a webhelyünk palettája, amely biztosítja az egységes színhasz-

nálatot. Ha némi jártasságra teszünk szert a színek viszonyainak terén, talán könnyebben kerüljük el az olyan szörnyűséges színösszeállításokat, mint a narancsszínű felirat világoskék háttéren, vagy az élénkkék felirat barna háttér előtt.

A webtervezésben megszokottak az alábbi színsémák:

- **Analog** – Ilyenkor a színceréken egymás szomszédságában elhelyezkedő színeket használunk (például sárga és zöld). Az egyik szín domináns, míg az analóg társa a megjelenés gazdagítására hivatott.
- **Komplementer** – Ebben a sémában a színceréken egymással szemben található színeket alkalmazunk, például egy meleg (vörös) és egy hideg (zöld) színt.
- **Triadikus** – Három szín használatát jelenti, amelyek egyenlő távolságra vannak egymástól a színceréken. Ez a séma egyszerre ad kiegyensúlyozottságot és gazdag színekészletet.

Egész könyveket és tanfolyamokat szenteltek a színelméletnek, így komoly kitérőt jelentene, ha ebben a könyvben belebonyolódnánk a témába. Mindazonáltal, ha valóban webtervezéssel és -fejlesztéssel szeretnénk foglalkozni, komoly segítséget jelenthet, ha átlátjuk a színelmélet alapjait. Töltsünk hát némi időt a kutakodással – az internetes keresők sokat segíthetnek.

Gyakoroljuk emellett a színcerék használatát is.

A <http://colorscheme designer.com/> címen található Color Scheme Generator (Színséma-generátor) lehetővé teszi, hogy egy alapként megadott színből kiindulva monokróm, komplementer, triadikus, tetradikus, analóg és hangsúlyozott analóg színsémákat hozzunk létre.

Webszínek

Amennyiben egy weboldalnak a fehértől eltérő háttérszínt szeretnénk adni, könnyebb dolgunk van, mint gondolnánk. Ha például kék háttérrel szeretnénk használni, helyezzük el a `style="background-color:blue"` kódot a `<body>` elemben, vagy a `body` elem stíluslap-bejegyzésében. Természetesen kék helyett bármilyen más színt is választhatunk. A W3C szabványa összesen 16 színt sorol fel, név szerint a következőket: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow (akvamarin, fekete, élénkkék, fukszia, szürke, zöld, élénkzöld, gesztenyebarna, sötétkék, olajzöld, lila, vörös, ezüst, kékeszöld, fehér és sárga).

Természetesen a Weben ennél sokkal több szín megjelenik, olyannyira, hogy 140 nevesített szín létezik, amelyeket a böngészők mindenütt hasonlóan adnak vissza. Érdekes felsorolnunk néhányuk nevét, csak ízelítőképpen: azure, bisque, cornflowerblue, darksalmon, firebrick, honeydew, lemons chiffon, papayawhip, peachpuff, saddlebrown, thistle, tomato, wheat és whitesmoke.

Ezek azonban meglehetősen szubjektív elnevezések – ha rápillantunk például a 140 böngészőbarát színre, nem tudunk különbséget tenni a fuchsia és a magenta között. Ha ez után megfigyeljük a mellettük álló tizenhatos számrendszerű (hexadecimális) kódot, mindjárt világossá válik minden, hiszen a két kód megegyezik: az értékük #FF00FF. Ezekről a kódokról hamarosan bővebben is olvashatunk, most azonban elég annyit megjegyeznünk, hogy ha a W3C-szabványban szereplő 16 színen felül továbbiakat szeretnénk használni, ha csak lehet, ragaszkodjunk a hexadecimális színkódokhoz.

A hexadecimális színkódok voltaképpen 16 millió szín leírására alkalmasak, jöllehet napjaink legtöbb számítógépén ebből „mindössze” 16 384 jeleníthető meg. Ez azonban még mindig sokkal-sokkal több, mint 140 vagy 16.



Ha meg szeretnénk ismerni mind a 140 böngészőbarát szín nevét, hexadecimális kódját, valamint megjelenését a böngészőnkben, látogassunk el a http://www.w3schools.com/HTML/html_colornames.asp címre.



Érdeemes megemlítenünk, hogy a színek neveinek értelmezése nem függ a kis- és nagybetűk használatától. Így a Black, a black és a BLACK egyaránt a fekete színt azonosítja, jöllehet a legtöbb webtervező megmarad a csupa kisbetűs, illetve a nagy kezdőbetűs változatnál (már amennyiben egyáltalán használnak neveket – a legtöbben ugyanis a hexadecimális kódok kifinomultabb rendszere mellett teszik le a voksukat).

Fontos tudnunk, hogy a számítógép-monitorok nem pontosan ugyanazokkal az árnyalatokkal jelenítik meg a színeket. Ami a saját képernyőnkön csodálatos világoskéknek tűnik, az másutt valamilyen fura lilás árnyalattal jelenhet meg. A semleges földszínek (mint a középszürke, a barna vagy az elefántcsont) még kiszámíthatatlanabb eredményt adnak egyes monitorokon – olyannyira, hogy még attól függően is másképp viselkednek, hogy milyen napszakban, illetve világítás mellett nézzük a monitort.

A háttérszín megváltoztatása mellett lehetőségünk van arra is, hogy beleszóljunk a hivatkozások különböző állapotainak színeibe is (például különbséget tehetünk aközött, amikor a látogató a hivatkozás fölé viszi az egérmutatót, illetve aközött, amikor rá is kattint – de erről már tanultunk az előző órákon). Módosíthatjuk a tárolóelemek (bekezdések, rétegek, idézetek és táblázatcellák) háttérszínét, és színes szegélyeket is meghatározhatunk a számukra. A színek és a tárolóelemek kapcsolatára még ezen az órán példával szolgálunk.

Rengeteg szörnyű webhelyet találhatunk az Interneten, amelyeknek az alkotói halálosan komolyan gondolták a művüket, az irónia leghalványabb jele nélkül. Mindazonáltal „a világ legrosszabb webhelyét” (The World’s Worst Website), amelyet a 9.1. ábrán láthatunk, kifejezetten azért készítették, hogy egy helyen bemutassák a webtervezésben elkövethető súlyosabb hibákat, különös tekintettel a színhasználatra. Ha az alábbi képernyőfelvétel nem lenne elég, magunk is megtapasztalhatjuk a webes pokol tüzes leheletét a <http://www.angelfire.com/super/badwebs/main.htm> címen.



9.1. ábra

Részleges képernyőkép „a világ legrosszabb webhelyéről”

Ha a kedvenc keresőnkbe beírjuk a „bad web site examples” (példák rossz webhelyekre) kifejezést, rengeteg olyan webhelyet találhatunk, amelyek rossz példák tucatjait lapátolják egy helyre, és még azt is elmagyarázzák, hogy miért terjesztették fel őket Oscar helyett citromdíjra. Sokukat a megjelenésük miatt tartjuk „rossznak”, a megjelenés pedig a színekkel kezdődik. Végeredményben tehát, ha sikerül megértenünk a színek használatát és a meghatározásuk finom részleteit, máris nagy lépést tettünk a tetszetős webhelyek világa felé.

Hexadecimális színkódok

Ha meg szeretnénk felelni a szabványoknak, emellett megfelelően pontosan kívánjuk meghatározni a webhelyünk árnyalatait, a színekre a hexadecimális színkódjakkal érdemes hivatkoznunk. Ez a kód adja meg, hogy milyen arányban kell a vörös, a zöld és a kék színeket összekeverni ahhoz, hogy az adott színhez jussunk. Olyan ez, mint a színes gyurma – egy kis vörös, egy kis zöld, egy kis kék, és már kész is a kívánt szín!

A hexadecimális színek kódja `#rrggbb` alakú, ahol az `rr`, a `gg` és a `bb` a vörös (`rr`), zöld (`gg`) és kék (`bb`) színösszetevőket határozza meg. Ha a hexadecimális számok terén még nem szereztünk tapasztalatokat, akkor sem kell elkeserednünk – elég, ha megjegyezzük, hogy az `FF` a maximum, a `00` pedig a minimum. Az összetevőkben az alábbi kódokat alkalmazzuk:

- `FF` – teljes fényerő.
- `CC` – 80 százalékos fényerő.
- `99` – 60 százalékos fényerő.
- `66` – 40 százalékos fényerő.
- `33` – 20 százalékos fényerő.
- `00` – az összetevő nem játszik szerepet a színben.

Az élénkpiros szín kódja tehát `#FF0000`, a sötétzöldé `#003300`, a kékesliláé `#660099`, a középzürkéé pedig `#999999`. Ha tehát egy oldalon piros háttéren sötétzöld szöveget szeretnénk megjeleníteni, a megfelelő HTML-kód így fest:

```
<body style="background-color:#FF0000; color:#003300">
```

Jóllehet a fentiekben mindössze hat lehetséges kétjegyű hexadecimális értéket mutattunk be, összesen 256 létezik – a számjegyek skálája ugyanis 0-tól 9-ig és A-tól F-ig terjed, és minden párosítás lehetséges. Így értelmes hexadecimális szám az `F0` (amelynek a tízes számrendszerbeli értéke 240) vagy a `62` (98 a tízes számrendszerben), és így tovább.

Amint a korábbiakban is említettük, az `#rrggbb` kódban az `rr`, a `gg` és a `bb` a vörös, a zöld és a kék összetevőt adják meg, amelyeknek az értéke tízes számrendszerben 0-tól (az összetevő nem jelenik meg) 255-ig (az összetevő teljes fényerővel jelenik meg) terjedhet.

A fehér szín (`#FFFFFF`) vörös, zöld és kék összetevőjének értéke egyaránt 255, míg a fekete szín esetében (`#000000`) mindhárom összetevő 0. A tiszta piros szín kódja az `#FF0000` (tiszta piros, semmi zöld, semmi kék), a tiszta zöldé `#00FF00` (semmi piros, tiszta zöld, semmi kék), a tiszta kéké pedig `#0000FF` (semmi piros, semmi zöld, tiszta kék). A többi hexadecimális színek kódja is hasonlóképpen bontható 0-tól 255-ig terjedő alkotóelemekre. A webbiztos CornflowerBlue szín kódja a `#6495ED` – vagyis a vörös összetevője 40, a zöld összetevője 149, a kék összetevője pedig 237 (vagyis szinte maximális mértékben tartalmaz kék színt).

A színek választásánál – akár grafikai programot használunk, akár az Interneten vadászunk színekre – a színösszetevőket hexadecimális vagy tízes számrendszerbeli alakjukban láthatjuk. A „hexadecimal color converter” (hexadecimális színátalakító) kifejezésre rákeresve számos olyan segédeszközt találhatunk, amellyel a kapott színt úgy alakíthatjuk át, hogy ezután felhasználhassuk a stíluslapjainkon.

A háttér, a szöveg és a szegélyek színének beállítása CSS-kódokkal

A CSS-stílusok használatakor három helyen adhatunk meg színt: a háttér, a szöveg, valamit a szegély esetében. Az előző órákon is szó esett a színek meghatározásáról, de a színelmélet és a színekódok témakörét nem érintettük. Így a 8. órán megtanultuk, hogyan alkalmazhatunk színeket a hivatkozások állapotainak jelölésére, míg a 7. óra egyik fejtörőjében arra kértünk választ, hogy miként tölthetjük ki színekkel a táblázatok celláit.

A 9.2. ábra jól mutatja, hogy miként juttathatunk egy weboldalt ügyetlen színválasztással a citromdíj várományosai közé. Jómagam el sem tudom képzelni, hogy valaki is komoly webhelyen használja ezt a színösszeállítást – elrettentő példaként azonban nagyszerűen mutatja, hogy a színeket így is alkalmazhatjuk elemek stílusainak meghatározására.

A 9.1. példa a 9.2. ábra alapjául szolgáló XHTML- és CSS-kódot mutatja be.

9.1. példa A háttér, a szöveg és a szegélyek színeinek meghatározása CSS-stílusokkal

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Background, Text, and Border Colors</title>
  </head>

  <body>
    <h1>Background, Text, and Border Colors</h1>

    <p style="background-color:#CCCCCC;
border:1px solid #000000; color:#FF0000">
    Grey paragraph, black border, red text with a
    <span style="color:#FFA500">orange span</span>.</p>

    <div style="width:300px; height:75px; margin-bottom: 12px;
background-color:#000000; border:2px dashed #FF0000;color:
    #FFFFFF">
    Black div, red border, white text. </div>

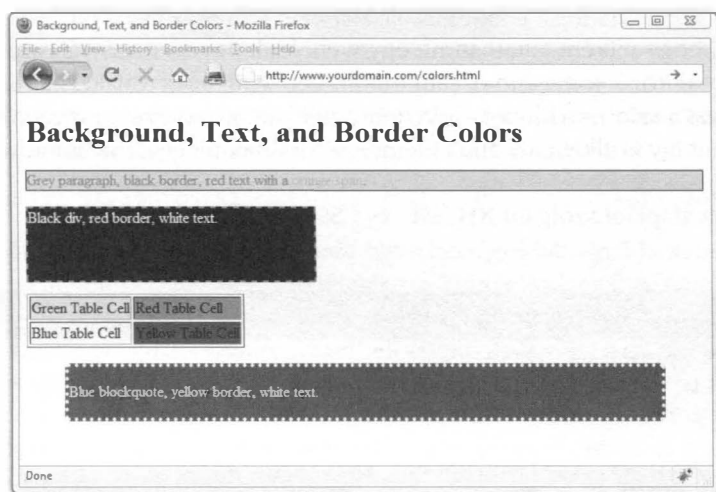
    <table border="1">
      <tr>
        <td style="background-color: #00FF00">Green Table Cell</td>
        <td style="background-color: #FF0000">Red Table Cell</td>
      </tr>
```

```

<tr>
<td style="background-color: #FFFF00">Blue Table Cell</td>
<td style="background-color: #0000FF">Yellow Table Cell</td>
</tr>
</table>

<blockquote style="background-color:#0000FF;
border:4px dotted #FFFF00; color:#FFFFFF"><p>Blue blockquote,
yellow border, white text.</p></blockquote>
</body>
</html>

```



9.2. ábra

CSS-stílusokkal meghatározhatjuk a háttér, a szöveg és a szegélyek színét

A 9.1. példában alkalmazott stílusokat nézve szinte minden világossá válhat, kivéve talán pár szegélystílust. A CSS-kódban csak úgy adhatunk színt a szegélyeknek, ha egyúttal meghatározzuk a szélességüket és a típusukat – a 9.1. példakód első példájában a szegély szélessége 1px, a típusa pedig `solid` (folytonos). A második példában ugyanakkor a szegély 2px széles, a típusa pedig `dashed` (szaggatott). Végezetül, a negyedik példa szegélye 4px széles, a típusa pedig `dotted` (pontosított).

Amikor a webhelyünk számára színeket választunk, mindig tartsuk észben, hogy sokszor a kevesebb több – ha valóban rajongunk valamilyen élénk és látványos színért, alkalmazzuk kiemelésként, ne pedig az elsődleges oldalelemek színezésére. Az olvashatóság érdekében használjunk világos háttéren sötét szöveget a sötét háttéren megjelenő világos betűk helyett.

Végezetül, ne feledkezzünk meg a látogatóink egy korántsem jelentéktelen csoportjáról, a színtévesztőkről. Ha a hozzáférhetőséget szeretnénk ellenőrizni, érdemes kipróbálnunk a <http://colorfilter.wickline.org/> címen található Colorblind Web Page Filter eszközt, így megfigyelhetjük, hogyan látja az oldalunkat egy színtévesztő.

Összefoglalás

Ezen az órán megismerkedtünk a színkezelés néhány hasznos módszerével, valamint a színtér használataival, amellyel könnyen megtalálhatjuk a szövegeinket jól kiemelő színeket. Emellett tanultunk a hexadecimális színkódokról is – amelyek az adott szín vörös, zöld és kék összetevőit adják meg –, valamint arról, hogy miként használhatjuk ezeket a kódokat az oldalon található elemek színeinek finomhangolására. Megismerkedtünk három stíluselemmel is, amelyek révén színeket rendelhetünk a tárolóelemek háttéréhez, szegélyeihez, valamint a bennük szereplő szöveghez a CSS-kódban.

A 9.1. táblázat ezekről a stílustulajdonságokról ad rövid összefoglalót.

9.1. táblázat A 9. órán bemutatott stílustulajdonságok

Jellemző/Stílus	Leírás
<code>style="background-color:szín"</code>	Egy elem (például <code><body></code> , <code><p></code> , <code><div></code> , <code><blockquote></code> vagy más tárolók) háttérszínét határozza meg.
<code>style="color:szín"</code>	Az elemen belül található szöveg betűinek színét határozza meg.
<code>style="border:méret típus szín"</code>	Az elem négy szegélyének színét határozza meg. A színbeállítást csak akkor alkalmazhatjuk, ha megadjuk a szegély vastagságát és típusát is.

Kérdezz-felelek

- K: *Igaz, hogy a böngészők lehetővé teszik a felhasználóknak a háttér- és betűszín megváltoztatását?*
- V: Igaz. A látogatóink módosíthatják a böngészőben azokat a színeket, amelyeket a webhely készítőjeként meghatároztunk. Így a kék háttéren szereplő fehér betűket egyesek zöld-fehérben vagy egyéb kedvenc színeikkel nézhetik – jóllehet kevesen élnek ezzel a lehetőséggel. Általában azok a színek jelennek meg, amelyeket a `<body>` elemben megadunk.
- K: *Úgy hallottam, hogy 231 „webbiztos” szín létezik, és ajánlatos ezek használatához ragaszkodnunk a weboldalainkon. Igaz ez?*
- V: A történet valódi háttere a következő: létezik 231 szín, amely kevésbé „zavarosan” jelenik meg azoknál a felhasználóknál, akik 256 színű (8 bites) videómódot használnak. Egyes webtervezők ezért próbálnak kitartani a használatuk mellett. Mindazonáltal a mai képernyők javarészt támogatják a bővebb színpaletták megjelenítését, így minden szín egyforma tisztasággal jelenik meg rajtuk. Ha tehát a grafikai programunk kidob egy hexadecimális színkódot, nyugodtan használjuk fel a stíluslapunkon, és alkalmazzuk az így kapott egyedi színsémát.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Hogyan jelenítenénk meg egy weboldalt fekete háttérrel és élénkzöld betűkkel? Érdemes így tennünk egyáltalán?
2. Egy weboldal színsémájának meghatározásakor melyik színösszeállítás ad tágabb teret a színválasztásra: az analóg, a komplementer vagy a triadikus rendszer?
3. Tegyük fel, hogy a `<body>` elem stílusbejegyzésében a `background-color #FFFFFF`, míg az első `<div>` elemben a `background-color #FF0000` beállítással élünk. Milyen háttérszínnel jelenik meg ez a `<div>` – vörössel vagy fehérrel?

Válaszok

1. Helyezzük el az alábbi kódsort az oldal kódjának elején, illetve a megfelelő bejegyzést a stíluslapon:

```
<body style="background-color:#000000; color:#00FF00">
```

2. A triadikus. Így három, a színkörön egyenletesen elosztott színt határozhatunk meg.
3. A <div> háttere vörös lesz, ugyanis a tárolóelem háttérszínének meghatározása felülírja az őt tartalmazó <body> elem, illetve a megfelelő stíluselem beállításait.

Gyakorlatok

- Válasszunk egy nekünk tetsző kiindulási színt – lehet ez például egy gyönyörű kék vagy valamilyen földszín –, és állítsunk elő színkészletet a Color Scheme Generator (<http://colorschemedesigner.com/>) segítségével. Érdeemes kipróbálnunk a tetradikus vagy a hangsúlyozott analóg sémákat.
- Ha elkészültünk a színpaletta összeállításával – akár többel is –, írjunk egy egyszerű HTML-oldalt egy <h1> elemmel, valamint egy bekezdésnyi szöveggel és esetleg néhány listaelemmel. A korábban megismert stílusokkal határozzuk meg az oldal háttérszínét, valamint az egyes tömbszintű elemek szövegében alkalmazott betűszínt, és figyeljük meg, hogyan jelennek meg együttesen. Az egymásra hatásuk tanulmányozása révén állapítsuk meg, hogy mely színek a legalkalmasabbak a tárolók megjelenítésére, továbbá melyek használhatók jól az egyszerű szöveg, a címsorok, valamint a hivatkozások betűihez.



10. ÓRA

Képek készítése webes használatra

A lecke tartalma:

- Mi alapján válasszunk grafikai alkalmazást?
- Hogyan készítsük elő a fényképeinket internetes használatra?
- Reklámcsíkok és nyomógombok létrehozása
- A színek számának csökkentése a képeken
- Átlátszó képek létrehozása
- Ismétlődő képekből álló háttér kialakítása
- Animációt tartalmazó webes grafika készítése

Bár a webhelyek készítésekor kétségtelenül törődnünk kell a színösszeállítással és a vonzó küllemmel, ahhoz, hogy hatásos képek kerüljenek a weboldalainkra, nem kell profi művésznek lennünk. Ami még fontosabb, nem kell sok-sok pénzt költenünk szoftverre sem. A mai óra végére képesek leszünk olyan képek készítésére, amelyeket felhasználhatunk a webhelyünkön. A fejezet mintapéldái ugyan egy bizonyos ingyenes, a Windows-, a Linux- és a Macintosh-felhasználók körében népszerű programot – a GNU Image Manipulation Program, azaz a GIMP nevűt – használnak, az órán meg-

szerzett tudás azonban bármely, a Windows, illetve a Macintosh operációs rendszeren futó jelentősebb grafikai szoftver esetében is alkalmazható, még akkor is, ha a menük és a beállítási lehetőségek kissé különböznek.

A mai órán csak azzal foglalkozunk, hogy miként készítsük el magukat a különféle célokra szolgáló képeket. A 11. órán tárgyaljuk azt, hogy a HTML és a CSS segítségével miként illesztetők ezek a képek a weboldalainkba.

A grafikai alkalmazás kiválasztása

A webhelyünkhöz készülő képek létrehozására jóformán bármelyik grafikai alkalmazást választhatjuk, kezdve azzal az egyszerű rajzolóprogrammal, amely rendszerint ingyenesen jár a számítógépünk operációs rendszeréhez, egészen az olyan drága és profi alkalmazásokig, mint az Adobe Photoshop. Ha pedig rendelkezünk a számítógépünkhöz csatlakozó digitális fényképezőgéppel, illetve lapolvasóval, akkor feltehetőleg a vele kapott program is alkalmas a weboldalainkhoz szükséges képek elkészítésére. Ugyanakkor számos olyan ingyenes, letölthető képszerkesztő program létezik – sőt webalkalmazásokat is találunk –, amely kifejezetten fényképek feldolgozására alkalmas.



A képszerkesztő programok krémjét kétségtelenül az Adobe Photoshop jelenti. Ha azonban a számítógépes programokkal való munka területén nem rendelkezünk gyakorlattal, akkor ez az alkalmazás meglehetősen összetettnek bizonyulhat, és az ára sem elhanyagolható. Az Adobe cég termékeiről bővebb információt a cég honlapján (<http://www.adobe.com>) találunk. Ha az Olvasót egyik-másik termékük érdekli, akkor töltse le az ingyenes próbaváltozatot az említett oldalról.

Ha az Olvasó már rendelkezik azzal az alkalmazással, amelyik megítélése szerint megfelel a webes képek elkészítésére, akkor használja azt az óra során. Ha a sorra kerülő feladatok némelyikének elvégzésével gondunk támad, akkor a választott szoftver esetleg mégsem elegendő tudású a webes grafika céljaira. Ez esetben a <http://www.gimp.org> honlapról töltsük le és telepítsük a GIMP nevű grafikai programot. A program teljes értékű, és teljesen ingyenesen használható.

1. Látogassunk el a <http://www.gimp.org> oldalra, és kattintsunk a Downloads (Letöltések) hivatkozásra.
2. Elvileg egy olyan hivatkozást látunk, amellyel a programnak az általunk használt operációs rendszerhez készített változatát tudjuk letölteni. A Show other downloads (A további letöltések megmutatása) hivatkozással valamennyi választási lehetőség láthatóvá válik. Ha előtűnik a mi operációs rendszerünkhöz készült változathoz vezető hivatkozás, akkor kattintsunk rá, és a letöltés megkezdődik.
3. A letöltés végeztével a fájlra duplán kattintva kezdhetjük meg a telepítést.

Más webhelyekről származó anyagok használata

A webhelyünkre kerülő képek és egyéb médiafájlok elkészítésével talán akkor végzünk a leggyorsabban, ha hozzá sem fogunk a munkához. Egy weboldalról képet szerezni nem túl bonyolult: rákattintunk a jobb egérgombbal (Macintosh-egér használata esetén a bal gombbal, és lenyomva tartjuk a billentyűt), majd a böngészőnk függvényében a Save Image As, illetve a Save Picture As (Kép mentése, Kép mentése más néven) menüpontot választjuk. Egy weboldal háttérképének kinyerése sem bonyolultabb: kattintsunk rá a jobb egérgombbal, és válasszuk a Save Background As (Háttérkép mentése) menüpontot.

Mindazonáltal a tulajdonos kifejezett engedélye nélkül sohasem tanácsos egy képet használnunk. Kérjünk tehát engedélyt, vagy keressünk Creative Commons felhasználási engedéllyel rendelkező képeket. Ha a képet kifejezett engedély nélkül használjuk, akkor megsértjük a szerzői jogokat – és ez úriemberhez kevéssé méltó cselekedet. A szerzői jogokról bővebben a <http://www.utsystem.edu/OGC/IntellectualProperty/cprtindx.htm> hivatkozást követve olvashatunk, a magyar olvasónak pedig az 1999. évi LXXVI. törvényt érdemes tanulmányoznia.

Érdemes lehet a bárki által szabadon használható képek használatát is megfontolnunk – ez esetben a használat nem engedélyköteles. Az Interneten ilyen képeket például a Microsoft cég Office Online Clip Art and Media webhelyén (<http://office.microsoft.com/clipart/>) találhatunk. A Barry's Clipart Server (<http://www.barrysclipart.com/>) szintén népszerű lelőhelye az ingyenesen használható képeknek.

Amit a grafikáról mindenképp tudnunk kell

Amikor a webhelyünkön képeket vagy egyéb multimédiás tartalmat helyezünk el, mindig két törekvés között kell megtalálnunk az ideális egyensúlyt. A felhasználó szemének és fülének az a jó, ha a tartalom annyira részletes és pontos, amennyire ez egyáltalán lehetséges, de a felhasználók azt is szeretik, ha az információ azonnal megjelenik. A bonyolult, színgazdag képek nagyobb fájlokat is jelentenek, ami viszont még gyors internetkapcsolat esetén is lassabb letöltéshez vezet. Hogyan növelhető a képek minősége minél kisebb fájl méret mellett? Ha helyes döntést szeretnénk hozni, meg kell értenünk, hogy a színmélység és a felbontás miként eredményezi a jó minőségű kép szubjektív érzetét.

A kép felbontása alatt a képet alkotó pontoknak, vagyis a képpontoknak (pixel) a számát értjük. A nagy méretű, magas felbontású képek általában lassabban töltődnek le, és jelennek meg, mint a kicsi, alacsony felbontású képek. A felbontást rendszerint képpontban, a kép szélességének és magasságának szorzataként adjuk meg. Például egy 300 x 200 képpontos kép 300 képpont széles és 200 képpont magas.



A képek felbontása többféleképpen is megadható. Megadhatjuk képpontban, távolságegységre vonatkoztatva, színtartományként, időben és radiometrikusan. Órákat tölthetünk azzal, hogy egy-egy módot megismerjünk, és ha ez grafikustanfolyam lenne, akkor ezt is tennénk. Egyelőre azonban mindössze annyit kell észben tartanunk, hogy nagy képek lassan töltődnek le, és a monitoron is sok helyet foglalnak. Az elfoglalt hely és a tárigény olyan tulajdonság, amelyet a webhelyünk megtervezésekor érdemes figyelembe vennünk.

Esetleg meglepve olvastuk, hogy a képhez szükséges tárterület – és így a hálózati átviteli idő – nagysága nem elsősorban a kép felbontásától függ. Ennek az az oka, hogy a weboldalakon tárolt képeket mindig tömörített formában tároljuk és továbbítjuk. A képek tömörítése egy olyan matematikai művelet eredménye, amelynek során a képekben ismétlődő mintázatokat keresünk. A képtömörítés matematikája meglehetősen összetett tananyag. Az alapelv az, hogy az ismétlődő mintázatok, illetve a nagy, egyszínű területek tömörítve is tárolhatók. A képfájl így lényegesen kisebb lesz, ezáltal az Interneten is gyorsabban tudjuk továbbítani. A kép megjelenítésekor a webböngésző feladata az eredeti kép visszaállítása.

Az óra hátralévő részében azt tárgyaljuk, hogy miként lehet látványos, de kis fájl méretű képeket készíteni. A használt módszerek az egyes képek tartalmától és céljától függően változnak. Ahány weboldal, annyiféle képhasználati mód, de a képek mégis főleg az alábbi négy feladatkörben kapnak szerepet a weboldalakon:

- emberekről, tárgyakról, tájakról, épített környezetről készült fényképek;
- rajzolt reklámcsíkok és emblémák;
- a lehetséges tevékenységekre utaló, illetve hivatkozásokat rejtő nyomógombok;
- a tárolóelemek háttérképei.

A fényképek előkészítése

Ha a weboldalainkon fényképeket szeretnénk elhelyezni, akkor a nyomtatott képeinket kell digitálissá alakítanunk, vagy digitális fényképezőgéppel kell azonnal digitálisan rögzített képeket készítenünk. Elképzelhető, hogy a lapolvasónkhoz, illetve fényképezőgépünkhöz kapott alkalmazást kell használnunk a képek merevlemezre mentéséhez, de lehet, hogy a fényképezőgépről az egerünkkel is áthúzzhatjuk a képeket a lemezre.

Ha lapolvasóval készítjük el a nyomtatott képeink digitális változatát, akkor érdemes tudnunk, hogy a lapolvasót jóformán bármely általunk választott grafikai alkalmazásból vezérelhetjük – a részleteket az alkalmazás leírásában találjuk.



Ha sem lapolvasónk, sem digitális fényképezőgépünk nincs, akkor jó tudni, hogy szinte valamennyi filmelőhívó helyen meg tudják oldani a 35 milliméteres filmjeink tartalmának CD-re vagy DVD-re írását. A CD-ről, illetve DVD-ről ezt követően a merevlemezre másolhatjuk a képeket, és egy grafikai alkalmazással megnyithatjuk, illetve módosíthatjuk azokat.

Miután a digitális kép a számítógépünkre került, a grafikai alkalmazásunk segítségével vághatjuk, átméretezhetjük, színekorekciót végezhetünk rajta, és tömöríthetjük – más szóval kedvünk szerint alakíthatjuk át őket a webes használatra.

Képek körülvágása

Minthogy a Webre szánt képeink méretét szeretnénk minél alacsonyabban tartani, rendszerint meg szoktuk vágni a digitális képeket. A fénykép *körülvágásakor* kijelöljük a megtartandó területet, és kidobjuk a többit.

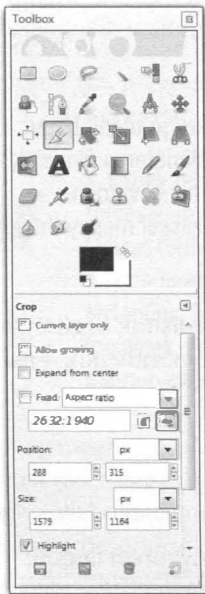
Önálló feladat

Vágás a GIMP-ben

A GIMP eszköztárában gyorsan megtaláljuk a Crop (Vágó) nevű eszközt és a lehetséges beállításait. Keressünk egy képfájl – lehet saját készítésű, a merevlemezünkön tárolt digitális kép, de letölthetünk egyet az Internetről is. Nyissuk meg a képet a GIMP-ben, és vágjuk meg, mégpedig az alábbiak szerint:

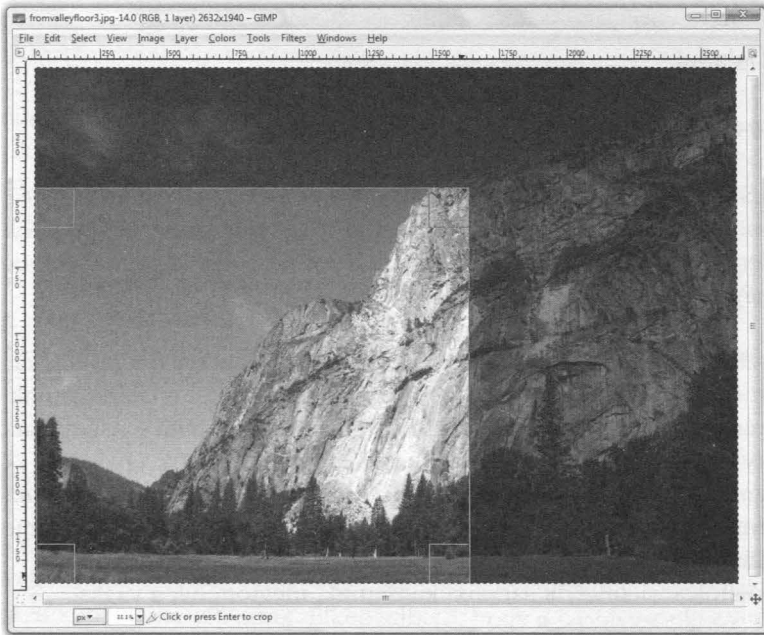
1. A GIMP eszköztárában kattintsunk a Crop eszköz ikonjára (lásd a 10.1. ábrát). A kiválasztott eszköztől függően esetleg további jellemzőket is beállíthatunk. A 10.1. ábrán történetesen a Crop eszköz beállítási lehetőségeit – Aspect ratio (Méretarány), Position (Pozíció), Size (Méret) és társaik – látjuk.
2. A körülvágni kívánt képen rajzoljuk körbe a kijelölni kívánt részt, a megtartani kívánt résznek kattintsunk a bal felső sarkába, tartsuk lenyomva a bal egérgombot és húzzuk az egérmutatót a rész jobb alsó sarkába (lásd a 10.2. ábrát).
3. Amikor a kijelölés valamelyik sarkába kattintunk, megtörténik a vágás.

Ha más grafikai alkalmazást használunk, akkor alighanem a fentiekől eltérően kell eljárunk, de az alapelv azonos marad: jelöljük ki azt, ami marad, és vágjuk le a maradékot.



10.1. ábra

Az eszköztárról válasszuk a Crop eszközt



10.2. ábra

Kijelöljük a kép megtartani kívánt részét

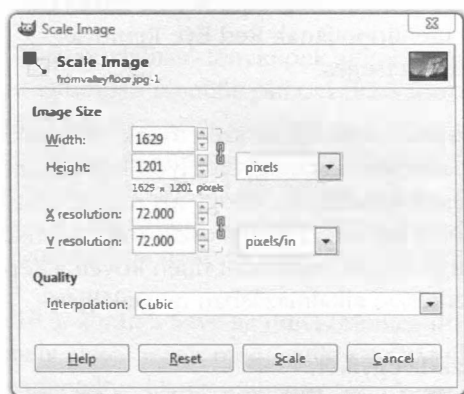
A kép esetleg a körülvágást követően is túl nagy marad – nagyobb annál, mint ami a weboldalunkra kell. Az adott weboldaltól függ ugyan, de elképzelhető, hogy azt szeretnénk, a nagy képek ne legyenek nagyobbak egy adott méretnél. Például ha a kép önmagában kerül a monitorra egy katalógus elemeként, akkor legfeljebb 800 x 600 képpont legyen, vagy 640 x 480 képpont, vagy akár még kisebb. Ha szöveg mellett kap helyet egy kép, akkor sokszor mindössze 250-350 képpont széles lehet, hogy a szövegnek is maradjon hely. Vannak esetek, amikor a kép nagyobb változatához a kép egy bélyeg méretű változatán keresztül kívánunk hivatkozást elhelyezni – ilyenkor a bélyegkép hosszabb oldala is legfeljebb száz képpontnyi lehet.



A grafikai alkalmazásunknak feltehetőleg lesz egy olyan része, amely – valahol a képet tartalmazó ablakban – folyamatosan megjeleníti a kép méretét. A GIMP az aktuális képméretet az ablak felső részén írja ki. Más programok esetleg a jobb alsó vagy a bal alsó sarokban jelenítik meg ezt az információt. Az ablakban látható esetleg a nagyítási arány, és – ugyanitt talán lehetőséget kapunk az arány megváltoztatására is – ráközelíthetünk a képre, vagy eltávolodhatunk tőle.

Kép átméretezése

A képek átméretezéséhez szükséges eszköz neve és helye a használt programtól függ. A GIMP-ben nyissuk meg az Image (Kép) menüt, és válasszuk a Scale Image (Kép átméretezése) pontot. Ekkor a 10.3. ábrán is látható Scale Image párbeszédablak nyílik meg.



10.3. ábra

Ha egy kép méretét meg szeretnénk változtatni, használjuk a Scale Image párbeszédablakot

Majdnem mindig a jelenlegi méretarány megtartása mellett szeretnénk az átméretezést végezni – azaz azt szeretnénk, ha a kép kívánt szélességét megadva a programunk automatikusan kiszámítaná a magasságot, és fordítva. Így a kép alakja nem torzul. A GIMP-ben a méretarány alapértelmezés szerint rögzített – ezt mutatja a 10.3. ábrán

megjelenő Width (Szélesség) és Height (Magasság) beviteli mezőket összekötő lánc is. Ha az egérrel a láncra kattintunk, akkor elszakad, és így olyan szélesség- és magasság-értéket adhatunk meg, amelyet úri kedvünk diktál – legfeljebb torzul a kép.



A GIMP sok eszközhöz hasonlóan a Scale Image párbeszédablak is az átméretezett kép ablaka előtt jelenik meg. Az ablakok ilyen elhelyezésével lehetővé válik, hogy a párbeszédablakban megadjuk, hogy min kell változtatni, alkalmazzuk a változtatást, és azonnal láthatjuk az eredményt.

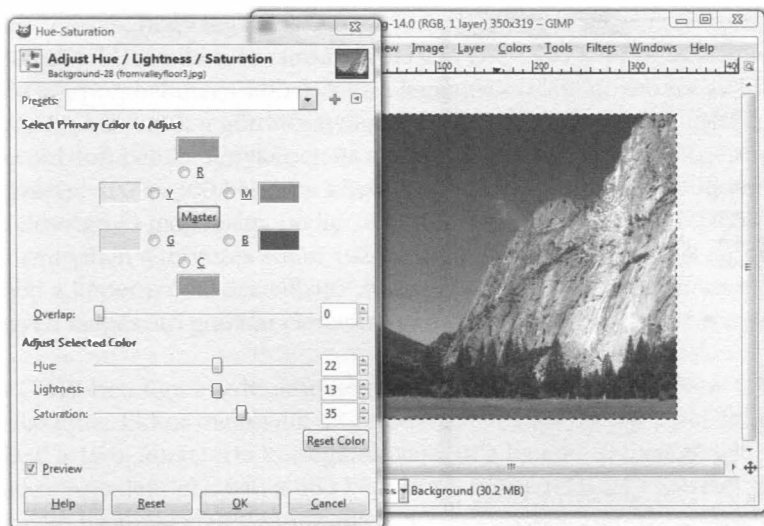
A legtöbb, ha nem valamennyi grafikai alkalmazásban lehetőségünk nyílik egy kép átméretezésére úgy is, hogy nem pontosan, képpontban adjuk meg az új méretet, hanem százalékban. Ha például van egy 1629 x 1487 képpontos képünk, és nem óhajtunk a szélesség felének kiszámításához szükséges osztással bajlódni, akkor válasszuk egyszerűen a Percent (Százalék) lehetőséget – esetünkben a 10.3. ábrán látható lenyíló listából (amelyben most a pixel lehetőség van kiválasztva). Az alapértelmezett 100 helyére írjuk azt, hogy 50. A kép így 815 képpont szélessé és 744 képpont magassá alakul, mi pedig megússzuk a számolgatást.

Fényképek színeskorrekciója

Ha saját grafika készítése helyett inkább fényképek átalakításával próbálkozunk, könnyen elképzelhető, hogy a megfelelő hatást színeskorrekcióval tudjuk csak elérni. Ahogy sok képszerkesztő alkalmazás, a GIMP is számos lehetőséget kínál a kép fényerejének, kontrasztjának és színegyensúlyának beállítására, és persze a rettegett vörösszem-hatás is kiküszöbölhető. A vörösszem-hatás megszüntetése a GIMP-ben a Filters (Szűrők) menü Enhance (Kiemelés) menüpontjának Red Eye Removal (Vörösszem-eltávolítás) lehetőségét választva lehetséges.

A műveletek nagy része magától értetődő. Ha fényesebb képet szeretnénk, állítsunk a fényerőn. Ha több vöröset szeretnénk látni a képen, a színegyensúlyon kell állítanunk. A GIMP Colors (Színek) menüjében számos eszközt találunk. Ahogy azt már az előző részben, a Scale Image párbeszédablakról szólva leírtuk, az egyes eszközök a munkaterület előtt jelenítik meg a párbeszédablakukat. A színek beállítását hűen követi a kép változása. Az ilyen előnézet a legtöbb képszerkesztő alkalmazásban megtalálható.

A 10.4. ábrán a Colors menüben lévő sok eszköz egyikét, az Adjust Hue/Lightness/Saturation (Árnyalat/fényerő/telítettség beállítása) eszközt látjuk. Ahogy az ábrán látható, sok színbeállítás egyszerűen a párbeszédablakok különféle csúszkáival végzett értékmódosításokkal valósítható meg. A Preview (Előkép) jelölőnégyzetet bekapcsolva a változtatások közben is nyomon követhetjük a ténykedésünk eredményét. A Reset Color (Szín visszaállítása) gomb hatására a kép az eredeti állapotába áll vissza, és a változtatásaink érvényüket veszítik.



10.4. ábra

Az *Adjust Hue/Lightness/Saturation* eszköz egyike a GIMP számos, csúszkával működő eszközeinek

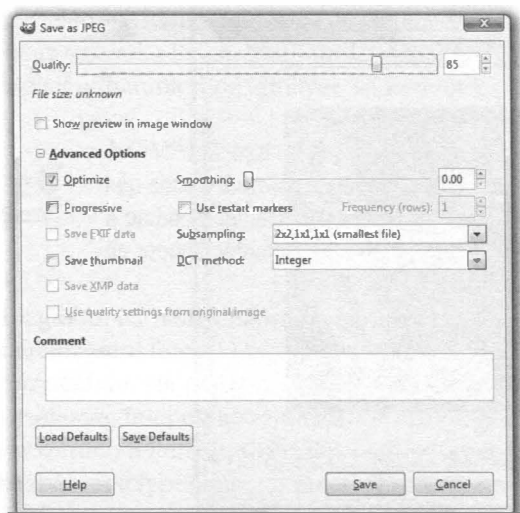
Mint hogy rendelkezésre álló eszköz van egy pár, és mindegyik képes előnézetet mutatni, az egyes eszközök szerepének megismerésére a legalkalmasabb módszer egy kis játékos kísérletezés.

A JPEG tömörítés beállítása

A Weben található fényképek akkor a leghasználhatóbbak, ha nem GIF, hanem JPEG formátumban mentjük őket. A JPEG formátummal a fájl méretét is kezelhető nagyságon tarthatjuk, és a színek számát sem kell csökkentenünk. Amikor elkészültünk a fénykép méretének és kinézetének beállításával, válasszuk a File, Save As (Fájl, Mentés másként) menüpontot, és a fájl típusaként adjuk meg a JPEG lehetőséget. Ha más grafikai alkalmazást használunk, valószínűleg abban is lesz egy hasonló párbeszédablak, amelyben a tömörítés mértéke és más különféle JPEG-beállítások adhatók meg.

A 10.5. ábrán a Save as JPEG (Mentés JPEG formátumba) párbeszédablak látható, amely akkor nyílik meg, amikor a GIMP-ben egy JPEG képet mentünk. Az ábrán megfigyelhetjük azt a Quality (Minőség) csúszkát, amellyel a tömörítés mértéke 1 (rossz minőség, kis fájl méret) és 100 (jó minőség, nagy fájl méret) között állítható.

Esetleg szívesen kísérletezgetnénk egy keveset, megfigyelve, hogy a különféle JPEG tömörítési szintek milyen hatással vannak a képek minőségére. Mindazonáltal a 85%-os minőség (ha úgy tetszik: 15%-os tömörítés) a legtöbb fénykép esetében megfelelő egyensúlyt jelent a fájl méret – így a letöltés sebessége – és a minőség között.



10.5. ábra

A GIMP a kép minőségének fenntartása mellett úgy teszi lehetővé a fájl méret csökkentését, hogy a képet JPEG formátumban menti

Reklámcsíkok és nyomógombok készítése

Azoknak a képeknek az esetében, amelyeket a semmiből hozunk létre – ilyenek a reklámcsíkok (banner) és a nyomógombok (button) – meg kell fontolnunk pár olyan dolgot is, amelyek alapvetően megkülönböztetik őket a fényképektől.

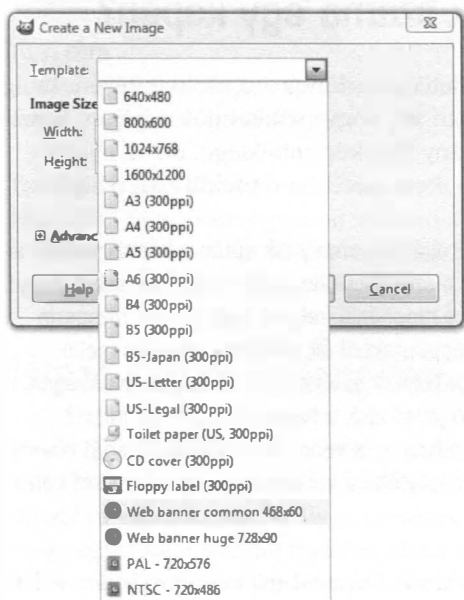


Tervezéskor sok éven át alapvetően 800 x 600 felbontású monitorokat feltételeztünk. Ezt az alacsony számot nem árt még most is fejben tartanunk, mert sokan nem állítják teljes képernyőre az alkalmazásaikat. Mindazonáltal ma már nem baj az sem, ha 1024 x 768 képpont felbontású képernyőre tervezzük a weboldalainkat.

A reklámcsíkok és a nyomógombok tervezésekor az első meghozandó döntésünk a kép méretével kapcsolatos. Manapság a legtöbb internetezőnek legalább 1024 x 768 képpont felbontású a képernyője – sokuknak pedig ennél lényegesen nagyobb. A szerző például éppen egy 1440 x 900 képpontos monitor előtt ül. Általában elmondható, hogy a képeinket úgy kell megterveznünk, hogy mindig elférjenek egy kisebb (1024 x 768-as) képernyőn is, és még a gördítősávoknak és a margóknak is maradjon hely. A méretet érintő megfontolások közül az oldal szélessége a fontosabb, mivel a vízszintes görgetés gondot okozhat, és megzavarja az internetezőket. Az oldalak függőleges görgetése lényegesen elfogadhatóbb, azaz nem jelent problémát, ha a weboldalaink magasabbak a minimális képernyőméretnél.

Feltételezve, hogy legalább 800 x 600-as felbontású képernyőre tervezünk, a teljes méretű reklámcsíkok és címsávok ne legyenek 770 képpontnál szélesebbek, illetve 430 képpontnál magasabbak – ez a legnagyobb látható terület az oldalon, miután számításba vettük a gördítősávokat, az eszköztárakat és a böngészőablak más részeit. Az oldalon belül a fényképek és a grafikák kiterjedése függőlegesen és vízszintesen egyaránt 100 és 300 képpont között változhat, a kisebb nyomógombok és ikonok szélessége és magassága pedig 20 és 100 képpont közé essen. Nyilvánvaló, hogy amennyiben a tervezés során 1024 x 768 képpontos felbontást tartunk szem előtt, akkor több a képernyő kihasználható „hasznos területe”, de a reklámcsíkokra, gombokra és egyéb kiegészítő grafikai elemekre vonatkozó alapelvek most is ugyanúgy érvényesek.

A GIMP-ben úgy kezdünk új képet, hogy a File menüből a New (Új) menüpontot választjuk. Ekkor megnyílik a Create New Image (Új kép létrehozása) párbeszédablak (lásd a 10.6. ábrát). Ha kétségeink vannak a leendő kép megfelelő méretét illetően, akkor fogadjuk el a 640 x 480 képpontos alapértelmezett méretet. A Template (Sablon) lenyíló listából választhatunk más, előre megadott méretű képet is – például a Web banner common 468x60, vagy a Web banner huge 728x90 méreteket. A weboldalak reklámcsíkjai esetében a beállítások valóban a szokásos („common”) és az óriási („huge”) méretet jelentik. Azt is megtehetjük, hogy az új kép szélességét és magasságát saját kezűleg adjuk meg.



10.6. ábra

Mielőtt munkához látnánk, döntenünk kell a kép méretéről

Ami a kép háttérszínét illeti, rendszerint érdemes fehéret választanunk, igazolva ahhoz, ahogy a legtöbb böngésző a weboldalt megjeleníti – bár, mint az előző órán megtanultuk, ez megváltoztatható. Persze, ha tudjuk, hogy a készülő weboldalunk háttere nem fehér, akkor adhatunk a képnek másmilyen háttérszínt is. Az is lehet, hogy egyáltalán nem kívánunk a képnek hátteret adni. Ilyenkor a háttérszínnek közül a Transparency (Átlátszóság) lehetőséget kell választanunk. Ha a kép háttere átlátszó, akkor a weboldalnak a kép mögötti részét látjuk a kép üresen maradt részei mögött. A GIMP-ben az új képek háttérszíne a Create New Image párbeszédablak Advanced Options (Speciális lehetőségek) részét megnyitva állítható be.

Amikor megtudtuk, hogy a kép hány képpont széles és magas legyen, és az OK-ra kattintottunk, egy üres rajzvásznat kapunk – ami elég elrettentő, ha az Olvasó e könyv íróihoz hasonlóan kétoldali rajzolitisszal bajlódik. Mindazonáltal annyi internetes grafikai tananyag létezik – és akkor a kizárólag ennek a témának szentelt könyvekről nem is szóltunk –, hogy most, bízva alkotásvágyában, nyugodt lélekkel engedjük el az Olvasó kezét. A mai órának mindössze az a célja, hogy a weboldalainkra szánt képekkel kapcsolatban felhívja a figyelmet néhány dologra – ez nem jelenti azt, hogy meg is tanuljuk elkészíteni a képeket. A *saját magunk* választotta program teljes megismeréséhez vezető úton akkor léphetünk előre, ha a fenti alpműveletekkel már tisztában vagyunk.

Hogyan csökkenthető a színek száma egy képen?

A kép méretének – és ebből következően a letöltési időnek – a csökkentésére szolgáló módszerek egyik legeredményesebbike abban áll, hogy csökkentjük a képen használt színek számát. Ezzel lényegesen romlik néhány fénykép minősége, de az eljárás a legtöbb reklámcsik, nyomógomb és egyéb ikon esetében remekül használható.



Árnyalásnak (dithering) nevezzük azt az eljárást, amelynek során a képszerkesztő alkalmazás úgy igyekszik visszaadni egy a színpalettában jelen nem lévő színt, hogy két hasonló szín képpontjait váltogatja a megfelelő helyen. Egy árnyalt rózsaszín például egymást váltó piros és fehér képpontokból áll, amelyek egymás mellé kerülve rózsaszíneknek látszanak. Az árnyalás sok esetben javít a képek minőségén, de a webes grafikák esetében mégsem javasoljuk a használatát. Hogy miért? Azért, mert a lényegéből adódóan a kép hordozta információ bonyolultságát növeli, ami viszont általában a fájl méret, illetve a letöltési idő lényeges növekedését vonja maga után.

Bizonyára örömmel értesülünk arról, hogy a kevés színnel bíró képek számára is létezik fájlformátum. A neve: Graphics Interchange Format (grafikamegosztási formátum), azaz GIF. Ha egy képet GIF formátumban mentünk, figyelmeztetést kaphatunk, miszerint az eddig különálló rétegeket a programunk egygé lapítja, illetve a palettás formátumba

való átalakítás miatt csökkenni fog a kép színeinek száma. Nos, ezek a GIF formátum miatt szükséges átalakítások – mint az a 10.7. ábrán is látható. A párbeszédablak arra szólít fel, hogy vegyük tudomásul az előzőekben említett változásokat, és amelyek ezután a mentési folyamat részeként meg is történnek. Ne aggódjunk, ha egyelőre nem teljesen értjük a beállítási lehetőségeket, de szánjunk időt arra, hogy elolvassuk, amit az alkalmazásunk súgófájljában a rétegekről, illetve a színpalettás képekről találunk.

Jegyezzünk meg annyit, hogy a GIF formátumot olyan képekhez szánták, amelyekben nagyobb egyszínű foltok vannak – azaz weboldalak címsávjai, vagy más, rajzolt képek – a fényképekhez azonban messze nem ideális.



10.7. ábra

Amikor egy képet GIF formátumban mentünk, esetleg figyelmeztetést kapunk, hogy így indexelt színpalettás kép jön létre

További, a jelentősebb webböngészők mindegyikében támogatott fájlformátum a PNG. Míg a GIF formátum egyetlen átlátszó szín használatát támogatja – ahol aztán majd a weboldal színe válik láthatóvá –, a PNG formátum továbblép: itt az átlátszóság különböző fokozatait adhatjuk meg.

Átlátszó képek használata

Találkozhattunk már olyan weboldalakkal, amelyek a tárolóikban háttérszín, illetve háttérképet használtak, de olyan képek is voltak bennük, amelyek egyes részein láthatóvá vált a háttér. Az ilyen esetekben az elől lévő kép egyes részei átlátszóak voltak, de az egyébként mindig téglalap alakú kép üresen maradt részei nem látszottak. Sok esetben remek alkalom nyílik az ilyen részlegesen átlátszó képek használatára, ha a rajzaink egyszerűen jobban néznek ki közvetlenül az alkalmazott háttérszín, illetve háttérkép előtt.

Ha egy kép egy részét átlátszóvá szeretnénk tenni, akkor vagy GIF, vagy PNG formátumban kell mentenünk. Ahogy az óra korábbi részében már szóltunk róla, a legtöbb GIF formátumot támogató alkalmazással egy színt tehetünk átlátszóvá, a PNG formátum esetében azonban az átlátszóság különböző fokozatairól lehet beszélni. A PNG formátum főként a többféle átlátszóság okán tekinthető jobbnak a GIF-nél. A legfrissebb webböngészők mostanra mind képesek a PNG formátum kezelésére. A PNG képformátumról a <http://www.libpng.org/pub/png/pngintro.html> weboldalon tudhatunk meg többet.

Az átlátszó képek létrehozása során alkalmazott eljárás a használt képformátumtól (GIF vagy PNG), illetve a létrehozásra használt grafikai alkalmazástól függ. Segítséget a grafikai alkalmazásunk súgófájljaiban találunk, de az is megfelelő megoldás, ha valamelyik internetes keresővel az *átlátszó kép készítése* (ide az *alkalmazásunk neve* kerül) kifejezésre (vagy angolul a *transparent image* kifejezésre) keresünk.

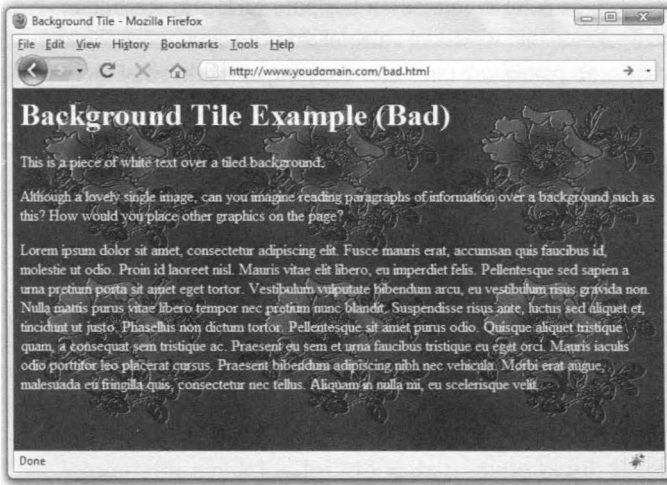
Mozaikos háttér kialakítása

Bármelyik GIF vagy JPEG formátumú kép alkalmas arra, hogy egy tárolóelem háttérének alkotója legyen. Mindazonáltal, mielőtt hozzálátnánk a mozaikszerű háttér – különösen egy mintagazdag mozaikos háttér – kialakításához, gondolkodjunk el azon, hogy mennyiben gazdagítja egy ilyen háttér a webhelyünk nyújtotta élményt. Sőt ami talán még fontosabb: gondolkodjunk el azon is, hogy az oldal szövege mennyire lesz könnyen olvasható egy ilyen háttér előtt.

Ha felidézzük az általunk nap mint nap meglátogatott webhelyeket, ráébredhetünk, hogy nem túl sok használ ismétlődő képekből álló mintás, az egész oldalra kiterjedő háttérrel. Ha a böngészőnkkel kizárólag cégek, termékek, sportegyesületek, illetve más, főként információszolgáltató céllal létrejött – azaz szövegközpontú – webhelyekre szoktunk ellátogatni, akkor a mintázott háttérű weboldalak aránya tovább csökken. Bár a webhelyünket olyanná alakítjuk, amilyenné csak akarjuk, ha a cégünk weboldalának elkészítéséről van szó, esetleg érdemes lemondani a színes szöveg mögül rikítóan kitűnő háttérrel.

Ha mégis az egész webhelyre kiterjedően egy ismétlődő képekből álló mintás háttérkép használata mellett döntünk, ne feledjük, hogy az ilyen háttér akkor néz ki a legjobban, ha nem látszik, hogy ismétlődik rajta a kép. Más szóval: akkor lehetünk biztosak abban, hogy megfelelő képet választottunk, ha a kép alja észrevehető átmenet nélkül folytatódik a tetejében, és az egyik oldala a másikkban.

A 10.8. és a 10.9. ábra egyaránt mozaikos háttérű webhelyet mutat be. A háttérkép saját magába történő átmenete mindkét esetben észrevehetetlen, az összehatás mégsem ugyanaz.



10.8. ábra

A háttérkép határai nem látszódnak, mégis tudjuk, hogy ismétlődő mintázatot látunk, ugyanis hat egyforma alak alkotja a hátteret



10.9. ábra

A háttérkép határai ezúttal sem látszódnak, és nem is feltűnő, hogy ismétlődik a kép

A 11. órán megtanuljuk majd, hogyan helyezhetjük el a háttérképeket tárolóelemekben. A mai órán elmondott minden figyelmeztetés ellenére lesznek olyan helyzetek, amikor a háttérképek a webtervezői eszköztárunk hathatós eszközeivé válnak – de nem egész oldalra kiterjedő háttérként.



Ha mindenképpen mozaikos hátteret szeretnénk, de sehogy sem sikerül a kívánt mintázat kialakítása, keressük fel az Interneten található több száz, szabadon használható háttérképeket ajánló weboldal valamelyikét. Az ilyen webhelyeken található képek – bár profi grafikusok munkái – vagy ingyenesek, vagy igen olcsók.

Animációt tartalmazó webes grafika készítése

A GIF formátum használatával olyan animációk készítésére is lehetőségünk nyílik, amelyekkel bármelyik weboldalt feldobhatjuk. Az animációt tartalmazó GIF képek sokkal rövidebb idő alatt tölthetők le, mint a hasonló célú mozgókép-, illetve multimédiás fájlok. Animált GIF fájlok úgy készíthetők a GIMP segítségével, hogy több réteget is létrehozunk a képen, és a fájl mentésekor módosítjuk az Animated GIF beállításokat. Ha pedig máris megvan az animálni kívánt képsorozat, használjuk a Gickr ingyenes, web alapú, animált GIF fájlok készítésére szánt szolgáltatását (<http://www.gickr.com/>).

A GIF animáció elkészítésének első lépéseként az egymás után megjeleníteni kívánt képeket – vagy rétegeket, a használt alkalmazástól függően – kell elkészítenünk. Az egyes képek lesznek az animáció *képkockái*. Bár az angol szakirodalomban ezekre a képkockákra ugyanúgy a „frame” névvel hivatkozunk, mint a keretekre, amelyekkel a 13. órán fogunk megismerkedni, a kettőnek semmi köze egymáshoz. A képkockákra úgy kell gondolnunk, mint azokra az állóképekre, amelyekből a filmek, illetve rajzfilmek készülnek. A monitoron megjelenő mozgás az egyes önálló képek közötti parányi eltérések eredménye. Ha már kigondoltuk, hogy mi lesz az egyes képeken, az animáció elkészítése viszonylag egyszerű – a tervezés folyamata okozza a legfőbb nehézséget. Szánjunk kis időt arra, hogy a forgatókönyvekhez hasonlóan felvázoljuk a képkockákat, főleg akkor, ha az animáció nem csak néhány képkockából áll majd. Ha már kigondoltuk, hogy a képkockák miként kapcsolódnak egymáshoz, akkor vagy a pár sorral korábban említett Gickr szolgáltatással készíthetjük el az animációt, vagy olvassuk el a grafikai alkalmazásunk leírásának idevágó részét, és ismerkedjünk meg az animációk készítésének módjával.

Összefoglalás

A mai órán a képek webes használatra történő előkészítésének alapjaival ismerkedhettünk meg. Ha mást nem is, azt biztosan megértettük, hogy igen összetett témakörrel van szó, ahhoz pedig biztosan elegendő volt a tananyag, hogy étvágygerjesztőül szolgáljon. A mai óra példáiban a népszerű (és ingyenes!) GIMP alkalmazást használtuk, de természetesen lehetőségünk van egy nekünk jobban tetsző programot választani.

Megtanultunk képet körülvágni, átméretezni, színekorrrekciót végrehajtani rajta, és megismerkedtünk néhány fájlformátummal is. Sok meggondolnivalónk van, ha a weboldalainkat képekkel kívánjuk gazdagítani – többek közt a kép mérete, felbontása, az átlátszóság használata, illetve az animált GIF képek és az ismétlődő elemekből építkező mozaikos hátterek elhelyezése.

Kérdezz-felelek

- K: *Nem okosabb-e ahelyett, hogy ezt a sok mindent megtanulnám, kiadni az oldal tervezési munkálatait egy számítógépes grafikusnak?*
- V: E könyv írójának nincs egyszerű dolga, ha válaszolnia kell a kérdésre. A hezitálásnak pedig nem az az oka, hogy az író maga is egy webfejlesztő és -tervező cégnél dolgozik, és így az áll érdekében, hogy a szakosodott cégek foglalkoztatását javasolja. Ez azonban nem mindig a legjobb megoldás. A feladatot egy grafikusnak kiadni időbe és pénzbe kerül. Ezen felül sok az olyan számítógépes grafikus, aki nem webes célra készíti a grafikáit – gondoljunk csak a nyomtatásra szánt grafikákra, amelyek egészen mások, mint az Internetre készült társaik. Sőt egy magát grafikai tervezőnek vélő egyén nem feltétlenül bír a mi igényeinknek megfelelő grafikai tervező minden képességével. Más szóval, lehet, hogy remekül megtervezi a webhely grafikai elemeit, de *annyira* nem remekel tartalomfejlesztőként, vagy csak kevésbé ért a HTML-hez és a CSS-hez. Ha a webhelyünk egyszerű személyes webhely, akkor a rá költendő pénznek könnyen találunk jobb helyet is. Ha azonban egy céget, egy terméket, egy iskolát, vagy bármi más olyasmit képvisel a webhely, aminek a sikere függ a róla kialakult képtől, megtérülhet, ha időt (és pénzt) szánunk arra, hogy egy hozzáértő webtervezőre bizzuk a teendőket.
- K: *Már készítettem nyomtatásra szánt képeket. A webes célra készülő képek esetében van valami eltérés?*
- V: Van bizony. A webes képeknél sok, a nyomtatott képekkel kapcsolatos szabályt félre kell tennünk. A webes képek esetében az alacsony felbontást szeretjük, míg a nyomtatásra szánt képeknél igyekszünk a lehető legmagasabb felbontással dolgozni. A számítógép monitorán feketére rajzolunk fehérrel, míg a papíron a fehér háttérre kerül a fekete rajz. Ha sok nyomtatott grafikát készítettünk már, akkor igencsak figyelniünk kell arra, hogy ne váljunk a szokásaink rabjává.
- K: *Ha van egy Windows AVI formátumú mozgóképem, tudok belőle GIF formátumú animációt készíteni?*
- V: Igen. Mindössze annyi a dolgunk, hogy egy alkalmas programmal, például az Animation Shoppal nyitjuk meg az AVI fájlt. A program lehetőséget ad a képkockák számának csökkentésére. Ha a készülő fájl méretét elfogadható szinten kívánjuk tartani, elég, ha csak minden harmadik képkockát használjuk a mintavételezéshez. A 12. órán azt is megbeszéljük, hogyan illeszthetjük be egy weboldalba magát az AVI fájlt.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdítjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Lapolvasóval beolvasunk egy lóról készült fényképet. Mekkora legyen a kép, ha egy weboldalon kívánjuk elhelyezni? Milyen fájlformátumban érdemes mentenünk a képet?
2. A cégünk emblémája egy fekete Z betű, mögötte egy vörös körrel. Mekkora méretben érdemes megrajzolnunk, illetve beolvasnunk? Milyen formátumban mentsük a képet, ha a weboldalunkon is használni szeretnénk?
3. Üzlettársunk egy részletgazdag, sötét, erdei lombkorona képét szeretné a cégünk webhelyén háttérként használni. Vastag fehér betűkkel szeretné írni a szöveget. Mit tegyünk?

Válaszok

1. A kép mérete a kép fontosságától függően 100 x 40 képponttól 300 x 120 képpontig változhat. A legalkalmasabb formátum a JPEG, nagyjából 85%-os tömörítési arány mellett. Természetesen az is megoldás, ha csak egy bélyegképet helyezünk el, és arra kattintva nyílik meg az oldalt kitöltő nagyságú kép. A következő órán tanuljuk meg, hogy miként lehet képeken hivatkozást elhelyezni.
2. Egy négyzet alakú embléma számára általában körülbelül 100 x 100 képpontnyi terület szükséges, de egy ennyire egyszerű grafika igen jól tömöríthető, így használhatunk 300 x 300 képpontos vagy akár ennél is nagyobb képet. Figyeljünk arra, hogy a sablonunkban meglegyen a kép számára szükséges hely, mivel ez egy elég nagy négyzet. A képet érdemes színpalettás GIF képként mentenünk, ugyanis nagyon kevés szín található benne.
3. Megtagadjuk a kérését, és többé nem engedjük beleszólni a cég arculattervének alakításába.

Gyakorlatok

- Ha van céges – vagy személyes – fényképalbumunk, keressünk benne olyan képeket, amelyek jól mutatnának a webhelyünkön. Olvassuk vagy olvastassuk be őket lapolvasóval. Az így létrehozott digitális képtár mindig jól jöhet, ha képek kellenek egy weboldalra. Ha digitális kamerával készült fényképeink is vannak, természetesen nem kell a beolvasással bajlódniuk, azonnal hozzáfoghatunk a képek webes használatra történő előkészítéséhez.
- Mielőtt a fontos, céges weboldalunkhoz nyúlnánk, próbálkozzunk a személyes honlapunk megszépítésével. Így lehetőségünk nyílik a GIMP – vagy az általunk választott egyéb alkalmazás – használatának elsajátítására, és a munkahelyünkön már minden úgy fog menni, mint a karikacsapás.



11. ÓRA

Képek használata a webhelyen

A lecke tartalma:

- Képek elhelyezése egy weboldalon
- Leírás mellékelése a képekhez
- Kép szélességének és magasságának megadása
- Képek igazítása
- Képek átalakítása hivatkozássá
- Háttérképek használata
- Képtérképek készítése

A 10. órán megtanultunk olyan digitális képeket keresni, illetve készíteni, amelyeket felhasználhatunk a webhelyünkön. A mai órán láthatjuk, hogy milyen egyszerű az említett képeket elhelyezni a weboldalainkon. Ezen az órán megismerkedünk a HTML nyelv képek elhelyezésére és leírására szolgáló elemeivel, a képek igazításának módjával, és megtanuljuk, hogy miként használhatjuk a képeket hivatkozásként, illetve más tartalomhoz vezető „térkép” gyanánt.



Önálló feladat

Képek előkészítése a webhelyünkön történő használatra

Már most készítsünk elő két-három olyan képet, amelyeket az órán tanultakat követve elhelyezhetünk a saját weboldalainkon. Ha van már kéznél néhány GIF, PNG vagy JPEG formátumú kép (a fájlnevek vége .gif, .png vagy .jpg), akkor használhatjuk azokat. Az előző órán elkészült képek is megfelelnek erre a célra.

Az olyan keresőszolgáltatások, mint a Google, igazi aranybányának bizonyulhatnak, hiszen a segítségükkel számtalan, a keresett témával kapcsolatos oldalhoz juthatunk el. A keresőoldalak segítségével továbbá rengeteg olyan oldalra lelhetünk, amelyek ingyen vagy igen olcsón használható médiagyűjteményeket kínálnak. Ne feledkezzünk meg a Microsoft sok-sok képet és multimédiás fájlt tartalmazó <http://office.microsoft.com/clipart/> webhelyéről sem. Értékes forrás a Google Images (<http://images.google.com/>) és a Flickr (<http://www.flickr.com/>) webhely is. Keressünk olyan Creative Commons felhasználási engedélyű képeket, amelyek a szerző megjelölésével szabadon felhasználhatók.

Képek elhelyezése egy weboldalon



A webkiszolgáló, a webböngésző, és a végfelhasználó szempontjából nincs jelentősége, hogy hol helyezzük el a képeket, amennyiben mi magunk tudjuk, hol vannak, és a HTML-kódban helyes elérési utakat adunk meg.

Ha képet szeretnénk elhelyezni egy weboldalon, akkor az első dolgunk az legyen, hogy a könnyebb átláthatóság végett a képet áthelyezzük a HTML-fájlt tartalmazó könyvtárba vagy egy Images (Képek) nevű mappába. A szövegben ahhoz a részhez érve, ahol a képet látni szeretnénk, helyezzük el az alábbi HTML-elemet.

A `myimage.gif` név helyett természetesen a saját képünk fájlnevét kell megadnunk.

```

```

Ha a weboldalakat tároló könyvtáron (a dokumentumgyökéren) belül létrehozott képkönyvtárba tettük a képet, akkor az alábbi sort kell írunk:

```

```

Az XHTML-weboldalak az `` elemekben mind az `src`, mind az `alt` jellemző használatát megkövetelik. Az `src` jellemző a képfájl megadására szolgál, az `alt` jellemző értékeként pedig a kép leírását adhatjuk meg – ez utóbbira akkor van szükség, ha a felhasználó nem látja a képet. Az `alt` jellemzőről bővebben a *Képek szöveges leírása* című részben olvashatunk.



Az `` elemben megadható a `title` (cím) tulajdonság is, ami szintén a kép szöveges leírását tartalmazza. Az `alt` jellemzőtől eltérően azonban a `title` jellemző célja a képnek egy olyan leírását megadni, amelynek az olvasójáról feltételezzük, hogy látja a képet. Az `alt` jellemző fontosabb célt szolgál, és főként akkor mutatkozik meg a jelentősége, amikor a kép nem látszik – például amikor egy vak felhasználó képernyőolvasó alkalmazással „tekinti meg” a weboldalunkat. Bár a kettő közül csak az `alt` jellemző használata kötelező, mégis, ha a képeink leírása fontos a számunkra, érdemes az `alt` mellett a `title` jellemzőnek is értéket adnunk.

Az `` elem használatát a 11.1. példa szemlélteti. A weboldal tetején – még a szöveget tartalmazó bekezdés előtt – elhelyeztünk egy képet. Ha a böngészőnkben megnyitjuk a 11.1. példában közölt HTML-fájlt, akkor az automatikus betöltés után a monitorunkon a 11.1. ábrán is látható kép jelenik meg.

11.1. példa Kép elhelyezése a weboldalon az `` címke segítségével

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>A Spectacular Yosemite View</title>
  </head>

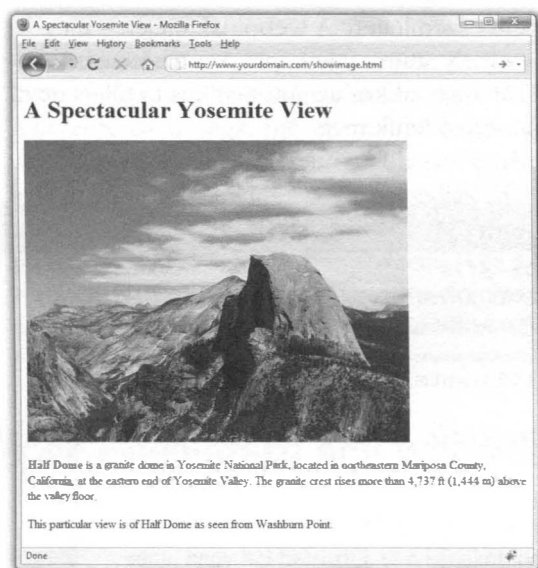
  <body>
    <h1>A Spectacular Yosemite View</h1>
    <p></p>
    <p><strong>Half Dome</strong> is a granite dome in Yosemite National
    Park, located in northeastern Mariposa County, California, at the
    eastern end of Yosemite Valley. The granite crest rises more than
    4,737 ft (1,444 m) above the valley floor.</p>
    <p>This particular view is of Half Dome as seen from Washburn
    ➡ Point.</p>
  </body>
</html>
```



Elvileg bármely webhelyről származó képet be tudunk illeszteni a saját oldalainkba. Minden olyan alkalommal, amikor valaki megnézi az oldalunkat, a kép a másik webhely kiszolgálójáról érkezik. Ezt megtehetjük, de jobb, ha nem tesszük. Nemcsak azért, mert nem szép dolog mások sávszélességét saját célra használni, de az sem kizárt, hogy a weboldalaink így lassabban töltődnek be. Ezen felül pedig semmilyen beleszólásunk nincs a kép megváltoztatásába vagy törlésébe.

Ha engedélyt kapunk arra, hogy egy másik webhelyen megjelent képet elhelyezzünk a saját oldalunkon, mindig töltsük át a kép másolatát a saját számítógépünkre, és az ``

hivatkozás helyett mindig a helyi, `` hivatkozást használjuk. Az előző tanács persze nem vonatkozik azokra az esetekre, amikor a képeinket – például a fényképeinket – olyan külső kiszolgálón tároljuk, amelyet éppen ilyen célra állítottak fel. Ilyen szolgáltatást nyújt például a Flickr (<http://www.flickr.com/>), ahol – az ilyen szolgáltatásoknál megszokott módon – pontos, a szolgáltató webkiszolgálójának címét tartalmazó URL-t is kapunk az egyes képekhez.



11.1. ábra

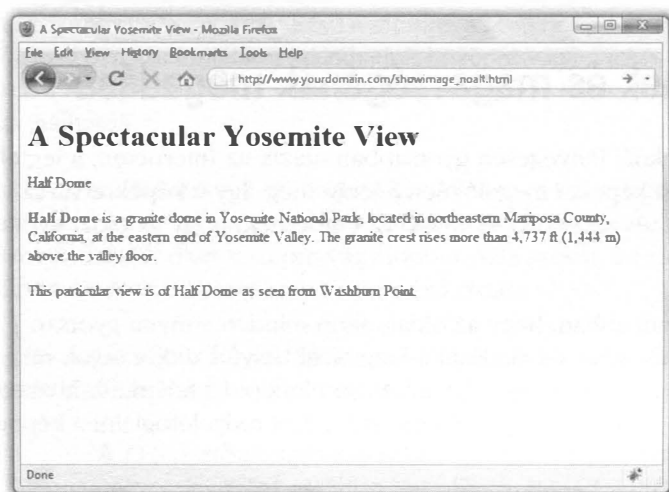
Amikor a webböngésző betölti a 11.1. példában közölt HTML-fájlt, a böngészőablakban a hd.jpg kép jelenik meg

Időközben rájöhettünk, hogy az `img` az *image*, azaz a *kép* szó rövidítése, az `src` pedig a *source*, azaz a *forrás* szövé, és a képfájl helyére hivatkozik. Ahogy a könyv korábbi részében már szó volt róla, a képeket mindig önálló fájlban, a szövegtől elkülönítve tároljuk, hiába tűnik a böngészőben megjelenítve az oldal részének.

Ahogy a hiperhivatkozások megadására szolgáló `<a href>` címkében, az `` címke `src` jellemzőjében is bármilyen teljes internetes cím megadható. A másik lehetőség, hogy ha a kép és a HTML-fájl egy mappában található, csak a fájlnevet adjuk meg. Természetesen viszonyított (relatív) címeket is használhatunk, például: `/images/birdy.jpg` vagy `../smiley.gif`.

Képek szöveges leírása

A 11.1. példa `` eleme tartalmaz egy rövid szöveges leírást is, az `alt="Half Dome"`-ot. Az `alt` szócska az *alternate text*, azaz *helyettesítő szöveg* kifejezés rövidítése. Ez a szöveg akkor jelenik meg, ha a kép nem töltődik be. A kép be nem töltődésének oka lehet egy hibás cím, de a felhasználó is kikapcsolhatta a böngésző beállításai között a képek automatikus betöltését – és az is lehet, hogy a kép az internetkapcsolat lassúsága miatt nem érkezett még meg a gépére. A 11.2. ábrán a kép helyén megjelenő, az `alt` jellemzőben tárolt szöveget figyelhetjük meg.



11.2. ábra

A felhasználók az alt jellemzőben tárolt szöveget látják, ha a kép nem töltődik be

Ha az egérmutatót a kép fölé visszük, az `alt` jellemző szövegét általában még olyankor is láthatjuk egy kis mezőben (a mező neve *eszközeírás*, angolul *tooltip*), ha a kép teljesen betöltődött, és megjelent a böngészőnkben. Az `alt` jellemző szövege segít a rosszul látó vagy más okból képernyőolvasót használó felhasználóknak is.

A weboldalaink minden képéhez meg kell adnunk az `alt` jellemző szövegét, mégpedig arra a sokféle lehetséges helyzetre gondolva, amikor ez a szöveg a felhasználók szeme elé kerülhet. Általában az a legokosabb, ha röviden leírjuk a képet, a weboldalak készítői azonban sokszor hirdetést vagy humoros gondolatokat helyeznek el a kép `alt` jellemzőjében – bár a túl sok, az információ helyét elfoglaló humorizálás esetleg nem nyeri el mindenkinek a tetszését. A kicsi vagy jelentőséggel nem bíró képek esetében nagy a kísértés a helyettesítő szöveg elhagyására, de az `` elembe kötelező

az `alt` jellemzőt megadni. Ha elhagyjuk, attól még az oldal hibátlanul megjelenik, de megsértjük az érvényben lévő XHTML-szabványt. A szerzők így azt tanácsolják, hogy ha tényleg nem akarunk semmit beleírni – ilyen lehet például a kis díszítőelemek képeinek esete –, akkor adjuk meg a jellemzőt üresen hagyva (`alt=""`).

Az `` elemben a `title` jellemzőt nem kötelező megadni, holott a szerepe az `alt` jellemzőhöz igen hasonló. Sőt ha mindkét jellemzőnek van értéke, akkor az eszközeírásban a `title` jellemző értéke jelenik meg. Ennek a ténynek az ismeretében a képek szöveges leírásakor a legokosabb mind az `alt`, mind a `title` jellemzőnek értéket adnunk – valami olyasmit, ami szerintünk abban az esetben is megfelelő, ha az eszközeírásban olvassák el, és akkor is megállja a helyét, ha képernyőolvasóval olvassák az oldalunkat.

A kép szélességének és magasságának megadása

Mint ahogy a szöveg a képeknél lényegesen gyorsabban utazik az Interneten, a legtöbb webböngésző a szöveget a képeket megelőzően jeleníti meg. Így a képekre várva van mit olvasnia a látogatónak, és az az érzése támad az embernek, hogy az oldal gyorsabban töltődik be.

Ha biztosak szeretnénk lenni abban, hogy az oldalunkon minden annyira gyorsan jelenik meg, amennyire csak lehet, és ráadásul a megfelelő helyen, akkor adjuk meg az egyes képek magasságát és szélességét. Így a böngésző az oldal felépítése közben és a kép betöltődésére várva azonnal a megfelelő méretű helyet tudja lefoglalni a képnek.

A weboldalainkra kerülő egyes képek pontos, képpontban kifejezett szélességét és magasságát a grafikai alkalmazásunk segítségével tudhatjuk meg. Az is elképzelhető, hogy az említett tulajdonságokat az operációs rendszerünk eszközeivel is kideríthetjük. A Windowsban például a kép szélességét és magasságát úgy állapíthatjuk meg, hogy az egér jobb gombjával a képre kattintunk, az így megnyíló menüből a Properties (Tulajdonságok) pontot, majd onnan a Details (Részletek) lehetőséget választjuk. Ha már ismerjük a kép szélességét és magasságát, az adatokat az alábbiak szerint helyezhetjük el az `` elemben:

```

```

A `width` jellemző a kép szélességét, a `height` pedig a kép magasságát tárolja.



A kép számára megadott szélesség- és magasságértéknek nem kell megegyeznie a kép valós szélességével, illetve magasságával. A böngésző a képet az általunk megadott méretűvé fogja kicsinyíteni, illetve nagyítani. Mindazonáltal rendszerint nem okos dolog a böngészők képméretezési képességeire alapoznunk, mivel ezen a téren nem kifejezetten jók. Ha egy képet kisebb méretben kívánunk megjeleníteni, akkor biztosan jobban járunk egy képszerkesztő program használatával.

Képek igazítása

Ahogy a szöveg oldalon belüli igazítását meg tudtuk oldani, a képek oldalon belüli igazítására is megvannak a megfelelő eszközeink. Nem csak vízszintes igazításról van szó; a képet függőlegesen is hozzáigazíthatjuk az őt körülvevő szöveghez, illetve további képekhez.

Képek vízszintes igazítása

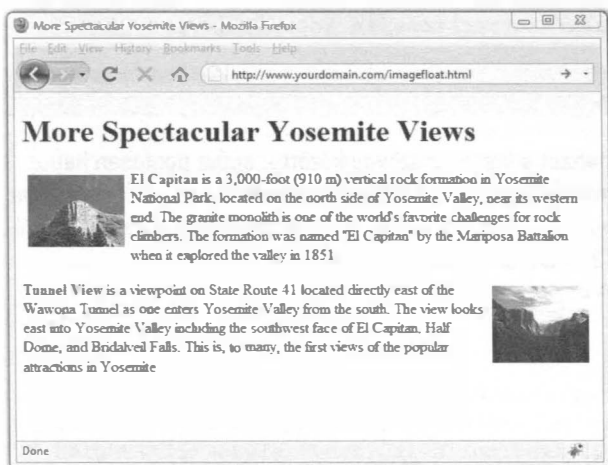
Ahogy az 5. órán megbeszéltük, egy weboldal elemeit a `<div style="text-align:center">`, a `<div style="text-align:right">` és a `<div style="text-align:left">` címkékkel igazíthatjuk középre vagy a jobb, illetve a bal margó mellé. Ezek a stílusbeállítások a szövegre és a képekre is hatással vannak, sőt a `<p>` elemekben is használhatók.

A képek rendszerint – a szöveghez hasonlóan – a bal margó mellé kerülnek, hacsak a `style="text-align:center"`, illetve a `style="text-align:right"` tulajdonsággal nem rendezzük őket középre vagy jobbra. Más szóval, a `text-align` stílusulajdonság alapértelmezett értéke a `left`, azaz a bal oldal.

A képeket körbe is folyathatjuk a szöveggel, ha a `float` (úsztatás) stílusulajdonságot közvetlenül az `` elembe adjuk meg.



A `float` stílusulajdonság sokkal több mindenre jó, mint ami ezúttal kiderült róla, sőt nem csak a képek esetében használható. A `float` ötletes használatával igen érdekes oldalelrendezések alakíthatók ki – mint az a könyv későbbi részében majd kiderül.



11.3. ábra

A 11.2. példában használt képigazítás eredménye

A 11.2. példában az `` címke az első képet balra rendezi, és engedélyezi, hogy a szöveg körbevelhesse a képet. Ehhez hasonlóan az `` címke jobbra rendezi a második képet. A szöveg ezúttal is körbeveszi a képet, csak most balról. A 11.3. ábrán a két képet tartalmazó weboldal látható. A középre igazított képeket nem szokás szöveggel körbevenni, mivel ilyenkor a szöveg elhelyezése kiszámíthatatlanná válik.

11.2. példa *Képek elrendezése a weboldalon a text-align stílus tulajdonsággal*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>More Spectacular Yosemite Views</title>
  </head>

  <body>
    <h1>More Spectacular Yosemite Views</h1>
    <p><strong>El
    Capitan</strong> is a 3,000-foot (910 m) vertical rock formation
    in Yosemite National Park, located on the north side of Yosemite
    Valley, near its western end. The granite monolith is one of the
    world's favorite challenges for rock climbers. The formation was
    named "El Capitan" by the Mariposa Battalion when it explored the
    valley in 1851.</p>
    <p><strong>Tunnel
    View</strong> is a viewpoint on State Route 41 located directly east
    of the Wawona Tunnel as one enters Yosemite Valley from the south.
    The view looks east into Yosemite Valley including the southwest face
    of El Capitan, Half Dome, and Bridalveil Falls. This is, to many, the
    first views of the popular attractions in Yosemite.</p>
  </body>
</html>
```



Jól látható egy kis üres terület a kép és a szöveg között – egész pontosan hat képpontnyi a kép minden oldalán. Ezt – a 11.2. példa mindkét `` címkéjébe beírt – belső margó (`padding`) alkalmazásával értük el, amelynek a használatáról részletesebben majd a 13. órán tanulunk.

Képek függőleges igazítása

Van úgy, hogy egy szöveget tartalmazó sor közepére szeretnénk egy kicsi képet beszúrni, vagy egyetlen sor szöveget szeretnénk egy ábra mellett felíratként elhelyezni. Mindkét esetben jó szolgálatot tesz az az eszköz, amellyel a szöveg és a kép függőleges helyzetét szabhatjuk meg. A kép aljával kerüljön egy magasságba a szöveg alja? Vagy a szöveg közepénél legyen a kép közepe is? Az említett kettőn kívül még további lehetőségek is adódnak:



A `vertical-align` stílusulajdonság további lehetséges értékei a `top` (fent), és a `bottom` (lent). A segítségükkel a képek a sorban lévő szöveget figyelmen kívül hagyva rendezhetők egy adott elemsor legtetejére, illetve legaljára.

- Ha a kép tetejét a vele egy sorban lévő képek, illetve betűk legmagasabbikának tetejéhez szeretnénk igazítani, akkor használjuk az `` címkét.
- Ha a kép alját szeretnénk a szöveg aljához igazítani, az `` címkét érdemes használnunk.
- Ha a kép közepét a vele egy sorban lévő objektumok közepéhez szeretnénk igazítani, akkor az `` címke lesz a segítségünkre.
- Ha a kép alját a szöveg betűvonalához kívánjuk igazítani, az `` címkét használjuk.

A 11.3. példában a fenti négy lehetőség mindegyikét használjuk; az eredmény a 11.4. képen tanulmányozható. A példában négy bélyegképet rendeztünk egymás alá, a képekhez tartozó szöveg pedig a képek bal oldalára került. Az egyes képeket, illetve a hozzájuk tartozó szöveget a `vertical-align` stílusulajdonság különböző értékeivel jelenítettük meg.

11.3. példa Képek szöveghez igazítása a `vertical-align` stílusulajdonságokkal

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Small But Mighty Spectacular Yosemite Views</title>
  </head>

  <body>
    <h1>Small But Mighty Yosemite Views</h1>
    <p><strong>El

```

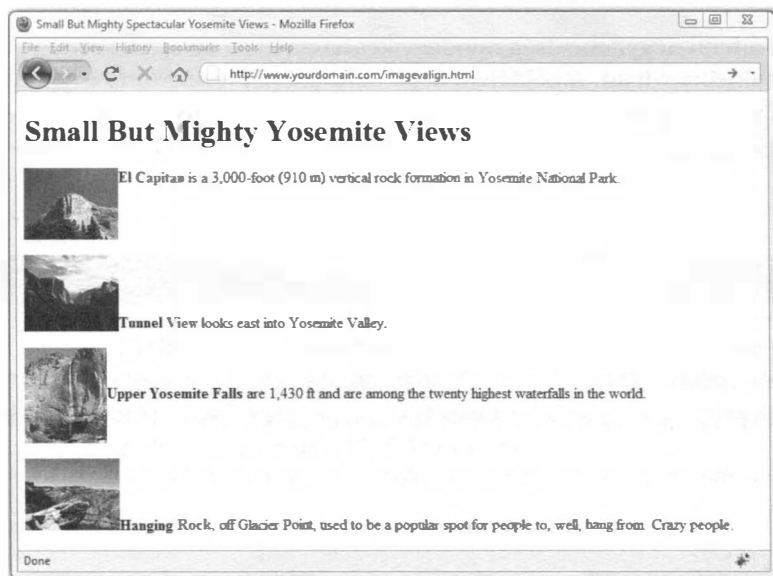
```

Capitan</strong> is a 3,000-foot (910 m) vertical rock formation
in Yosemite National Park.</p>
<p><strong>Tunnel
View</strong> looks east into Yosemite Valley.</p>
<p><strong>Upper
Yosemite Falls</strong> are 1,430 ft and are among the twenty highest
waterfalls in the world. </p>
<p><strong>Hanging
Rock</strong>, off Glacier Point, used to be a popular spot for
people to, well, hang from. Crazy people.</p>
</body>
</html>

```



Ha az `` elemben semmilyen `align` jellemzőt nem adunk meg, a kép a mellette lévő szöveg betűvonalához igazodik. Ez annyit tesz, hogy a `style="vertical-align: baseline"` tulajdonságot soha nem kell begépelünk, hiszen alapértelmezés szerint is ez a tulajdonság értéke. Ha azonban egy képnél margót állítunk be, és azt szeretnénk, hogy a kép pontosabban igazodjon a szöveghez, akkor érdemes a `vertical-align` jellemzőnek a `bottom` értéket adnunk. A 11.4. ábrán megfigyelhető, hogy a szöveg a margó miatt egy kevésbé a kép alá kerül – ez annak az eredménye, hogy a `vertical-align` jellemző értéke `baseline` maradt.



11.4. ábra

A 11.3. példában használt függőleges képelrendezési lehetőségek

Önálló feladat

Kísérletezzünk a képek igazításával!

Helyezzünk el néhány képfájlt a webhelyünk oldalain, és kísérletezzünk a `text-align`, a `vertical-align` és a `float` tulajdonság különféle értékeivel. Indulásként íme egy gyors áttekintés egy halat ábrázoló képzeletbeli kép (`fish.jpg`) elhelyezéséről egy weboldalon:

1. A `fish.jpg` képet másoljuk abba a könyvtárba, ahol a HTML-fájl is található, vagy hagyjuk a jelenlegi helyén, és jegyezzük meg, hogy hol hagytuk.
2. Találjuk ki, hogy hol legyen a kép, és egy szövegszerkesztővel írjuk az `` kódot a HTML-fájl megfelelő helyére.
3. Ha a képet középre szeretnénk igazítani, akkor az `` elem előtt helyezzük el a `<div style="text-align:center">` címkét, mögötte pedig az elem lezárására szolgáló `</div>` kódot. Ha azt is szeretnénk, hogy a szöveg körbevegye a képet, akkor az `` címkét bővítsük a `style="float:right"`, illetve a `style="float:left"` kóddal. Végül, közvetlenül az `` elembe adjuk meg a `vertical-align` stílustulajdonságot, ezzel állítva be a kép helyzetét a többi képhez, illetve a képet körülvevő szöveghez viszonyítva.
4. Ha tudunk még egy kis időt a kísérletezgetésre szánni, akkor próbáljunk több, különféle méretű képet elhelyezni az oldalon, és próbáljuk ki a függőleges igazításhoz használható különféle értékek működését.

Képek átalakítása hivatkozássá

A 11.1. ábrát szemlélve alighanem észrevettük, hogy az oldalon lévő kép meglehetősen nagy méretű, ami a könyv példájánál nem okoz gondot, de ha több képet is el szeretnénk helyezni az oldalon, akkor messze áll az ideálistól. Ilyen esetekben több értelme van kisebb, az adott kép nagyobb változatához vezető hivatkozást tartalmazó bélyegképeket létrehozni. Ha így járunk el, akkor a bélyegképek elrendezhetők úgy, hogy a látogatók az oldal teljes tartalmáról képet alkothassanak, bár egyelőre csak egy-egy kisebb változatot látnak az igazi (nagyobb) képek helyett. A bélyegképek persze csak az egyiket jelentik a sok lehetőség közül – a képeken elhelyezett hivatkozásokkal sokféleképp feldobhatjuk az oldalainkat.

Ha egy képből kattintható, más oldalra vagy képre mutató hivatkozást szeretnénk készíteni, a szöveges hivatkozások létrehozásakor már használt `<a href>` címkét kell használnunk. A 11.4. példa a 11.2. példában lévő kód módosított változatát tartalmazza.

Már az előző változatban is benne voltak a bélyegképek, de a mostaniban a képekre kattintva betöltődik az adott kép nagyobb változata. Szeretnénk a felhasználók tudomására hozni, hogy a bélyegképre kattintva mi fog történni? Akkor munkára fel: a képek és a segítő szövegek kerüljenek egy-egy <div> címkébe. Az eredmény a 11.5. ábrán látható.

11.4. példa *Bélyegképeken elhelyezett hivatkozások használata*

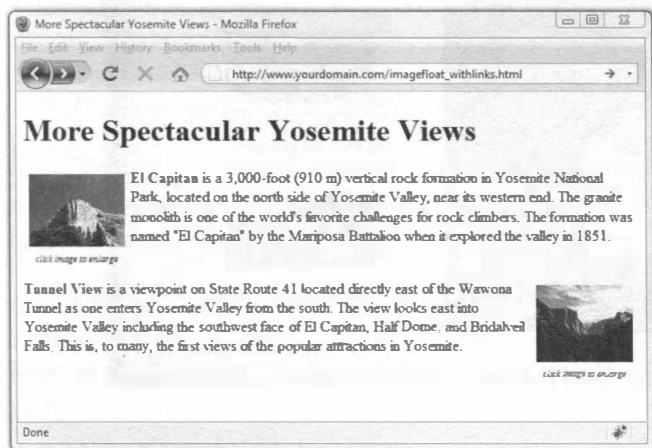
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>More Spectacular Yosemite Views</title>
    <style type="text/css">
      div.imageleft {
        float:left;
        clear: all;
        text-align:center;
        font-size:9px;
        font-style:italic;
      }
      div.imageright {
        float:right;
        clear: all;
        text-align:center;
        font-size:9px;
        font-style:italic;
      }
      img {
        padding: 6px;
        border: none;
      }
    </style>
  </head>
  <body>
    <h1>More Spectacular Yosemite Views</h1>
    <p><div class="imageleft">
      <a href="http://www.flickr.com/photos/nofancyname/614253439/"></a>
      <br/>click image to enlarge</div><strong>El Capitan</strong>
      is a 3,000-foot (910 m) vertical rock formation in Yosemite National
      Park, located on the north side of Yosemite Valley, near its western
      end. The granite monolith is one of the world's favorite challenges
      for rock climbers. The formation was named "El Capitan" by the
      Mariposa Battalion when it explored the valley in 1851.</p>
    <p><div class="imageright">
      <a href="http://www.flickr.com/photos/nofancyname/614287355/"></a>
```

```

<br/>click image to enlarge</div><strong>Tunnel View</strong> is
a viewpoint on State Route 41 located directly east of the Wawona
Tunnel as one enters Yosemite Valley from the south. The view looks
east into Yosemite Valley including the southwest face of El Capitan,
Half Dome, and Bridalveil Falls. This is, to many, the first views of
the popular attractions in Yosemite.</p>
</body>
</html>

```



11.5. ábra

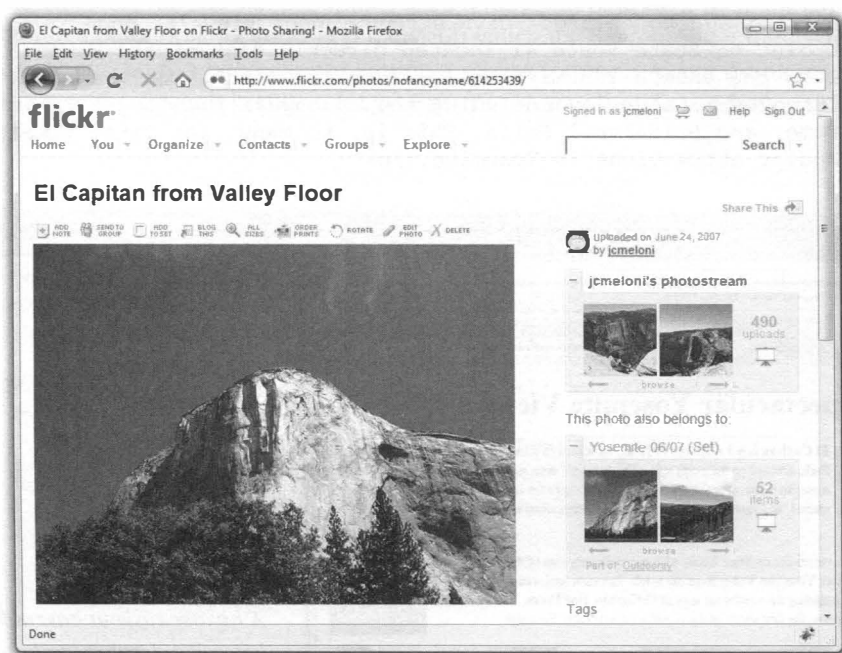
A hivatkozásként használt bélyegképek sokat javítanak a nagy képeket bemutató weboldalak elrendezésén

A 11.4. példában olyan stíuselemeket használunk, amelyeket részletesebben a 14. és a 15. órán ismertetünk majd, de a lényeget alighanem most is látjuk már:

- Az `<a>` címkék felelősek a kis képektől a nagyobb változathoz vezető hivatkozások létrehozásáért. A nagyobb változatot ezúttal külső kiszolgálón – a Flickr oldalán – helyeztük el.
- A `<div>` címkéket a bélyegkép-képfelirat párok elrendezésére, illetve némi belső margó kialakítására használtuk.

Hacsak nem rendelkezünk másként, a webböngésző egy színes téglalapot rajzol a képekből létrehozott hivatkozások köré. Ez a téglalap – a szöveges hivatkozásokhoz hasonlóan – általában kék, ha mostanában nem látogattuk meg a hivatkozás céljaként beállított oldalt; és más színű, ha a stíluslapon mást adtunk meg. Minthogy ritka az olyan alkalom – ha van ilyen egyáltalán –, amikor ezt az idélen vonalat a hivatkozás alakított képeink körül látni szeretnénk, általában érdemes a hivatkozások belsejében szereplő `` címkékben a `style="border:none"` kódot elhelyeznünk. A fenti példában a `border:none` kitétel az `img` elem stíluslap-bejegyzésének része, mivel a stílust kétszer is alkalmazzuk.

Amikor a bemutatott példaoldal bélyegképeire kattintunk, a böngésző a 11.6. ábrán látható módon megnyitja a hivatkozott oldalt.



11.6. ábra

Ha a hivatkozásként használt bélyegképre kattintunk, megnyílik az az oldal, amelyre a hivatkozás mutat

Háttérképek használata

Ahogy az előző órán megtanultuk, a háttérképek a tárolóelemekben egyfajta „tapétaként” használhatók, azaz mind a szöveg, mind a többi kép a háttérkép előtt jelenik meg.

Az alábbiakban ismertetjük a háttér kialakításáért felelős stílustulajdonságokat:

- **background-color:** Az elem háttérszínét adja meg. Bár a képek témaköréhez nincs köze, a háttér kialakításában ez a tulajdonság is szerepet kap.
- **background-image:** Az elem hátteréül használni kívánt képet adhatjuk meg vele. Használatakor az `url('képnév.gif')` utasításformát kell követnünk.
- **background-repeat:** A kép ismétlődésének módját adjuk meg ezzel a tulajdonsággal, méghozzá vízszintes és függőleges irányban is. Alapértelmezés szerint – azaz, ha nem adunk meg semmit – a háttérkép vízszintes és függőleges irányban egyaránt ismétlődik. A további beállítási lehetőségek: `repeat`, azaz ismétlés (a hatása megegyezik az alapértelmezett viselkedéssel), `repeat-x` (vízszintes ismétlés), `repeat-y` (függőleges ismétlés) és `no-repeat` (nincs ismétlés).

- `background-position`: Azt adja meg, hogy a tárolójához képest hová kerül a kép. A lehetséges értékek: `top-left` (bal felső sarok), `top-center` (felül középen), `top-right` (jobb felső sarok), `center-left` (baloldalt középen), `center-center` (középen), `center-right` (jobb oldalt középen), `bottom-left` (bal alsó sarok), `bottom-center` (lent középen), `bottom-right` (jobb alsó sarok), illetve bizonyos képpontban vagy százalékban kifejezett helyzetek.

A háttérkép megadásakor a fenti beállítások egy sorba kerülnek, egyetlen tulajdonságot alkotva. Valahogy így:

```
body {
    background: #ffffff url('képnév.gif') no-repeat top right;
}
```

A fenti stíluslap-bejegyzésben a weboldal `body` eleme fehér lesz, a `képnév.gif` nevű fájl pedig a háttér jobb felső sarkába kerül.

A `background` tulajdonság arra is jó, hogy rendezetlen listákhoz saját felsorolásjelet adjunk meg. Ha képeket szeretnénk felsorolásjelként használni, akkor az első dolgunk, hogy az alábbiak szerint adjuk meg az `` elem stílusát:

```
ul {
    list-style-type: none;
    padding-left: 0;
    margin-left: 0;
}
```

Ez után az alábbi kódot követve adjuk meg a `` elem stílusát:

```
li {
    background: url(mybullet.gif) left center no-repeat
}
```

Bizonyosodjunk meg arról, hogy a `mybullet.gif` fájl – illetve, ha más nevet adunk a felsorolásjelet tartalmazó képnek, akkor az – a webkiszolgálón van, és letölthető. Ha így van, akkor az alapértelmezett teli karika felsorolásjel a megadott képre cserélődik.

A háttérrel kapcsolatos beállításokat megadó tulajdonságokhoz a következő leckék során még visszatérünk, amikor már teljes oldalrendezéseket fogunk CSS-ben megadni.

Képtérképek készítése

Előfordul, hogy túl akarunk lépni az egyszerű, nyomógombos vagy hivatkozásokra épülő, a webhelyeken gyakran látott navigációs megoldáson, és a webhelyünk különböző oldalaira egy kép segítségével szeretnénk eljutni. Például elképzelhető, hogy van egy orvosi információkat tartalmazó webhelyünk, és egy emberi test képéből szeretnénk a különféle testrészekről szóló oldalakhoz hivatkozásokat kialakítani. Vagy lehet egy olyan, világtérképet mutató webhelyünk, ahol a felhasználók az egyes országokkal kapcsolatos tudnivalókat az ország képére kattintva érhetik el. Egy képet feloszthatunk különböző részekre, és az egyes részekkel hivatkozhatunk a megfelelő weboldalakra – attól függően jutunk ide vagy oda, hogy a kép melyik részére kattintunk. Ezt a megoldást nevezik *képtérképnek* (imagemap). Képtérkép bármilyen képből készíthető.



11.7. ábra

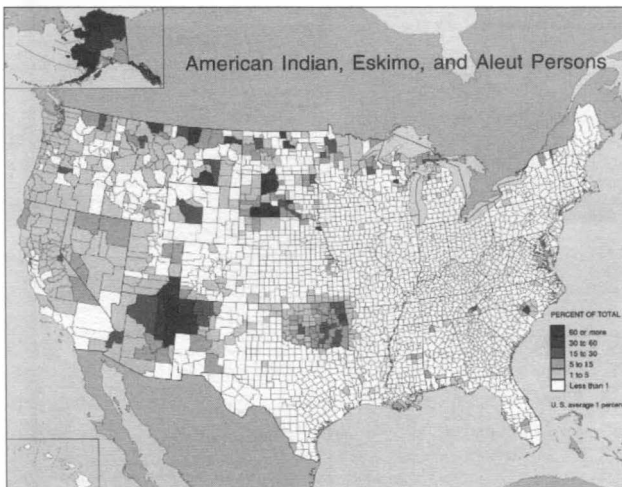
Weboldal tizenkét különféle cég emblémájával – megvalósítható lenne egy tizenkét részes tagolt képtérképpel is

Nem kell mindig képtérkép

A képtérképekről szólva először azt kell elmondanunk, hogy esetleg soha nem kell használnunk képtérképet, kivéve néhány különleges esetet. Majdnem minden esetben egyszerűbb, ha néhány szokványos képet használunk, közvetlenül egymás mellé helyezve és külön-külön hivatkozássá alakítva őket.

Lássuk például a 11.7. ábrát. Ezen a weboldalon tizenkét különféle cég emblémája látható. Az efféle weboldalak az üzleti világban igen gyakoriak – a cégek a partnereiknek kedveskednek az ilyen kis ingyenes hirdetésekkel. Meg *lehetne* tenni, hogy az emblémákat egyetlen nagy képen jelenítjük meg, majd létrehozunk egy-egy hivatkozást a tizenkét céghez, hogy a felhasználók az egyes cégek emblémáira kattintva juthassanak el az adott cég honlapjára. Az is járható út azonban, ha tizenkét önálló kép segítségével jelenítjük meg a cégeket az oldalunkon, mégpedig úgy, hogy minden kép egyben a hozzá tartozó cég honlapjára vezető hivatkozásként is szerepel.

A képtérkép akkor jelenti a legjobb választást, ha egy olyan képünk van, amelyen számos részlet látható, és ezek össze-vissza helyezkednek el a képen, vagy a kép felépítése olyan, hogy túl bonyolult lenne önálló részekre osztani. A 11.8. ábrán egy olyan képet látunk, amely kitűnő kiindulópont képtérkép létrehozásához.



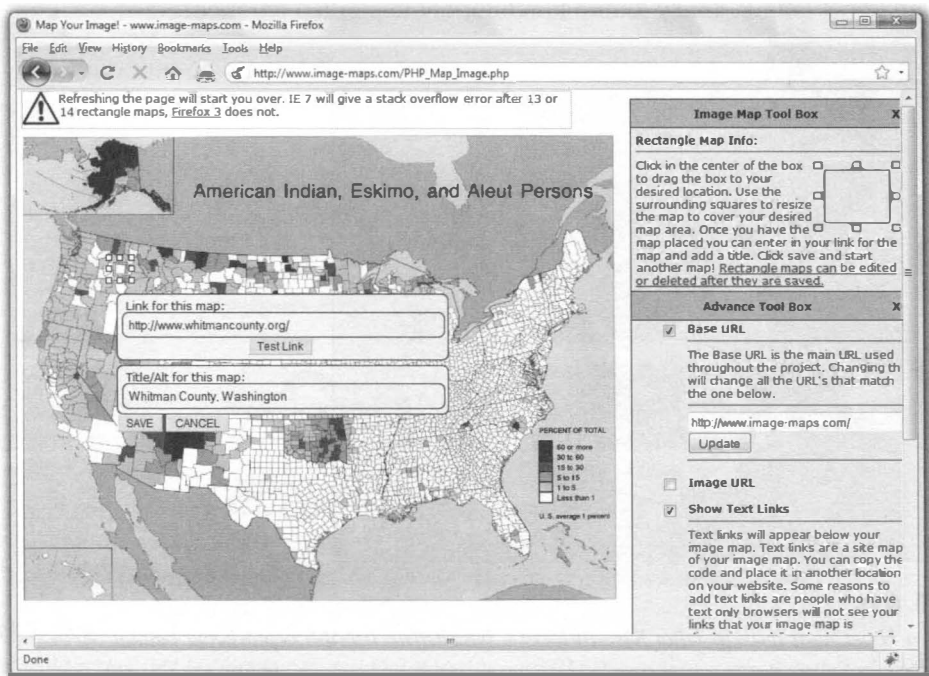
11.8. ábra

Ezt a képet nem igazán lehet felszabdalni – készítsünk inkább képtérképet belőle!

A kép területekre osztása

Ha képtérképet szeretnénk készíteni, az első dolgunk, hogy megkeressük azoknak a részeknek a pontos, képpontban mért koordinátáit, amelyeket kattintható hivatkozássá szeretnénk alakítani. Az efféle kattintható hivatkozásokat nevezzük *területnek* (region). Könnyen lehet, hogy magával a grafikai alkalmazásunkkal is egyszerűen megtaláljuk a szükséges koordinátákat, de használhatunk önálló képtérkép-készítő alkalmazást is. Ilyen program például a Mappedit (<http://www.boutell.com/mappedit/>) vagy a <http://www.image-maps.com/> webhelyen üzemelő internetes képtérképkészítő. Ezek az eszközök – azon túl, hogy a koordináták megtalálásában segítenek – megadják a képtérképeket működtető HTML-kódot is.

A képtérképkészítő eszköz használata általában annyiból áll, hogy az egérrel téglalapot – vagy más alakzatot – rajzolunk a hivatkozássá alakítandó terület köré. A 11.9. ábra a téglalap alakú kijelölések eredményét mutatja, illetve azt a felületet, amelyen megadhatjuk a hivatkozás célját, a képszelet címét és helyettesítő szövegét. A képtérképünk elkészítéséhez tehát az alábbi információkra van szükség: az egyes területek koordinátáira, a cél URL-jére, valamint a hivatkozás címeire és helyettesítő szövegére.



11.9. ábra

Kép egyes területeinek hivatkozássá alakítása képtérképkészítő eszköz segítségével



Önálló feladat

Készítsünk saját képtérképet!

Valószínűleg mélyrehatóbban elsajátítjuk a képtérképkészítés lépéseit, ha gyorsan előhúzzuk valamelyik saját képünket, és az óra folyamán el is készítünk belőle egy képtérképet:

- Ha először csinálunk ilyesmit, akkor a legokosabb, ha egy viszonylag nagyobb képpel kezdjük, méghozzá olyanal, amelyet ránézésre is könnyedén osztunk fel nagyjából téglalap alakú részekre.
- Ha nincs kéznél megfelelő kép, készítsünk egyet a kedvenc grafikai programunkkal. Használhatunk egy egyszerű, pár embert ábrázoló képet is, amelyet a rajta lévő embereknek megfelelően osztunk területekre.
- Próbáljunk ki pár képtérképkészítő eszközt, és döntsük el, hogy melyik tetszik a legjobban. Kezdjük egy olyan önálló alkalmazással, mint a Mapedit (<http://www.boutell.com/mapedit/>), majd nézzük meg az internetes képtérképkészítőt a <http://www.image-maps.com/> webhelyen. Ezeken felül is van még pár lehetőség; a választott internetes keresőnkkel keressünk az „imagemap software” kifejezésre.

A képtérkép HTML-kódjának elkészítése

Ha képtérképkészítőt használunk, akkor azonnal megkapjuk a képtérképhez szükséges HTML-kódot is. Mindazonáltal nem árt, ha megértjük a kód működését, így bármikor ellenőrizhetjük, hogy tényleg jól működik-e. A képtérképeket az alábbi HTML-kódsor vezeti be:

```
<map name="térképnev">
```

Jó, ha tudjuk, hogy bár a `<map>` címke `name` (név) jellemzőjének olyan értéket adunk, amelyet akarunk, hasznos dolog valamilyen, a térképre minél pontosabban utaló nevet megadni. A következő feladatunk, hogy a kép minden területéhez meg kell adnunk egy-egy `<area />` címkét. Íme egy egyszerű `<area />` címke, amelyet a térképből készült képtérkép használ:

```
<area shape="rect" coords="100,136,116,152"
href="http://www.whitmancounty.org/"
alt="Whitman County, WA"
title="Whitman County, WA" />
```

Az `<area />` elemnek öt jellemzője van, és ezeket a képtérképen megadott minden területhez meg kell határoznunk:

- **shape (alak):** Azt jelzi, hogy a terület téglalap (`shape="rect"`), kör (`shape="circle"`) vagy szabálytalan sokszög (`shape="poly"`).
- **coords (koordináták):** Itt adjuk meg az egyes területek koordinátáit, képpontban. Téglalap alakú terület esetén a bal felső csúcs x és y koordinátáját kell megadnunk, amit a jobb alsó csúcs x és y koordinátája követ. Kör esetén előbb a középpont x és y koordinátáját adjuk meg, amit a képpontban mért sugár követ. A sokszögeknél egymás után meg kell adnunk minden csúcs x és y koordinátáját.
- **href (hiperhivatkozás):** Azt az oldalt adjuk meg itt, amelyre a területre kattintva jutunk. Bármilyen olyan cím, illetve fájlnev használható, amelyet egy szokásos `<a href>` címkében is megadhatnánk.
- **alt (helyettesítő szöveg):** Azt a szöveget adjuk meg itt, amelyet a legtöbb böngésző – de a Firefox nem – akkor jelenít meg, amikor a felhasználó a kép fölé viszi az egérmutatót. A szöveg nem túl feltűnő, mégis fontos információ azoknak, akik egyébként nem vennék észre, hogy képtérképpel van dolguk. A Firefox – helyesen – a `title` jellemzőt használja a látható jelzés megjelenítésére az `alt` jellemző mellett. Ez is újabb ok arra, hogy mindkét jellemzőt megadjuk – ahogyan azt az óra korábbi részében kértük.

A képtérkép minden kattintásérzékeny részét külön `<area />` címkével kell megadnunk. Ennek köszönhető, hogy egy képtérkép kódja rendszerint `<area />` címkék sorából áll. Az `<area />` elemek kódjának megadásával a térkép elkészült; már csak annyi a dolgunk, hogy a kódreszletet egy `</map>` címkével lezárjuk.

A képtérkép kialakításának utolsó lépése, hogy a térképet összekötjük magával a képpel. A képet az oldalon egy szokásos `` címkével helyezzük el, de ezúttal egy új `usemap` jellemző is kerül a címkébe. A kód a következőhöz hasonló lesz:

```

```

Amikor a `usemap` jellemzőnek értéket adunk, azt a nevet kell megadnunk, amelyet a `<map>` címke `id` jellemzőjében adtunk meg – ne feledkezzünk meg a kettőskeresztről (`#`). A `style` jellemző segítségével adjuk meg a kép szélességét és magasságát, és itt kapcsoljuk ki a képet körbevevő keretet is – amit persze a saját képtérképeinknél meg is tarthatunk, ha éppen olyan kedvünkben vagyunk.

A 11.5. példa annak az oldalnak a kódját tartalmazza, amelyik megjeleníti a képet, majd pár területet megadva létrehozza rajta a képtérképet.

11.5. példa *A <map> és az <area /> címkékkel kialakított képtérkép*

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Testing an Imagemap</title>
  </head>

  <body>
    <h1>Testing an Imagemap</h1>
    <p style="text-align:center">Click on a logo to go to the
    county's web site.<br/>
    </p>

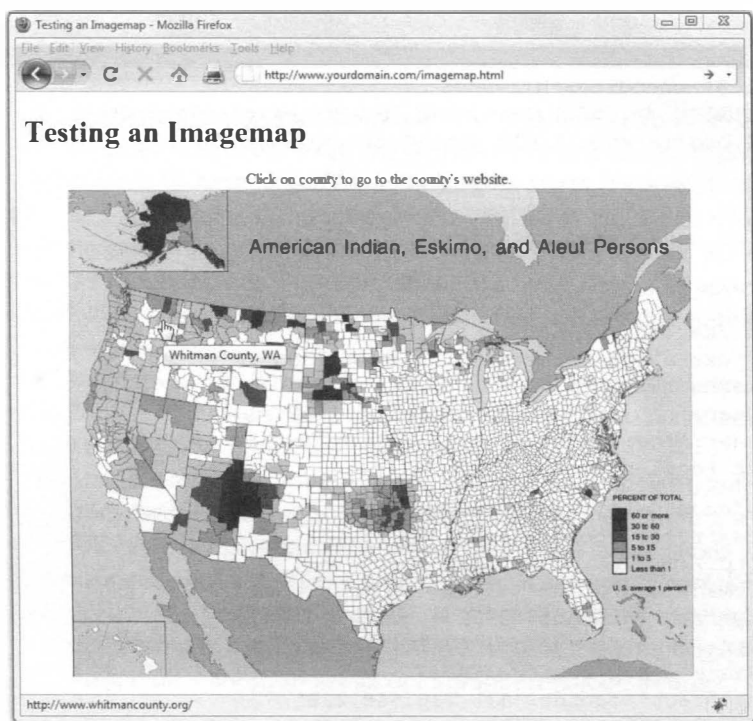
    <map name="countymap" id="countymap">
      <area shape="rect" coords="100,136,116,152"
        href="http://www.whitmancounty.org/"
        alt="Whitman County, WA" title="Whitman County, WA" />
      <area shape="rect" coords="29,271,42,283"
        href="http://www.sccgov.org/" alt="Santa Clara County, CA"
        title="Santa Clara County, CA" />
      <area shape="rect" coords="535,216,548,228"
        href="http://visitingmifflincounty.com/"
        alt="Mifflin County, PA" title="Mifflin County, PA" />
    </map>
  </body>
</html>

```



Ha az Olvasó akkurátus személy, akkor feltűnhetett neki, hogy a kép legtöbb példájától eltérően ez a kód nem az XHTML 1.1, hanem az XHTML 1.0 változatot használja. Nézzük csak meg a kód első soraiban! A változásnak az az oka, hogy pár böngésző – elég, ha mindjárt az Internet Explorert említjük – le van maradva, legalábbis ami az XHTML 1.1 szabvány képtérképekkel kapcsolatos részét illeti. A kérdéses változtatás a `usemap` jellemzővel kapcsolatos, amelynél az XHTML 1.1 nem kéri a kettőskeresztet a térkép azonosítójában – sőt az a helyzet, hogy a kettőskereszt használata az XHTML 1.1-ben sehol sem engedélyezett. Engedélyezett viszont az XHTML 1.0 szerint, és csak úgy tudunk a jelenlegi webböngészők igényeinek megfelelő, mégis érvényesíthető kódot írni, ha ebben a példában megmaradunk az XHTML 1.0 használatánál.

A 11.10. ábrán működés közben látható a képtérkép. Figyeljük meg a kép jobb alsó sarkát. A böngésző – esetünkben a Firefox – megmutatja az egérmutató alatt lévő területen érvényes hivatkozást. Ha pedig az egérmutatót egy terület fölé húzzuk, a területhez tartozó `alt`, illetve `title` jellemzőben tárolt szöveg – esetünkben az, hogy „Whitman County” – jelenik meg a képtérkép előtt.



11.10. ábra

A 11.5. példában megadott képtérkép, ahogy a weboldalon láthatjuk



Képtérképeket nem csak a HTML `<map>` címkéjével hozhatunk létre. A másik módszer kizárólag a CSS használatára támaszkodik, és a 16. órán lesz szó bővebben róla.

Összefoglalás

A mai órán a képek weboldalon való elhelyezését szolgáló `` elemmel ismerkedtünk meg. Megtanultuk, hogyan adható meg olyan rövid szöveges leírás, amely a kép betöltődése alatt jelenik meg a kép helyén, illetve akkor, ha a felhasználó a kép fölé viszi az egérmutatót. Azt is megtanultuk, hogy miként szabályozható az egyes képek vízszintes és függőleges igazítása, és szöveggel körbe is tudjuk már venni a képeket, akár bal, akár jobb oldalról.

Megtanultuk, hogyan lehet a képeket hivatkozássá alakítani – használhatjuk az `<a>` címkét, de létrehozhatunk képtérképeket is. Ezen kívül érintettük a képek tárolóelemek háttérképeként való használatának témakörét is.

A 11.1. táblázat az órán megismert elemek jellemzőit foglalja össze, és tartalmazza a megfelelő stílustulajdonságokat is.

11.1. táblázat A 11. órán megismert HTML-elemek és jellemzőik

Elem/Jellemző	Leírás
<code></code>	Egy képfájl helyez el a weboldalon.
<code><map> ... </map></code>	Egy ügyféloldali képtérképet ad meg. A térképre az <code></code> címkével hivatkozhatunk. A belsejében legalább egy <code><area /></code> címke található.
<code><area /></code>	Egy ügyféloldali képtérképen kattintásérzékeny területet határoz meg.
Jellemzők	
<code>src="cím"</code>	A képet tartalmazó fájl címe.
<code>alt="helyettesítő szöveg"</code>	A kép helyén megjelenő, a kép tartalmával kapcsolatos leírás, elsődlegesen azoknak a felhasználóknak a kedvéért, akik magát a képet nem tudják megtekinteni.
<code>title="cím"</code>	Szöveges, a kép címeként megjelenő leírás, amelyet rendszerint a kép felett felbukkanó kis mezőben – az eszközeleírásban – látunk.
<code>width="szélesség"</code>	A kép szélessége (képpontban).
<code>height="magasság"</code>	A kép magassága (képpontban).
<code>style="border:none"</code>	Ha a képet hivatkozássá alakítjuk, ezzel a jellemzővel szabadulhatunk meg a képet körülvevő kerettől.
<code>style="vertical-align:elrendezés"</code>	A képet függőlegesen igazítja a szöveg tetejéhez (text-top), felülre (top), a szöveg aljához (text-bottom), alulra (bottom), középre (middle) vagy a szöveg betűvonalához (baseline).
<code>style="float:úsztatás iránya"</code>	A képet oldalra úsztatja, hogy körbevehesse a szöveg. A lehetséges értékek: left (balra), right (jobbra) és none (nincs). Az alapértelmezett beállítás a none.
<code>usemap="azonosító"</code>	Az ügyféloldali képtérképként használt, HTML nyelven megadott térkép azonosítója. A <code><map></code> és az <code><area /></code> címkében használjuk.

11.1. táblázat A 11. órán megismert HTML-elemek és jellemzőik

Elem/Jellemző	Leírás
Jellemzők	
shape="érték"	Az <area /> címkében adja meg a kattintásérzékeny terület alakját. A jellemző lehetséges értékei: rect (téglalap), poly (sokszög) és circle (kör).
coords="értékek"	Az <area /> címkében adja meg a kattintásérzékeny terület koordinátáit. Értelmezése és beállítása a terület alakjától függ.
href="hivatkozás URL-je"	Az <area /> címkében adja meg azt az URL-t, amelyik a területre kattintva betöltődik.

Kérdezz-felelek

- K:** Milyen hosszú lehet az címke alt jellemzőjében megadott leírás?
- V:** Elméletileg bármekkora. A gyakorlatban érdemes olyan rövidre fognunk a szöveget, hogy ne foglaljon el nagyobb helyet a képnél. A nagyobb képeknél tíz szó még jó lehet, bár e könyv írója látott már olyan weboldalakot is, ahol a készítők egész bekezdéseket írtak ide. A kis képeknél egyetlen szónál többet ne adjunk meg.
- K:** A könyv útmutatásának megfelelően használtam az címkét, mégis, amikor megnézem az oldalt, mindössze egy kis doboz látszik a kép helyén, benne pedig egy X vagy hasonló alakzat. Mit rontottam el?
- V:** Az említett ikon két dolgot jelenthet: a böngésző nem találja a képet, vagy a kép a böngésző számára ismeretlen formátumú. A probléma megoldásának első lépése, hogy megnézzük, hogy a kép a helyén van-e. Ha igen, akkor nyissuk meg egy grafikai programban, és mentjük újra, mégpedig GIF, JPG vagy PNG formátumban.
- K:** Mi történik, ha egy képtérképen egymást átfedő területeket alakítunk ki?
- V:** Erre megvan a lehetőség. Azt kell fejben tartanunk, hogy amikor a böngésző eldönti, hogy melyik hivatkozást követi, az egyik hivatkozás elsőbbséget fog élvezni a másikkal szemben. Az elsőbbséget az a terület kapja, amelyik hamarabb következik a képtérkép HTML-kódjában. Az első terület például elsőbbséget élvez a másodikkal szemben, azaz, ha valaki az átfedő területre kattint, az első területhez tartozó hivatkozás célja nyílik meg. Ha a képtérképnek van olyan része, ahová nem tettünk hivatkozást – más szóval, „holttere” van –, akkor az átfedés használatával biztosíthatjuk, hogy erre a területre kattintva tényleg ne nyíljon meg semmilyen hivatkozás, mégpedig úgy, hogy a holtter területét a többi terület előtt adjuk meg, úgy, hogy azokkal átfedésbe kerüljön, majd ennek a területnek a href jellemzőjét " " (üres) értékűre állítjuk.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Hogyan illeszthetjük be az `elephant.jpg` képet egy weboldal legtetejére?
2. Oldjuk meg, hogy minden olyan alkalommal jelenjen meg az Elephant szó, amikor a böngésző nem tudja megjeleníteni az `elephant.jpg` képet!
3. Egy 200 x 200 képpontos, `quarters.gif` (negyedek.gif) nevű képből szeretnénk képtérképet készíteni. Azt szeretnénk, hogy ha a felhasználó a kép bal felső negyedébe kattint, akkor a `topleft.html` oldal töltődjön be; ha a jobb felső negyedbe kattint, akkor a `topright.html` fájl jelenjen meg; a bal alsó negyedbe kattintva a `bottomleft.html` oldal nyíljon meg, a jobb alsó negyedbe kattintva pedig a `bottomright.html`. A vázolt képtérkép milyen HTML-kóddal valósítható meg?

Válaszok

1. A képet tartalmazó fájlt másoljuk az oldal HTML-fájljával azonos könyvtárba. A HTML-fájl belsejében közvetlenül a `<body>` címkét követően szúrjuk be az alábbi HTML-kódot:

```
<p></p>
```

2. Használjuk az alábbi HTML-kódot:

```

```

3. Az alábbi képtérképet kell kialakítanunk:

```
<map name="quartersmap" id="quartersmap">
  <area shape="rect" coords="0,0,99,99" href="topleft.html"
    alt="top left" />
  <area shape="rect" coords="100,0,199,99" href="topright.html"
    alt="top right" />
  <area shape="rect" coords="0,100,99,199" href="bottomleft.html"
    alt="bottom left" />
  <area shape="rect" coords="100,100,199,199"
href="bottomright.html"
    alt="bottom right" />
```

```
</map>  

```

Gyakorlatok

- Az órán megismert képelhelyezési módszerek gyakorlása sokat segít abban, hogy felismerjük azt a szerepet, amelyet az általunk készített weboldalakon a képek játszhatnak. Néhány tetszés szerint kiválasztott képpel gyakoroljuk a float stílustulajdonság használatát: helyezzünk egy kép mellé más képeket, illetve szöveget. Ha még emlékszünk, a float tulajdonság lehetséges értékei a következők: left, right és az alapértelmezett none.
- A képek igazítása a webhely keltette benyomás kialakításában is fontos szerepet játszik. Néhány kiválasztott képpel gyakoroljuk a vertical-align stílustulajdonság használatát: helyezzünk egy kép mellé más képeket, illetve szöveget. Ha még emlékszünk, a vertical-align tulajdonság lehetséges értékei a következők: text-top, top, text-bottom, bottom, middle és baseline.

12. ÓRA



Multimédia a webhelyen

A lecke tartalma:

- Hivatkozás multimédiafájlokra
- Multimédiafájlok beágyazása
- További tippek a multimédia használatához

A **multimédia** szó magában foglal mindent, amit egy weboldalon láthatunk vagy hallhatunk: hangot, mozgóképet, animációt csakúgy, mint az állóképeket és a szöveget. Az állóképek és a szöveg használatával az előző órák során már megismerkedtünk, így a mai órán arról tanulunk, hogy miként illeszthetők be egy webhelyre a multimédiás tartalom további típusai. Semmilyen hang, mozgókép vagy animáció létrehozását nem fogjuk tárgyalni – arról lesz szó, hogy miként válhatnak az ilyen fájlok a webhelyünk részévé, legyen szó akár hivatkozásról, akár beágyazásról.

Mielőtt még elgondolkodnánk arról, hogy multimédiás tartalommal gazdagítjuk a webhelyünket, érdemes az eszünkbe idézni, hogy nem minden felhasználónak vannak ilyen tartalom lejátszásához szükséges eszközei, és az ekkora méretű fájlok gyors letöltéséhez szükséges szélessávú internetkapcsolattal sem rendelkezik mindegyikük.

Ha egy hivatkozás multimédiafájltra mutat, mindig figyelmeztessük a látogatókat. Ajánljuk fel nekik a tartalom meghallgatását, illetve megtekintését, de dönthessenek ők arról, hogy meg kívánják-e nyitni az ilyen fájlokat.



Önálló feladat

Hozunk létre vagy keressünk az Interneten a weboldalunkon használható multimédiafájlokat!

Mielőtt megismernénk a multimédia-tartalmak weboldalon való elhelyezésének módját, nem árt, ha kiindulásként rendelkezünk valamilyen használható multimédia-tartalommal.

Multimédia-tartalmat létrehozni – legyen szó a multimédia bármely típusáról – könnyen nagy kihívást jelentő és bonyolult feladatnak bizonyulhat. Ha a tartalmat az alapoktól kívánjuk megalkotni, ennél a könyvnél lényegesen többre lesz szükségünk – nem lesz az ember egyik pillanatról a másikra multimédia-guru. Ha azonban a tartalom maga már készen áll, a mai órán megtanulhatjuk, hogy miként helyezzük el legújabb művünket a weboldalunkon.

Azok, akiknek egy művészi alkotás elkészítése nem napi gyakorlat, próbálkozhatnak más módszerekkel is szert tenni használható multimédiás tartalomra. A nyilvánvaló módszeren kívül – rendeljük meg a munkát egy művésztől – íme néhány lehetőség:

- Az Interneten található tartalom nagy része ingyenes. Természetesen nem rossz gondolat, ha előzőleg azért egyeztetünk a szerzővel, illetőleg a tartalom jelenlegi tulajdonosával, főleg ha nem szeretnénk szerzői jogi pert a nyakunkba. Ezen felül megemlítenénk, hogy az Amerikai Egyesült Államokban sok az olyan kormányhivatal, amelyekre igaz, hogy az általuk létrehozott tartalom az amerikai állampolgárok közös tulajdona (akik ingyenesen használhatnak például minden, a NASA által az Interneten közzétett anyagot.)
- Sok az olyan internetes kereső – például a google.com, a yahoo.com, a lycos.com és mások –, amely rendelkezik multimédiafájlok keresését szolgáló képességekkel is. Amennyiben tiszteletben tartjuk a szerzői jogokat, ezzel a módszerrel könnyen lelhetünk egy adott témához kapcsolódó multimédiás tartalomra. Egy gyors, Flash-animációt, QuickTime-filmrészletet vagy valamilyen hangfájlt célzó keresés több eredménnyel szolgál, mint amennyivel elboldogul az ember.
- Ha ki szeretnénk élni a kreativitásunkat, válasszuk ki a kedvenc médiumunkat – van, akinek ez a videózás, másoknak a hangfelvétel-készítés lesz, megint mások inkább egy animáció készítésével pepecselnek szívesen. Ha meghoztuk a döntést, keressük meg a mestermű elkészítéséhez alkalmas alkalmazást. Sok cég foglalkozik multimédia-tartalom készítésére használatos programokkal, ilyen például az Adobe (<http://www.adobe.com/>) és az Apple (<http://www.apple.com/>), hogy csak kettőt említsünk.

Hivatkozás multimédiafájlokra

Mozgóképet, illetve hangfelvételt a legegyszerűbben és legmegbízhatóbban úgy helyezhetünk el egy weboldalon, ha az `<a href>` címkével létrehozunk egy a fájlra mutató hivatkozást – éppen úgy, ahogy egy másik HTML-fájlra mutatnánk.



Az órán bemutatott, multimédiás tartalom weboldalon történő elhelyezésére szolgáló módszerek egymáshoz igen hasonlóan működnek, a felhasznált tartalom típusától függetlenül.

Az alábbi sorral például egy MOV formátumú, jégkorong-mérkőzést rögzítő felvételt hivatkozhatunk:

```
<a href="hockey.mov">View the hockey video clip.</a>
```

Amikor a látogató a "View the hockey video clip." (A jégkorong-mérkőzés megtekintése) mondatra kattint, a webkiszolgálónkról a `hockey.mov` fájl letöltődik a számítógépére, a letöltés végétével pedig automatikusan elindul a gépre telepített segédalkalmazás vagy bővítmény. Ha nincs megfelelő segédalkalmazás vagy bővítmény a gépen, a böngésző felajánlja a megfelelő bővítmény letöltését, illetve lehetőséget ad arra, hogy a látogató a fájlt a merevlemezére mentse későbbi megtekintés végett.

A 12.1. példa egy olyan weboldal elkészítésére szolgáló kódot tartalmaz, amelyben egy képre kattintva nyitható meg a Windows Media formátumú fájlban tárolt mozgóképek lejátszásához szükséges hivatkozás. A hivatkozást a jobb használhatóság kedvéért a képen kívül a szövegben is megismételjük.

12.1. példa Kép Windows Media mozgóképhez kapcsolása az `<a>` címke segítségével

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Fun in the Pond</title>
  </head>

  <body>
    <h1>Fun in the Pond</h1>
    <p><a href="pond.wmv"></a>
      Michael's backyard pond is not only a fun hobby but also an
      ongoing home improvement project that is both creative and
      relaxing. He has numerous fish in the pond, all Koi from various
      places as far as Japan, Israel, and Australia. Although they
```

```

don't bark, purr, or fetch anything other than food, these fish
are his pets, and good ones at that. You can <a href="pond.wmv">
click here</a> or on the animated graphic on the left
to see a movie clip of some fish in the pond.</p>

```

```

</body>

```

```

</html>

```



Ha az Olvasó számára a *segédalkalmazás* (helper application) kifejezés ismeretlenül cseng, elmondjuk, hogy azokról a külső programokról van szó, amelyeket a webböngésző hív segítségül, amikor olyan fájlokkal találkozik, amelyeket egyébként nem tudna megnyitni. A segédalkalmazások meghívása így jobbára azokhoz a fájltypusokhoz kapcsolódik, amelyeknek a megnyitásával a böngésző egyedül nem boldogul.

A *bővítmények* (plug-in) olyan különleges segédalkalmazások, amelyeket közvetlenül a webböngészőbe telepítünk, és a segítségükkel a multimédiás tartalom megtekintésére közvetlenül a webböngészőből nyílik módunk.

A fenti kód a `projector.gif` animált GIF-fájl felhasználásával hoz létre a `pond.wmv` mozgóképhez vezető hivatkozást. A 12.1. ábrára a medencéről szóló mintaoldal került – látható a filmvetítőt ábrázoló kép. Ha a képre kattintunk, a megnyíló Windows Media Playerben (Médialejátszó) megnézhetjük a `mozgóképfelvételt`.



12.1. ábra

A `projector.gif` nevű, animált GIF formátumú képre kattintva nyílik meg a Windows Media fájlra mutató hivatkozás. A fájl megnyitását külső segédalkalmazás végzi

Ha a felvételt meg szeretnénk tekinteni, mindössze az animált vetítőgépre – esetleg a szövegben lévő hivatkozásra – kell kattintanunk. A művelet eredményeképpen megkezdődik a felvétel lejátszása, mégpedig vagy egy bővítmény segítségével – ha van a böngészőhöz telepítve a felvételt lejátszani képes bővítmény –, vagy egy megfelelő segédalkalmazással.

A böngészőnk másképp kezeli a hivatkozást, ha az a Windows Media formátumú `pond.wmv` fájl helyett a QuickTime formátumú `pond.mov` fájlra mutat. A QuickTime bővítménynek köszönhetően külső program indítása helyett közvetlenül a böngészőablakban tekinthetjük meg a felvételt (lásd a 12.2. ábrát).



12.2. ábra

A képre kattintva a hivatkozott fájl – `pond.mov` – a böngésző QuickTime bővítményének használatával nyílik meg

Alighanem az Olvasó is kitalálta már, hogy a multimédia-fájlokra mutató egyszerű hivatkozás biztosítja leginkább a korábbi programváltozatokkal való együttműködést, mivel így a böngésző felelőssége kitalálni, hogy egy multimédia-felvétel miként játszható le. A dolog hátulütője, hogy a felvétel megjelenését kevésbé tudjuk szabályozni, és szó sincs arról, hogy a felvétel az oldal szerves részét képezhetné.



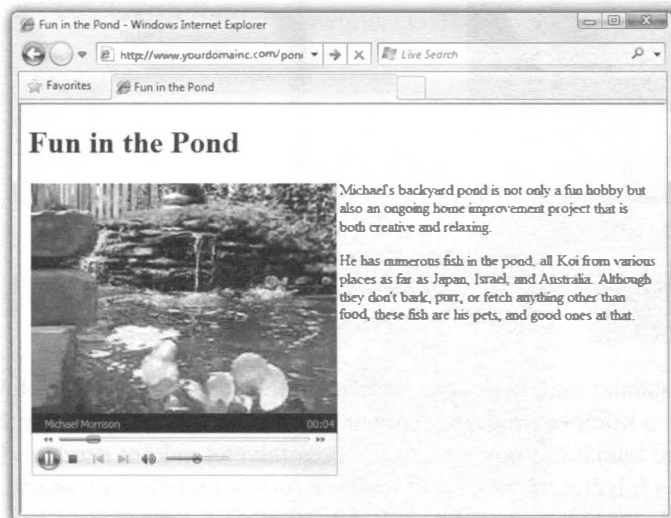
Amennyiben a böngészőnk nem támogatja a QuickTime formátumot, a QuickTime lejátszó ingyenesen letölthető a <http://www.apple.com/quicktime/> webhelyről. A böngészők akkor sem feltétlenül egyformán játsszák le a QuickTime formátumú felvételeket, ha már van QuickTime lejátszó telepítve a gépünkre. A könyv szerzőjének Windowst futtató gépén például az Internet Explorer és a Firefox egyaránt bővítmény használatával, a böngészőablakban játssza le a felvételt, míg az Opera a QuickTime lejátszót segédalkalmazásként indítja el.

Multimédiafájlok beágyazása

Az XHTML szabvány része az `<object>` címke, amelyet mostanra valamennyi, weboldalakba ágyazott multimédia-típusnál használni illene. Ez a címke váltja le azt az `<embed />` címkét, amellyel még mindig találkozhatunk néhány HTML-forráskódban.

Amikor egy multimédiafájl beágyazunk egy weboldalba, létrejön néhány olyan szoftveres vezérlőelem, amely a közvetlen, másodlagos ablak megnyitását, illetve az éppen olvasott oldalról való továbblépést szükségtelenné tevő fájlmegnyitáshoz kell. Az alábbi kód a korábban már látott medencés felvétel beágyazását végzi. A beágyazáshoz pusztán az `<object>` címkét használjuk.

```
<object classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
width="320" height="305">
<param name="type" value="video/x-ms-wmv" />
<param name="URL" value="pond.wmv" />
<param name="uiMode" value="full" />
<param name="autoStart" value="false" />
</object>
```



12.3. ábra

Az `<object>` címke használatával lehetőségünk nyílik egy mozgóképet ágyazni egy weboldalba, mégpedig a használni kívánt lejátszó megjelölésével

Tekintettel arra, hogy teljes egészében elvégzi a felvétel beágyazását az oldalba (lásd a 12.3. ábrát), a kód igazán nem számít szörnyen bonyolultnak. Az oldal legkacifántosabb része az `<object>` címke `classid` (osztályazonosító) jellemzője, azaz az ott olvasható hosszú, betűkből és számokból álló kód. Ez a Windows Media Player alkalmazás egyedi azonosítója; ezzel mondjuk meg az `<object>` címkének, hogy a felvétel lejátszását a Windows Media Player beágyazásával végezze. A fenti kód így módosítás nélkül használható a saját weboldalainkon.

Az `<object>` címke `width` és `height` jellemzője adja meg a beágyazott Windows Media Player alkalmazás ablakának méretét. Ha ezeket a jellemzőket elhagyjuk, akkor pár böngésző a tartalom méretének megfelelően alakítja a beágyazott alkalmazás ablakának méretét, más böngészők ellenben meg sem jelenítik a beágyazott alkalmazást. Ha biztosra akarunk menni, állítsuk a lejátszani kívánt felvétel méretének megfelelő nagyságúra a fenti tulajdonságokat.

Az `<object>` címkepár tagjai között találjuk meg a felvétel lejátszásának további részleteit hordozó négy `<param>` címkét. Minden címkének két jellemzője van, a `name` (név) és a `value` (érték). Ezek rendelkeznek adatot (értéket) egy adott beállításhoz (névhez). Esetünkben a felvétel URL nevű jellemzőjének értéke a `pond.wmv` fájlnev. A harmadik, `uiMode` nevű paraméter azt határozza meg, hogy a Windows Media Player mely gombjai és felületi elemei látszódnak. A `full` érték a felhasználói felület valamennyi elemének – vezérlőgomboknak, hangerőszabályzóknak – a megjelenítését jelenti. Az utolsóként megadott `autoStart` paraméter értéke `false` (hamis), azaz amikor a webböngészőben megnyitjuk az oldalt, a felvétel lejátszása nem indul el automatikusan.

A legfurcsább alighanem a `type` paraméter. Ez a paraméter végzi a megjelenített tartalom formátumának (esetünkben Windows Media Video, azaz WMV) azonosítását. A paraméter értékét az internetes MIME-szabványban foglaltak közül kell kiválasztanunk.

A *MIME-típus* egy olyan azonosító, amely az Interneten előforduló különféle tartalomtípusok azonosítására szolgál. A MIME betűszó a Multipurpose Internet Mail Extensions (többcélú internetes levélkiterjesztések) kifejezés rövidítése, mégpedig azért, mert a MIME-típusok eredetileg az e-mailekhez csatolt mellékletek azonosítására szolgáltak. Az `<object>` címke `type` jellemzőjében is ezeknek a típusoknak a használatával adjuk meg a `data` jellemzőben hivatkozott médiafájl típusát.

Az alábbiakban néhány olyan népszerű hang- és mozgóképformátum MIME-típusát ismertetjük, amelyeket esetleg az Olvasó is szívesen használ a weboldalain.

- WAV hangfájl: `audio/x-wav`
- AU hangfájl: `audio/basic`
- MP3 hangfájl: `audio/mpeg`
- MIDI hangfájl: `audio/midi`
- WMA hangfájl: `audio/x-ms-wma`
- RealAudio hangfájl: `audio/x-pn-realaudio-plugin`
- AVI fájl: `video/x-msvideo`
- WMV fájl: `video/x-ms-wmv`
- MPEG videófájl: `video/mpeg`
- QuickTime fájl: `video/quicktime`

A 12.2. példa az úszómedencés weboldal kódját ismerteti. Lehetőségünk nyílik az `<object>` címke használatát megfigyelni.

12.2. példa WMV formátumú mozgóképek közvetlen beágyazása az <object> címke segítségével

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Fun in the Pond</title>
  </head>

  <body>
    <h1>Fun in the Pond</h1>
    <div style="float:left; padding:3px">
      <object classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
        width="320" height="305">
        <param name="type" value="video/x-ms-wmv" />
        <param name="URL" value="pond.wmv" />
        <param name="uiMode" value="full" />
        <param name="autoStart" value="false" />
        <embed width="320" height="305" type="video/x-ms-wmv"
          src="pond.wmv" controls="All" loop="false" autostart="false"
          pluginspage="http://www.microsoft.com/windows/windowsmedia/" />
      </object>
    </div>
    <p>Michael's backyard pond is not only a fun hobby but also
    an ongoing home improvement project that is both creative and
    relaxing.</p>
    <p>He has numerous fish in the pond, all Koi from various places
    as far as Japan, Israel, and Australia. Although they don't bark,
    purr, or fetch anything other than food, these fish are his pets,
    and good ones at that.</p>
  </body>
</html>

```



Minthogy az XHTML nem támogatja az <embed /> címke használatát, a címke használata megakadályozza az oldalaink érvényesítését. A probléma sajnos nem kerülhető meg – meg kell várunk, amíg a böngészők teljeskörűen támogatni fogják az <object> címke használatát, vagy átállhatunk a HTML 5 <embed /> elemére.

Észrevehettünk pár újabb kódrészletet, amely nem szerepelt az <object> címke használatát bemutató korábbi példánkban. Sajnos, ahogy arra már egy korábbi lecke alkalmával kitértünk, nem minden webböngésző valósítja meg az <object> elem támogatását. Ezen oknál fogva válik szükségessé az <object> címkén belül az <embed /> használata: így tudjuk figyelembe venni a böngészők közötti eltéréseket. A megoldás nem ideális, de amíg a böngészőgyártók folyamatos lemaradásban vannak az uralkodó szabványokhoz képest, nincs más eszközünk. Igazán figyelmesen olvasva a kódot, azt is észrevehetjük, hogy az <embed /> az <object> elembe hordozottal megegyező információt hordoz.

Az <object> címke használata némileg összetettebb annál, mint amire a fenti bekezdésekben fény derült. Szerencsére, ha csak multimédia-tartalom lejátszása a cél, nem szükséges az <object> elem bonyolultabb felhasználási módjaival is tisztában lennünk. Más szóval: nem kell multimédia-guruvá lennünk, ha csak néhány hang- vagy mozgóképfelvételt szeretnénk a weboldalainkon megosztani.



Az <object>, illetve az <embed /> címke használatával nem kizárólag mozgóképet tároló médiafájlokat helyezhetünk el egy weboldalon. Valamennyi multimédia-fájl esetében az ismertetett eljárást kell követnünk. Ha meg szeretnénk állapítani a pontos osztályazonosítót és a kódalapot (codebase), valamint a <param /> elemek kitöltéséhez szükséges adatokat, akkor érdemes valamelyik keresővel rákéresnünk az **object embed tartalomtípus** kifejezésre – a **tartalomtípus** szót a Real Audio, a QuickTime, a Flash, illetve a használni kívánt típus nevével helyettesítve.

További ötletek a multimédia használatához

Mielőtt mozgóképet, hangot vagy animációt helyezünk el egy weboldalon, gondoljuk át, hogy valóban szükség van-e rá. Minden alkalommal, amikor az említett típusokba tartozó multimédiás tartalmat szeretnénk használni, győződjünk meg róla, hogy alapos okunk van rá. A szükségtelenül használt hang és videó – a felesleges képekhez hasonlóan – eltereli a figyelmet az oldal valódi üzenetéről. Természetesen, ha ez az üzenet az, hogy „Nézzétek, ezt én filmeztem le!”, vagy az, hogy „Hallgassa meg mindenki a zenémet, töltsétek le a dalaimat!”, akkor mindenképpen érdemes multimédiás tartalommal bővíteni a webhelyünket.

Íme pár további, megfontolásra érdemes gondolat:

- Ne helyezzünk el multimédiás tartalmat úgy a weboldalunkon, hogy a lejátszás az oldal betöltésekor automatikusan elinduljon. Mindig adjuk meg a felhasználónak a lehetőséget a tartalom elindítására és leállítására.
- Ha lehetséges, engedjük, hogy a felhasználó maga választhassa meg a lejátszásra használt eszközt. Ne használjunk olyan multimédiás tartalmat, amelyet csak egy operációs rendszer egyetlen lejátszója képes megnyitni.
- A multimédia-fájlok nagyobbak, mint a szokásos grafikát és szövegfájlokat tároló fájlok. Ez annyit tesz, hogy a webkiszolgálónkon több tárhelyet igényelnek, és a weboldalunk látogatóinak kiszolgálásához nagyobb sávszélességre lesz szükségünk.
- Legyünk tisztában azzal, hogy amennyiben a weboldalunk teljes mértékben a hang-, illetve videó-tartalomra támaszkodik, és csak nagyon korlátozott mennyiségű szöveget és képet tartalmaz, akkor a lehetséges látogatók egy része nem hallja, illetve látja majd a közlendőnket. Ennek oka a webhely látogatóinak

rendelkezésre álló, korlátozott képességű számítógép, illetve az alacsony sáv-szélesség. Az ilyen felhasználók számára nyújtunk más lehetőséget a tartalom megismerésére.

- Élünk a YouTube (<http://www.youtube.com/>) és a hozzá hasonló internetes videómegosztó szolgáltatások nyújtotta lehetőséggel. A YouTube azon felül, hogy tárhelyet biztosít a felvételeinknek, még a felvétel beágyazásához szükséges kódot is megadja. A 12.4. ábrán például a világ legcukibb kutyusának YouTube-oldala látható. Ha az ábrán lévő, Embed (Beágyazás) feliratú részen megadott példakódot átmásoljuk valahová, az alábbi kódsorokat kapjuk:

```
<object width="425" height="344">
<param name="movie"
value="http://www.youtube.com/v/yPxiHd2BOpo&rel=0&color1=0xb1b1b1
&color2=0xcfcfcf&feature=player_profilepage&fs=1"></param>
<param name="allowFullScreen" value="true"></param>
<param name="allowScriptAccess" value="always"></param>
<embed
src="http://www.youtube.com/v/yPxiHd2BOpo&rel=0&color1=0xb1b1b1&c
olor2=0xcfcfcf&feature=player_profilepage&fs=1"
type="application/x-shockwave-flash" allowfullscreen="true"
allowScriptAccess="always" width="425" height="344"></embed>
</object>
```

Ez a kódrészlet már beilleszthető egy weboldalba.



12.4. ábra

A YouTube nem csak tárhelyet biztosít a felvételeinknek, de megadja a weboldalon használható hivatkozások és <object> címkek kódját is

Összefoglalás

A mai órán megtanultuk, hogy miként ágyazhatunk mozgóképet és hangot a weboldalainkba. Megismertük a multimédiaifájllhoz vezető egyszerű hivatkozások használatának módját – ez a multimédiás tartalom lejátszására használható legáltalánosabban támogatott, de legkevésbé rugalmas lehetőség. Tudjuk már azt is, hogy miként ágyazható multimédiás tartalom közvetlenül egy weboldalba az `<object>` címkével, sőt azt is, hogy a minél több böngészővel való együttműködés céljából miként egészítsük ki az `<object>` címkét az `<embed />` címkével. Az `<object>` és az `<embed />` elem rengeteg multimédiás fájltypus beágyazására alkalmas – kezelhető vele a például a WAV, az MP3, a RealAudio és a MIDI, nem beszélve az AVI, a WMV és QuickTime típusról, hogy csak néhányat említsünk.

A 12.1. táblázat a mai órán tárgyalt HTML-elemeket foglalja össze.

12.1. táblázat A 12. órán tárgyalt HTML-elemek és -jellemzők

Elem/jellemző	Leírás
<code><object>...</object></code>	Képet, mozgóképet, Java-kisalkalmazást, ActiveX-vezérlőt és más objektumokat illeszthetünk be vele egy weboldalba.
<code><param>...</ param ></code>	Egy objektum futásidejű beállításait – például az oldalon elfoglalt hely szélességét és magasságát – adhatjuk meg vele.
Jellemzők	
<code>name="név"</code>	Paraméterként megadott tulajdonság neve.
<code>value="érték"</code>	Paraméterként megadott tulajdonsághoz rendelt érték.
<code><embed /></code>	Böngészőbővítmény által beolvasható, illetve megjeleníthető multimédiaifájl beágyazására szolgál. Technikai értelemben elavultnak számít, de máig is hasznos, mivel a böngészők egyelőre nem támogatják teljes körűen az <code><object></code> címkét.
Jellemzők	
<code>width="szélesség"</code>	A beágyazott objektum képpontban mért szélessége.
<code>height="magasság"</code>	A beágyazott objektum képpontban mért magassága.
<code>type="MIME-típus"</code>	A multimédiás tartalom MIME-típusa.
<code>src="tartalom_URL"</code>	A beágyazandó fájl helyét megadó URL.

12.1. táblázat A 12. órán tárgyalt HTML-elemek és -jellemzők

(folytatás)

Elem/jellemző	Leírás
controls="vezérlők"	A médialejátszó alkalmazás felhasználó által használható vezérlői; az all érték hatására minden vezérlő használható.
loop="ismétlés"	Megadja, hogy a médiafájl végéhez érve újra kell-e kezdeni a lejátszást. Lehetséges értékei: true (igaz) és false (hamis).
autostart= "automatikus_indítás"	Megadja, hogy az oldal betöltésekor automatikusan kell-e elindítani a lejátszást. Lehetséges értékei: true (igaz) és false (hamis).
pluginspage= "bővítmény_helyét_megadó_URL"	A médiafájl lejátszására alkalmas bővítmény helyét megadó URL.

Kérdezz-felelek

- K: Sokat hallani az áramló videóról és hangról (streaming video/audio). Mit takarnak ezek a fogalmak?
- V: Nem is olyan rég a hang- és mozgóképfájlok letöltése a legtöbb modemmel percekbe, néha órákba telt, ami komolyan korlátozta a weboldalakba ágyazott mozgókép és hang elterjedését. Manapság mindenki az áramló hang és videó felé mozdul el, ugyanis ezek lejátszása az adatok letöltése közben is megoldható. Más szóval: az ilyen fájlokat nem kell a lejátszásukat megelőzően teljes egészükben letöltenünk.
- Az áramló tartalom lejátszása mostanra széles körben, a legtöbb médialejátszó által támogatott – legyen szó akár önálló alkalmazásokról, akár bővítményekről. Az <object> címkével beágyazott tartalmat a háttérben működő lejátszó – ha képes rá – igyekszik adatfolyamként lejátszani.
- K: Mi alapján válasszak audiovizuális formátumot a QuickTime, a Windows AVI, a Windows WAV, a RealVideo, a RealAudio és az MPEG formátumok közül? Van-e közöttük jelentős különbség?
- V: A QuickTime – bár létezik lejátszó Windowsra is – a Macintosh-felhasználók körében a legnépszerűbb. Hasonló a helyzet az AVI és a WMV, illetve a WAV és WMA formátumokkal: ezek a Windows-felhasználók számára készültek, de Macintosh-on is találunk olyan lejátszót, amely támogatja őket. Az MPEG szintén népszerű hang- és mozgókép-tárolási szabvány. Az MPEG-3 (MP3) mostanra különös népszerűsége tett szert a használható hi-fi szabványok köréből. Szintén az MPEG-szabványon alapul az Apple cég AAC formátuma, amely az Olvasó számára esetleg inkább, mint az iTunes médialejátszó-alkalmazás saját zenei formátuma ismert.

Ha a leendő látogatóink többnyire Windowst használnak, akkor a mozgóképeket érdemes AVI, illetve WMV, a hangfelvételeket pedig WAV, WMA, illetve MP3 formátumban tárolnunk. Ha a látogatók között sok a Macintosh-felhasználó, akkor használjuk a QuickTime, a RealVideo, illetve a RealAudio formátumot, vagy legalább adjuk meg a választás lehetőségét. Az MP3 formátum szintén használható Macintosh-on. Ha az operációs rendszerek közötti átjárhatóság létfontosságú, érdemes megfontolnunk, hogy ne tegyük-e a weboldalunkon általánossá az MP3, illetve a RealVideo és RealAudio formátum használatát – bár ez utóbbiak megtekintéséhez és meghallgatásához csak a <http://www.real.com/player/> weboldarról letölthető alkalmazás használható.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Mi az a legegyszerűbb módszer, amellyel a webhelyünkön a lehető legtöbb látogató számára elérhető mozgóképet tehetünk közzé?
2. Mi az a HTML-kód, amellyel úgy valósítható meg a `myvideo.avi` nevű fájl beágyazása egy weboldalba, hogy minden jelentősebb böngészőt használó látogató meg tudja nézni? A felvétel egy 320 képpont széles és 305 képpont magas területet foglal el a képernyőn.
3. Milyen beállításokat kell megadnunk az `<object>` elemen belüli `<param>` címkében, ha azt szeretnénk, hogy a weboldal a beágyazott felvételt újra meg újra lejátszsa?

Válaszok

1. Egyszerűen létrehozuk a rá mutató hivatkozást:

```
<a href="myvideo.avi">my video</a>
```

2. Minthogy a felvétel a Microsoft cég AVI formátumában áll rendelkezésünkre, a Windows Media Playert érdemes az oldalba illeszteni, mégpedig az alábbi HTML-kóddal:

```
<object classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
width="320" height="305">
<param name="type" value="video/x-ms-avi" />
<param name="URL" value="myvideo.avi" />
```

```
<param name="uiMode" value="full" />
<param name="autoStart" value="false" />
<embed width="320" height="305" type="video/x-ms-avi"
src="myvideo.avi" controls="All" loop="false" autostart="false"
pluginspage="http://www.microsoft.com/windows/windowsmedia/">
</embed>
</object>
```

3. `<param name="loop" value="true" />`

Gyakorlatok

- Próbáljunk meg saját kezűleg elkészíteni egy felvételt, majd a kész művet ágyazzuk be egy weboldalba, vagy keressünk ingyenesen elérhető felvételeket a Weben, és gyakoroljuk a beágyazott objektumok elhelyezését valamilyen szöveget tartalmazó oldalon.
- A mai órán tanult, a médiafájlok beágyazását segítő címkék, illetve módszerek a Flash formátumú fájlok esetében is működnek. Látogassunk el a Flash honlapjára, azaz a <http://www.adobe.com/products/flash/> weboldalra, és ismerkedjünk meg a weboldalak interaktív animációkkal való bővítésével kapcsolatos lehetőségekkel.



13. ÓRA

Keretek használata

A lecke tartalma:

- Keretváz kialakítása
- Hivatkozások keretek és ablakok között
- Belső keretek használata

Talán jártunk már olyan webhelyen, ahol a böngészőablakon belül látszólag több különböző oldal jelent meg. Az a helyzet, hogy a böngészőablakon belül ilyenkor valóban több különböző oldal jelenik meg egyszerre, még hozzá úgy, hogy a böngészőablak több, különböző weboldalt megjeleníteni képes területre oszlik. Ezek a különálló területek keret (frame) néven ismeretesek. A felhasználó nézőpontjából természetesen az önálló keretek egységes webes tartalmat hordozó ablakká formálódnak, de a háttérben önálló oldalak működnek.

Mik azok a keretek?

A *keret* olyan téglalap alakú terület a böngészőablak belsejében, amely képes egy weboldalt más weboldalakat megjelenítő keretek mellett megjeleníteni. A 13.1. ábra első pillantásra talán úgy néz ki, mint egy szokványos weboldal, pedig valójában két különálló HTML-oldalt tartalmaz, amelyek azonban ugyanabban a böngészőablakban jelennek meg. Mindkét oldal a saját keretén belül látható. A keretek egymás felett kaptak helyet, és egy vízszintes sáv választja el őket.



13.1. ábra

Keretek segítségével egyszerre több weboldalt is megjeleníthetünk egy böngészőablakban



13.2. ábra

A Products felíratra kattintva az alsó oldal lecserélődik, de a felső keret változatlan marad

Ha egy weboldal kereteket használ, akkor a keretek rendszerint arra szolgálnak, hogy egy állandóan jelen lévő, a webhely különböző részeire mutató hivatkozásokat tartalmazó menüt jelenítsünk meg velük – ilyen menü látható a 13.1. ábra felső részén. Amikor a példában szereplő valamelyik hivatkozásra kattintunk, a felső keret változatlan marad – az új oldal betöltésére és megjelenítésére az alsó keret szolgál (lásd a 13.2. ábrát).

Nem árt, ha az Olvasó tudja, hogy a keretek használata régóta a webtervezés sokat vitatott és bosszantó problémái közé tartozik. A keretek használatával járó előnyök sosem ellensúlyozták a fellépő hátrányokat. Ugyanakkor amiatt, hogy a böngészők eltérően kezelték a HTML- és CSS-szabványokat, sokáig minden hátrányuk ellenére keretekkel valósítottunk meg bizonyos célokat. A könyv szerzői webfejlesztőként az alábbi okok miatt nem javasolják a keretek használatát:

- A keretek nem elégítik ki a Világhálónak azt az alapelvét, miszerint a webes tartalom egyes részei olyan egységes hiperszöveggé szerveződnek, amelynek a részei között egyetlen webcímből – URL-ből – álló hivatkozások jelentik az összeköttetést.
- A keretekre épülő oldalakat igen nehéz kinyomtatni. Csak a keretváz látszik a nyomtatásban, kivéve ha a kinyomtatni kívánt keretre kattintunk, és a helyi menüből – már ha van ilyen – a „Keret nyomtatása” menüpontot választjuk.
- Ha egy keret kódolása hibás, vagy ha hibátlan ugyan, de aljas szándékkal készült, akkor a felhasználó bennragadhat az oldalon, elzárva a kereten kívül eső tartalom megtekintésének lehetőségétől.
- A kereteket a Web történetében mindig a szabványos, profi és könnyen feldolgozható oldalakat eredményező webfejlesztési és -tervezési módszerek helyett használták. Semmi okunk arra, hogy a jobb, CSS-elrendezésre épülő módszer helyett a kevésbé kifinomult módszert válasszunk.
- A fent említett – és más egyéb – okok miatt a HTML 5 szabványból törölték a kereteket. Az elavult `<frame />`, `<frameset>` és `<noframes>` címkék a jövőben egyszerűen megszűnnek létezni.

A fenti hátrányok ellenére a mai órán létrehozunk egy keretekre épülő, nagyon egyszerű webhelyet. Igen valószínű ugyanis, hogy találkozunk még keretekre épülő oldalakkal, és jól jöhet, ha tudunk hasonló kinézetű és felhasználói élményt nyújtó oldalt készíteni, de keretek használata nélkül. Ha ilyen feladatunk akad, fontos tisztában lennünk a keretek kialakításának módjával – így sikeresebben szabadulhatunk meg tőlük. Ezen kívül megismerkedünk egy olyan kerettípussal – az ilyen keret az `<iframe>` címkével hozható létre –, amely igen fontos célt szolgál, és a HTML 5-ben is fennmarad.

A könyv későbbi részében megtanuljuk, hogy ugyanilyen működés miként oldható meg az XHTML és a CSS segítségével. Addig is – a fejezet anyagát követve – készítsük el az egyszerű, keretekre épülő webhelyünket, hogy megtanuljuk, miként lehet az ilyen webhelyeket az alkotóelemeikre szedni, hogy más módszerekkel újra összerakhassuk azokat.

Keretváz kialakítása

Ebben a részben azt mutatjuk be, hogy miként készíthető el a 13.1. és a 13.2. ábrán is látható, egyszerű, keretekre épülő oldal. Az egyes keretek tartalmát szokásos HTML-oldalként hozzuk létre. Az oldalak a következők: `top.html` (ez tartalmazza a többi oldalhoz vezető hivatkozásokat), `home.html`, `products.html`, `services.html` és `contact.html`. Az oldalak egyike sem tartalmaz egy olyan címkét sem, amellyel az előző órák során ne találkoztunk volna. Az oldalak egybefogására egy különleges szerepű weboldalt, egy úgynevezett keretvázat használunk. Esetünkben ennek az oldalnak a neve `index.html`.

A keretvázat tartalmazó oldal elkészítése

A *keretváz* (frameset) egy olyan HTML-oldal, amely arra utasítja a böngészőt, hogy a böngészőablakot ossza több részre, és azt is megadja, hogy az egyes keretekben mely weboldalnak kell megjelenünie.

A keretváznak otthont adó dokumentum tehát nem bír saját tartalommal, csak arra való, hogy a böngészőnek megadja a további betöltendő oldalak nevét, illetve az elrendezésük módját. A 13.1. példa a 13.1. és a 13.2. ábrán látható webhely keretvázat tartalmazó oldalát mutatja be.

13.1. példa A 13.1. ábrán látható webhely keretvázat tartalmazó oldala

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Framed Site</title>
  </head>

  <frameset rows="50,*">
    <frame src="top.html" name="top" />
    <frame src="home.html" name="main" />
  <noframes>
    <body>
      <h1>Sample Framed Site</h1>
      Your browser does not support frames. Sorry!
    </body>
  </noframes>
</frameset>
</html>
```

A 13.1. példában a `<body>` címke helyére a `<frameset>` (keretváz) címke került. Azok a címkék, amelyek a `<body>` címképár által közrefogva szerepelnek, nem kerülhetnek a `<frameset>` címkék közé. A példában szereplő `<frameset>` elem `rows` (sorok) jellemzője hatására a keretek egymás felett, egy táblázat soraihoz hasonlóan helyezkednek el. Ha a kereteket egymás mellé kívánjuk elhelyezni, akkor a `rows` jellemző helyett a `cols` (oszlopok) jellemzőt kell használnunk.



Fontos, hogy észrevegyük, hogy a mintaoldalon nem a könyvben eddig használt DOCTYPE-bejegyzés szerepel. Ennek az az oka, hogy az XHTML 1.1 szabványban nem szerepel a keretek használata. Vagyis, ha azt szeretnénk, hogy az oldalaink érvényesek legyenek, az XHTML 1.0 keretvázakhoz való dokumentumtípus-meghatározást (Document Type Definition, DTD) kell használnunk. Egy olyan különleges meghatározásról van szó, amelyet éppen a kereteket használó oldalakhoz terveztek.

A sorok, illetve az oszlopok számát nekünk kell megadnunk, mégpedig vagy képpontra pontosan, vagy a teljes böngészőablak százalékos arányában. Csillaggal (*) jelölhetjük, ha azt szeretnénk, hogy a keret kitöltse az ablakból még rendelkezésre álló részt. Ha több keretnél is csillag szerepel, akkor az ilyen ablakok a fennmaradó részen egyenlő arányban osztoznak.

A 13.1. példában a `<frameset rows="50, *">` címke úgy osztja függőlegesen két részre az ablakot, hogy a felső keret pontosan 50 képpont magas lesz, az ablakból fennmaradó részt pedig az alsó keret kapja. A felső keret tartalmazza a `top.html` oldalt – lásd a 13.2. példát –, az alsó keretben pedig a 13.3. példában ismertetett `home.html` oldal jelenik meg.

13.2. példa A mintaoldal navigációs sávja, azaz a `top.html` oldal

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Framed Site</title>
  </head>

  <body style="background-color:#0000FF;">
    <div style="text-align:center;color:#FFFFFF;font-weight:bold;
      font-size:16pt">
      <a style="color:#FFFFFF;" href="home.html"
        target="main">HOME</a> ::
      <a style="color:#FFFFFF;" href="products.html"
        target="main">PRODUCTS</a> ::
      <a style="color:#FFFFFF;" href="services.html"
        target="main">SERVICES</a> ::
```

```

<a style="color:#FFFFFF;" href="contact.html"
  target="main">CONTACT</a>
</div>
</body>
</html>

```



A 13.1. példában a keretvázakat követően a `<body>` és a `</body>` címke között egy teljes értékű weboldalt találunk. Megfigyelhetjük, hogy ez a weboldal sem a 13.1., sem a 13.2. ábrán nem látható. A kereteket támogató webböngészők ugyanis figyelmen kívül hagyják a `<noframes>` és a `</noframes>` címke közötti részt. Manapság minden jelentősebb böngésző képes a keretek kezelésére, azaz ma már lényegesen kevesebb a keretek használatából eredő, ilyen jellegű probléma, mint akár csak néhány éve. Ugyanakkor a még mindig valamilyen ősi böngészőt használó néhány látogatóra gondolva nem túl megterhelő elhelyezni a `<noframes>` címkét az oldalon – persze ha egyáltalán használunk kereteket az oldalon.

13.3. példa A mintaoldal alsó részén található, valódi tartalommal bíró home.html oldal

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Framed Site</title>
  </head>

  <body style="background-color:#FFFFFF">
    <h1 style="text-align:center">Sample Framed Site: Home</h1>
    <p style="text-align:center">This is an example of the "home"
page.</p>
  </body>
</html>

```



A 13.2. és a 13.3. példában egyaránt az XHTML 1.0 átmeneti (transitional) dokumentumtípus-meghatározását használjuk. Az XHTML 1.1 és az újabb változatok lényegesen szigorúbbak, de a keretek miatt nekünk amúgy is az XHTML 1.0-s változat szükséges. Ezzel indokolható, hogy a keretek belsejében is ezt a dokumentumtípus-meghatározást használjuk.

A fenti példában a felső navigációs sáv magasságát 50 képpontban rögzítettük. Minthogy azonban nem tudjuk megjósolni a felhasználók böngészőablakának méretét, sok esetben kényelmesebb képpont helyett százalékban kifejeznünk a sorok és oszlopok méretét. Ha például azt szeretnénk, hogy a bal oldali keret a böngészőablak 20%-át foglalja el, a jobb oldali kereté pedig legyen a maradék 80%, akkor az alábbi címkét is használhatjuk:

```
<frameset cols="20%,80%">
```

Amikor a keret méretét képpontban adjuk meg, okos dolog legalább a keretváz egyik keretében változó szélességet (*) megadnunk. A dokumentum így megnőhet, és alkalmas lehet akár mekkora böngészőablak kitöltésére.

Az egyes keretek kitöltése

A `<frameset>` és a `</frameset>` címke között kell elhelyeznünk azokat a `<frame />` címkéket, amelyekből kiderül, hogy az egyes keretekben mely HTML-oldalak jelenjenek meg. Megjegyeznénk, hogy amennyiben kevesebb `<frame />` címkénk van, mint ahány keretet a `</frameset>` címkében megadtunk, akkor a létszámon felüli keretek üresen maradnak. Az egyes `<frame />` címkéken belül megadott `<src>` elemmel jelezhetjük a betöltendő weboldal helyét. Egy weboldal URL-je helyett kerülhet ide egy képfájl is – ilyenkor a keretben csak az adott kép jelenik meg.

Hivatkozások keretek és ablakok között

A keretek biztosította igazi lehetőségek akkor kezdenek nyilvánvalóvá válni, amikor egy `<frame />` elemben a `name` (név) jellemzővel egyedi nevet adunk a keretnek. Ettől a pillanattól kezdve az oldalon elhelyezett hivatkozásokkal megváltoztathatjuk az elnevezett keret tartalmát. Ehhez az `<a>` elem `target` (cél) jellemzőjét kell használnunk. A 13.1. példában például megtalálható az alábbi címke:

```
<frame src="home.html" name="main" />
```

A fenti kódsor az oldal betöltésekor az alsó keretben a `home.html` oldalt jeleníti meg, a keretnek pedig a `main` nevet adja.

A felső keret kialakítását végző, a 13.2. példában olvasható kódban található az alábbi hivatkozás:

```
<a style="color:#FFFFFF;" href="services.html"
target="main">SERVICES</a>
```

Amikor a felhasználó a fenti hivatkozásra kattint, a `main` nevű (alsó) keretben a `services.html` oldal jelenik meg. Ha elhagynánk a `target="main"` jellemzőt, az oldal az aktuális (felső) keretben jelenne meg.



Technikai értelemben a `name` jellemző elavultnak számít, és a pótlására már létezik az `id` jellemző. Mindazonáltal napjaink webböngészői az `id` helyett továbbra is a `name` jellemzőt használják az olyan esetekben, amikor keretek szerepelnek célként megadva. A `name` jellemző használata ugyanakkor nem ütközik az XHTML szabvány érvényességi szabályaiba, így aztán még egy darabig a keretek azonosítására a `name` jellemzőt kell használnunk. Természetesen nem baj, ha mindkét jellemzőt megadjuk.

A `services.html` oldal kódját helytakarékoság miatt nem mutatjuk be – szokványos weboldal, amely semmilyen, a keretek témaköréhez tartozó érdekességgel nem bír. A 13.2. ábrán látható, hogy miként néz ki a keretváz belsejében.

Vannak HTML-jellemzők, amelyek arra szolgálnak, hogy a kereteket használó weboldalon eltüntessük a keretek közötti határvonalakat, vagy a margók méretének csökkentésével nagyobb helyet kapjunk a kicsi keretek belsejében, esetleg a kereteknek megtiltsuk a gördítősávok megjelenítését. A 13.4. példában a 13.1. példakód módosított változatát találjuk. Két változás került a kódba: a `<frame>` címkén belül elhelyeztük a `scrolling="no"` és a `frameborder="0"` jellemzőt.

13.4. példa A 13.3. ábrán bemutatott webhely keretváza

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Framed Site</title>
  </head>

  <frameset rows="50,*">
    <frame src="top.html" name="top" scrolling="no" frameborder="0" />
    <frame src="home.html" name="main" scrolling="no" frameborder="0" />
  <noframes>
    <body>
      <h1>Sample Framed Site</h1>
      Your browser does not support frames. Sorry!
    </body>
  </noframes>
</frameset>
</html>
```



13.3. ábra

A 13.4. példa alapján kialakuló oldal – a `<frame />` elemet bővítettük újabb jellemzőkkel

Belső keretek használata

A belső kereteket (inline frame; szövegekőzi keret) nem sújtják azok a használhatósággal kapcsolatos problémák, amelyekkel a szokványos kereteknél találkozunk. Persze az is igaz, hogy a belső kereteket másra használjuk – nem az elrendezés kialakításakor ügyeskedünk velük. Az `<iframe>` címke szerepe inkább az `<object>` címkéhez hasonló, azaz egy létező dokumentumban tudunk vele elhelyezni valamit. Ez a „valami” az `<object>` címke esetében általában valamilyen multimédiás tartalom, az `<iframe>` elemet viszont egy teljesen önálló HTML-oldal, kép vagy egyéb dolog beágyazására használhatjuk. A 13.5. és a 13.6. példában azt a kódot ismertetjük, amelyik a 13.4. ábra belső keretének kialakítását végzi.

13.5. példa *<iframe>* címkét használó XHTML-kód

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

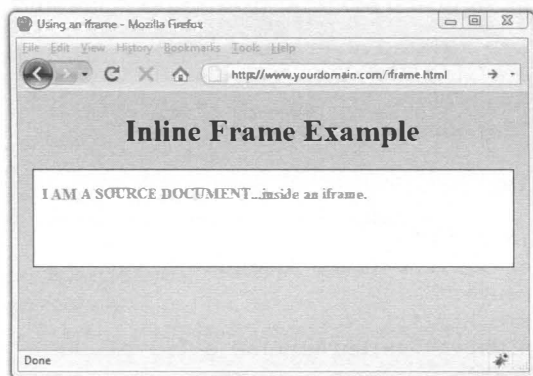
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Using an iframe</title>
  </head>
  <body style="background-color:#CCCCCC">
    <h1 style="text-align:center">Inline Frame Example</h1>
    <div style="text-align:center">
      <iframe src="iframe_src.html"
        style="width:500px;height:100px;border:1px solid black;
        background-color:#FFFFFF">
        <p>Uh oh...your browser does not support iframes.</p>
      </iframe>
    </div>
  </body>
</html>
```

A 13.5. példában egyetlen olyan XHTML-kódrészlet található, amellyel eddig még nem találkoztunk, nevezetesen maga az `<iframe>` címke. Látjuk, hogy az `src` (source, azaz forrás) jellemzőnek kell értéket adnunk, és azt is látjuk – többek közt –, hogy miként történik a szélesség, a magasság, a kerettípus és a háttérszín beállítása. A 13.6. példakód az `<iframe>` elem forrásaként megadott fájl tartalmát ismerteti – egy szokványos XHTML-fájlról van szó, némi szöveggel és néhány formázási beállítással.

13.6. példakód *A 13.5. példa <iframe> eleme ezt a kódot hűjja*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>iframe source</title>
  </head>
  <body>
    <p style="color:#FF0000;font-weight:bold">I AM A
      SOURCE DOCUMENT...inside an iframe.</p>
  </body>
</html>
```

**13.4. ábra**

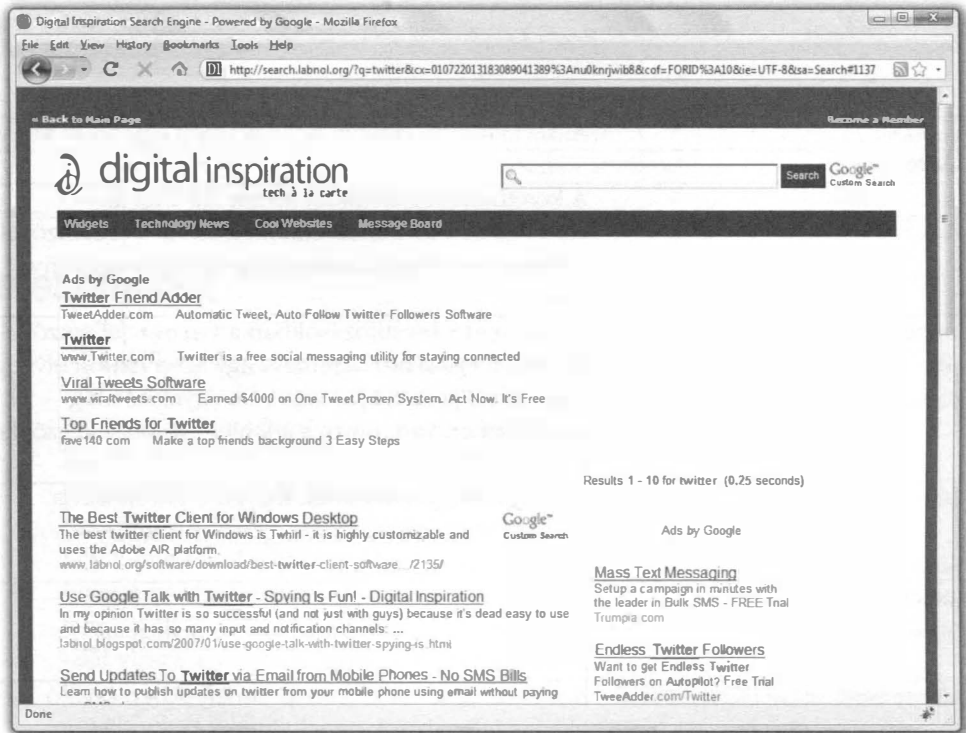
A 13.5. példa <iframe> eleme azt a belső keretet határozza meg, amelynek a kódja a 13.6. példában látható

Belső kereteket sokszor használunk olyankor, amikor egy weboldalba más webhelyekről származó tartalmat akarunk beilleszteni. Gyakran van a segítségünkre, ha a felhasználónak egy külső hirdetésszolgáltató hirdetéseit szeretnénk megmutatni, vagy ha a Google Site Search szolgáltatás eredményeit kívánjuk a felhasználó elé tárni (ilyenkor valójában a Google cég keresőtechnológiáját használjuk a saját céljainkra). A 13.5. ábrán azt látjuk, ahogy egy oldalsablonban az <iframe> címke megjeleníti egy keresés eredményét.

A 13.5. ábrán minden, ami a fehér területen található, valójában egy belső keretben kap helyet. A keretet kialakító <iframe> elem src jellemzője egy a Google cég webhelyén futó parancsfájlról mutat, amely a Digital Inspiration webnapló oldalán belül jeleníti meg

a keresés eredményét. A 13.5. ábrát figyelmesen szemlélve – bár nem valószínű, hogy a képen is látszik – egy vékony, szürke határolóvonalat fedezhetünk fel a belső keret körül.

Az óra korábbi részében megismert `<frame />` címkével ellentétben az `<iframe>` címkének van létjogosultsága, és még a HTML 5 szabványnak is része.



13.5. ábra

A Google Site Search kereső eredményeit egy belső keretben jelenítjük meg

Összefoglalás

A mai órán megtudtuk, hogy miként jeleníthető meg több oldal egyszerre úgy, hogy a webböngésző ablakát keretekkel tagoljuk. Megismerkedtünk a keretváz-oldalak létrehozásának módjával, és így már képesek vagyunk szabályozni a keretek méretét és elrendezését, és meg tudjuk adni a keretekbe betöltendő oldalakat. Megtanultunk olyan hivatkozásokat megadni, amelyek úgy változtatják meg egy-egy keret tartalmát, hogy a többi keret változatlan marad. Tanultunk néhány olyan elhagyható beállításról

is, amelyek a keretek között lévő méretezhető határolókat és a keretekben lévő gördítősávokat vezérlik. Az óra végén pedig elsajátítottuk a belső vagy szövegekői keretek használatát, amelyekkel akár a saját webhelyünkről származó, akár máshol található oldalak tartalmát illeszthetjük be a saját oldalainkba.

A 13.1. táblázat a mai órán tárgyalt elemeket és jellemzőket foglalja össze.

13.1. táblázat A 13. órán tárgyalt HTML-elemek és -jellemzők

Elemek és jellemzők	Leírás
<code><frame /></code>	A <code><frameset></code> címkén belül ad meg egy adott keretet.
Jellemzők	
<code>src="url"</code>	A keretben megjeleníteni kívánt fájl URL-je.
<code>id="név"</code>	Az <code><a href></code> hivatkozásokban a <code>target</code> jellemzővel célként megadható keretnév az XHTML-szabvány szerint.
<code>name="név"</code>	Az <code><a href></code> hivatkozásokban a <code>target</code> jellemzővel célként megadható keretnév. Egy szép napon elvileg majd felváltja az <code>id</code> jellemző, de egyelőre még használatban van, mivel a jelenlegi webböngészőkkel ez működik.
<code>scrolling="yes/no/auto"</code>	Megadja, hogy a keretnek legyen-e gördítősávja. A lehetséges értékek: <code>yes</code> (igen), <code>no</code> (nem) és <code>auto</code> (ha kell, akkor legyen).
<code>noresize="noresize"</code>	Megakadályozza, hogy a felhasználó az egérrel átméretezze az adott keretet – és esetlegesen a szomszédos kereteket.
<code><frameset>...</frameset></code>	A főablakot keretekre osztja, amelyekben önálló oldalak jeleníthetők meg.
Jellemzők	
<code>rows="sorok_száma"</code>	Az ablakot, illetve a keretvázat függőlegesen, sorokra osztja fel. A sorok számát megadhatjuk közvetlenül is (például: 7), vagy az ablak teljes szélességének százalékában (például: 25%). A csillag (*) azt jelzi, hogy a keret a teljes fennmaradó helyet magába foglalhatja – ha több helyen adunk meg csillagot, akkor az ilyen keretek között a fennmaradó hely egyenlő arányban oszlik el.
<code>cols="oszlopok_száma"</code>	A <code>rows</code> jellemzőhöz hasonlóan működik, azzal a különbséggel, hogy az ablakot, illetve keretvázat vízszintesen, oszlopokra osztja fel.

13.1. táblázat A 13. órán tárgyalt HTML-elemek és -jellemzők

(folytatás)

Elemek és jellemzők	Leírás
<code>frameborder="yes/no"</code>	Azt adja meg, hogy egy keret kapjon-e határolóvonalat. A lehetséges értékek: <code>yes</code> (igen) és <code>no</code> (nem).
<code><noframes>...</noframes></code>	A keretvázlat tartalmazó dokumentumban olyan további szövegtörzset adhatunk meg vele, amely a keretek használatát nem támogató böngészőkben jelenik meg – rendszerint egy <code><body>...</body></code> címkepár kerül a <code><noframes></code> címkepár belsejébe.
<code><iframe>...</iframe></code>	Belső keretet hoz létre. A <code><frame /></code> elem valamennyi jellemzőjét használhatjuk vele, és CSS segítségével formázható.

Kérdezz-felelek

- K:** *Megoldható, hogy az egyik keretben valaki más weblapját jelenítsem meg, miközben a másikban ezzel egyidejűleg a saját weboldalam látható? Mi történik, ha ezek a weboldalak is használnak kereteket?*
- V:** Egy keretbe az Internet vagy a belső hálózat bármely weboldala betölthető. Ha a betöltött weboldal egy keretváz, akkor a benne lévő keretek az oldalnak helyet adó keret kiterjedésének megfelelően méreteződnek át. Egy keretben elhelyezhetjük például a kedvenc hivatkozásainkat, és egy másik keretet használhatunk a hivatkozott tartalom megjelenítésére. Így annak kockázata nélkül nyújthatunk hivatkozásokat, hogy a látogató eltéved, és sohasem talál vissza a webhelyünkre.
- Persze nem árt azt is tudnunk, hogy ha egy oldalunk keretében valaki másnak a webhelyét jelenítjük meg, akkor esetleg jogi hercelhúrcának nézünk elébe. Okosabb, ha a kereteinkben elhelyezett oldalak tulajdonosaitól írásos engedélyt szerzünk – pontosan úgy, mint amikor valaki más oldaláról származó szöveget vagy képet helyezünk el egy weboldalunkon.
- K:** *Minden keretbe kerülő oldalknak kell <title> címkével, illetve címmel rendelkeznie? Ha mindegyiknek van címe, akkor melyik fog megjelenni az ablak tetején?*
- V:** A keretvázlat tartalmazó oldal címe lesz az egyetlen látható cím. A keretbe kerülő oldalak esetében sem a `<head>`, sem a `<title>` címke nem elvárás, bár okosan tesszük, ha mégis megadjuk ezeket a címkéket, hátha egy látogató önálló oldal-ként nyitja meg a fájlt, és nem egy keret belsejében.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Írjuk olyan HTML-kódot, amely a Mickey, a Minnie és a Donald nevet egy böngészőablak bal oldalának 25%-át elfoglaló keretben jeleníti meg. Oldjuk meg, hogy ha a felhasználó az egyes nevekre kattint, akkor a böngészőablak maradjék 75%-ában egy megfelelő weboldal nyíljon meg.
2. Milyen `<iframe>`-kóddal alakíthatunk ki egy határolóvonal nélküli, az oldal 98%-ára kiterjedő, 250 képpont magas belső keretet?

Válaszok

1. Öt külön HTML-oldalt kell készítenünk. Az első oldalban a keretváz kap helyet:

```
<html>
  <head>
    <title>Our Friends</title>
  </head>

  <frameset cols="25%,75%">
    <frame src="index.html" />
    <frame src="mickey.html" name="mainframe" />
  </frameset>
</html>
```

A bal keretbe kerül az `index.html` oldal:

```
<html>
  <head>
    <title>Our Friends Index</title>
  </head>

  <body>
    <p>Pick a friend:</p>
    <p><a href="mickey.html" target="mainframe">Mickey</a><br />
      <a href="minnie.html" target="mainframe">Minnie</a><br />
      <a href="donald.html" target="mainframe">Donald</a></p>
  </body>
</html>
```

Ezt követően még három oldalt kell létrehoznunk, mégpedig `mickey.html`, `minnie.html` és `donald.html` néven. Ezekbe kerülnek az egyes rajzfilmfigurákkal kapcsolatos információk.

2. Íme egy lehetséges megoldás:

```
<iframe src="some_source.html"
style="width:98%;height:250px;border:none;
background-color:#FFFFFF">
  <p>Put message here for people not able to see the inline
    ➡ frame.</p>
</iframe>
```

Gyakorlatok

- Gondoljuk át, hogy milyen okok készíthetnek bennünket egy keretekre épülő elrendezés használatára, és vázoljuk fel az elrendezést egy lapra. Tegyük el a vázlatot a következő leckék idejére, hiszen akkor fogjuk megtudni, hogy miként tervezhető elrendezés az XHTML és a CSS használatával. Ezzel a módszerrel olyan, a szabványoknak megfelelő és felhasználóbarát eszköz kerül a kezünkbe, amellyel a keretekhez hasonló megjelenés és működés valósítható meg.
- Találjunk módot egy-két belső keret használatára a webhelyünkön. Lehet szó például hirdetésről vagy a felhasználóink hasznára szolgáló, ingyenes Google Site Search elhelyezéséről az oldalainkon. A szerkezetben tartsunk fenn helyet ezeknek a kereteknek.



14. ÓRA

A külső és a belső margó, az igazítások és az úsztatás használata

A lecke tartalma:

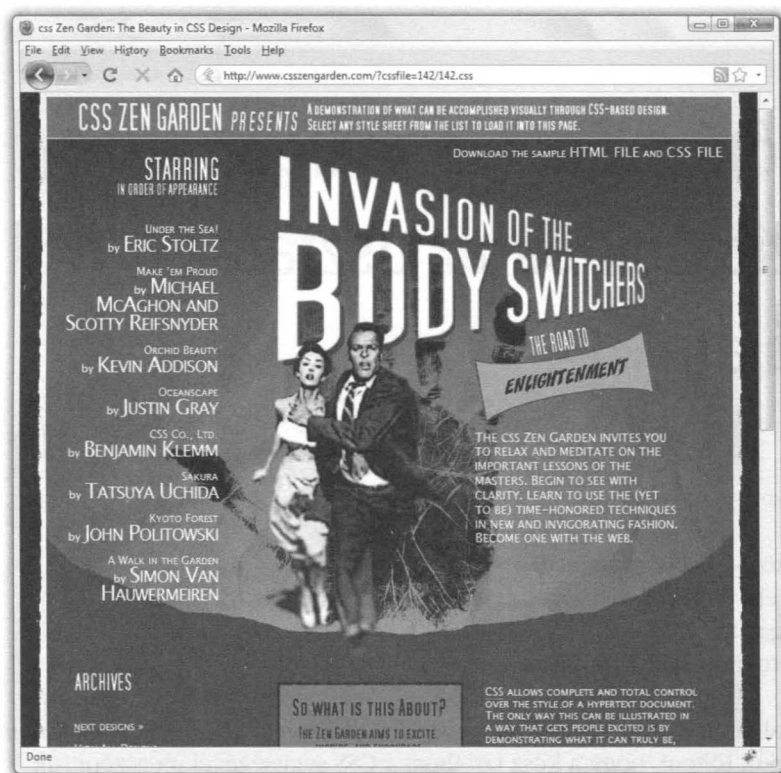
- Elemek körbevételre margóval
- Hogyan tehetünk belső margót egy elemen belülre?
- Elemek igazítása
- A `float` tulajdonság használata

Most, hogy tisztában vagyunk a webes tartalom kialakításának alapvető fogásaival, ezt az órát azzal fogjuk tölteni, hogy a kész tartalmat a CSS segítségével megszépítsük. Az előző órák során megismertük a CSS alapvető, megjelenítésre szolgáló elemei – például a betűméret és a betűszín beállítását végzők – kezelését. A most következő leckék során belemerülünk annak tárgyalásába, hogy a CSS miként használható önálló szövegrészletek és grafikai elemek vezérlésén túl egész „oldalak” kinézetének formázására.

Mielőtt azonban az oldal elrendezésének tárgyalásába foglalnánk, fontosnak véljük, hogy még az együttes használatukat megelőzően egyenként megismerjük az alábbi négy CSS-tulajdonságot:

- A `margin` (margó) és a `padding` (belső margó vagy kitöltés) tulajdonság arra szolgál, hogy az elemek körül üres területet foglaljunk le.
- Az `align` (igazítás) és a `float` (úsztatás) tulajdonság pedig arra való, hogy az elemek elhelyezkedését egymáshoz képest megadhatassuk.

A mostani lecke során bemutatott példák nem tartoznak a valaha készült legszebb weboldalak közé, ám nem is ezzel a szándékkal készültek. Ugyanakkor remekül alkalmasak az XHTML nyelv és a CSS együttműködésének bemutatására. Mire a mostani és az elkövetkezendő órák során az Olvasó is a CSS szakértőjévé válik, képes lesz olyan webes remekművek elkészítésére, mint a 14.1. ábrán látható, a CSS Zen Garden webhelyen megtekinthető weboldal.



14.1. ábra

Az ábrán csak egy látható a CSS Zen Garden webhelyen található számtalan, az XHTML nyelv és a CSS együttműködését példázó alkotás közül

A CSS Zen Garden oldalai nem feltétlenül néznek úgy ki, mint a szokásos webáruházak és közösségi hálózatok webhelyei, amelyeket időről időre felkeres az ember. Éppen ellenkezőleg: ezek az oldalak igyekeznek megvalósítani mindazt, ami a CSS használatával művésziileg egyáltalán lehetséges. Tévedés ne essék: minden itt található oldal mögött egy remek ötlet és gondos tervezés áll, de a CSS nyújtotta lehetőségek végtelenek.



A CSS Zen Garden (<http://www.csszengarden.com/>) olyan weboldalterveket mutat be, amelyek a CSS-szabványoknak megfelelő lehetőségeken alapulnak.

A felhasználók által beküldött összes mű pontosan ugyanazt a HTML-fájlt használja, de a művészek a saját látványtervük megvalósítása végett szabadon módosíthatják a CSS-állományt. A 14.1. ábrán látható weboldalt a Stuff and Nonsense cég (<http://www.stuffandnonsense.co.uk/>) Andy Clarke nevű munkatársa készítette.

Margók használata

A stíluslapok *margói* segítségével a weboldalak egy téglalappal határolható elemén *kívül* biztosíthatunk üres területet. Tartsuk észben, hogy a margin tulajdonság az elemet körülvevő üres terület nagyságát szabályozza.

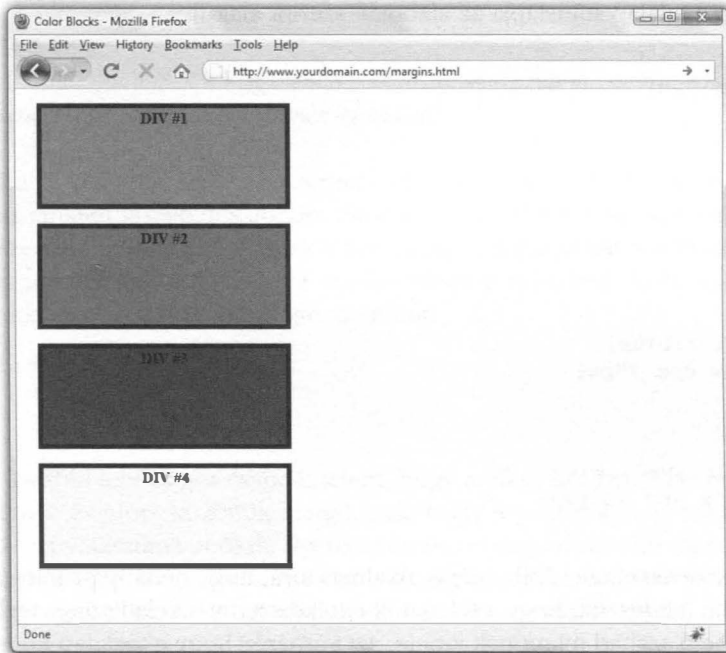
A margó beállítására az alábbi tulajdonságok szolgálnak:

- `margin-top`: A felső margó beállítását végzi.
- `margin-right`: A jobb margó beállítását végzi.
- `margin-bottom`: Az alsó margó beállítását végzi.
- `margin-left`: A bal margó beállítását végzi.
- `margin`: A felső, a jobb, az alsó és a bal margókat állítja be egy tulajdonságként.

A margók beállítása mind az egyedi margótulajdonságok megadásával, mind az önálló `margin` tulajdonság megadásával lehetséges. A margók beállítása lehet automatikus, azaz a böngésző maga végzi el a margó beállítását az adott mértékegységben (képpont, pont, em), vagy százalékos arányban kifejezve. Ha a margó százalékos beállítása mellett döntünk, az arány kiszámítása nem az adott elem, hanem a teljes oldal mérete alapján történik. Ez annyit tesz, hogy amennyiben a `margin-left` tulajdonság értékéül 25%-ot adunk meg, az elem bal oldalára a teljes oldalszélesség huszonöt százalékának megfelelő vastagságú margó kerül.

A 14.1. példa négy darab, 250 képpont széles és 100 képpont magas, 5 képpont vastag egyszínű fekete kerettel határolt téglalapot rajzol az oldalra (lásd a 13.2. ábrát). Minden téglalapnak – esetünkben: `<div>` címkével határolt területnek – más és más a háttérszíne. Ha azt szeretnénk, hogy a téglalapok körül 15 képpontnyi margó legyen, az alábbi tulajdonságokat kell megadnunk:


```
div#d2 {  
    background-color:green;  
    margin:15px;  
}  
  
div#d3 {  
    background-color:blue;  
}  
  
div#d4 {  
    background-color:yellow;  
    margin:15px;  
}  
</style>  
</head>  
  
<body>  
    <div id="d1">DIV #1</div>  
    <div id="d2">DIV #2</div>  
    <div id="d3">DIV #3</div>  
    <div id="d4">DIV #4</div>  
</body>  
</html>
```



14.2. ábra

A színes téglalapokat megjelenítő oldal kezdeti változatában a téglalapok margói egyformák

A következő gyakorlat, hogy próbálkozzunk a 14.1. példában szereplő margin tulajdonságoknak más értékeket adni. A fenti példában a <div> címkékkel határolt téglalapok jobb oldali margói nem igazán látszanak, hiszen semmi nincs tőlük jobbra, és nincsenek jobbra rendezve sem. Ennek fényében állítsuk be valamennyi téglalap margin-right tulajdonságául a 0px értéket. Egyben tegyünk eleget az alábbi céloknak is:

- Az első színes téglalap körül szüntessük meg a margót.
- A második színes téglalap bal margója legyen tizenöt képpontos, a felső margója öt képpontos, alul pedig ne legyen margója.
- A harmadik színes téglalap bal margója legyen hetvenöt képpontos, és távolítsuk el a felső és az alsó margóit.
- A negyedik színes téglalap bal margója legyen 250 képpontos, a felső pedig 25 képpontos.

A feladat megoldása magától értetődőnek tűnik: az első téglalap körül nem állítunk be margót. A második téglalap tetejére azonban szeretnénk margót, vagyis az első két téglalap között akkor is margót fogunk látni, ha az első téglalap körül nem adunk meg margót.

A névvel ellátott négy téglalap stíluslapja az alábbiak szerint alakul:

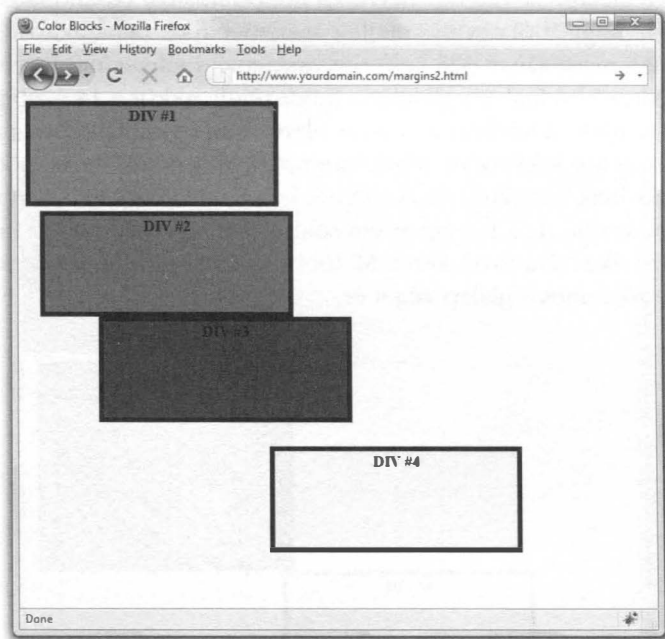
```
div#d1 {
  background-color:red;
  margin:0px;
}

div#d2 {
  background-color:green;
  margin:5px 0px 0px 15px;
}

div#d3 {
  background-color:blue;
  margin:0px 0px 0px 75px;
}

div#d4 {
  background-color:yellow;
  margin:25px 0px 0px 250px;
}
```

A 14.3 ábra eléggé összevisszának tűnik, mégis alkalmas arra, hogy néhány problémát tisztázzunk. Például ha felidézzük, hogy a feladatok egyike szerint az első színes téglalap körül egyáltalán nem szabad margónak lennie, azt várnánk, hogy a téglalap kerete egybefolyik a böngészőablakkal. Ahogy azonban a 14.3. ábrán látható, az oldal tartalma és a böngészőablak kerete között üres hely található.



14.3. ábra

A színes téglalapokat megjelenítő példaoldalon végzett módosítások hatására a margók mostanra némileg különböznek egymástól

Ha az elemek elhelyezése lenne a cél – ami a következő órán lesz a feladatunk –, akkor az említett viselkedés gondot okozna az elrendezéssel kapcsolatosan. Ha biztosítani szeretnénk, hogy az elemek elhelyezése és a margók a böngészőablak szélétől legyenek számolva, akkor a margót a `<body>` elemnél is be kell állítanunk. Esetünkben az alábbiakkal kell a stíluslapot kiegészítenünk:

```
body {
    margin: 0px;
}
```

További lehetséges csapdát jelent, hogy amikor két bekeretezett elemünk van egymás felett, és nincs közöttük margó, akkor úgy fog látszódni, mintha kétszeres vastagságú keret választaná el őket. Ilyenkor érdemes elgondolkodni azon, hogy a felső elem alsó keretét (`border-bottom`) és az alsó elem felső keretét (`border-top`) fele olyan vastagra állítsuk. Ha így járunk el, akkor az egymásra kerülő elemek közötti keret egyforma vastagnak fog tűnni az elemek többi oldalán lévőkkel.

Esetleg azt vélné az ember, hogy ha 250 képpontos margót használunk – azaz a `<div>` címkékkel határolt területek szélességével megegyezőt –, akkor a negyedik színes téglalap ott fog kezdődni, ahol a harmadik téglalap széle található. Nos, nem ez a helyzet,

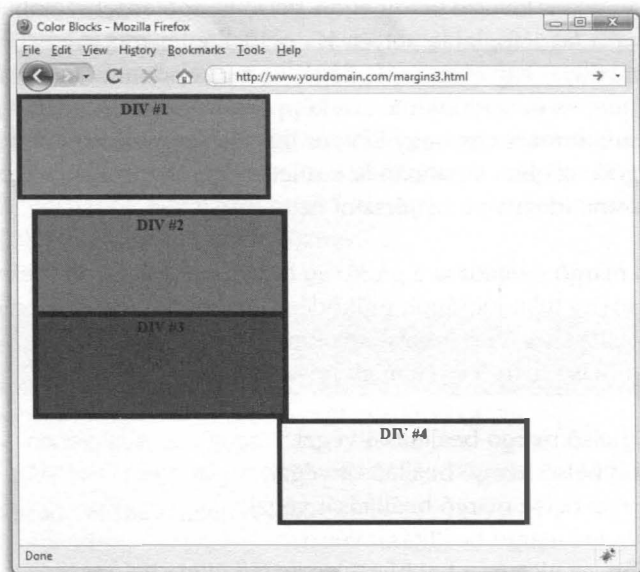
ugyanis a harmadik színes téglalapnak van egy hetvenöt képpontos bal oldali margója. Ha a két téglalapot akár csak nagyjából egymás mellett szeretnénk kezdeni, akkor a negyedik téglalap bal oldali margójának 325 képpont vastagnak kellene lennie. Ha a stíuselemeket a következő kódnak megfelelően módosítjuk, akkor a 14.4. ábrán megfigyelhető elrendezést kapjuk. A kódban a <body> elem margója nulla képpontos lesz, ami biztosítja, hogy a huszonöt képpontos bal margójú elem valóban huszonöt képpontra kerül a böngészőablak keretétől. Az is látható, hogy a második és a harmadik téglalap egymás tetejére került, de a border elem stílusát érintő változtatások miatt nem kétszeres vastagságú az őket elválasztó keret. Mi több, a negyedik színes téglalap ott kezdődik, ahol a harmadik színes téglalap véget ér.

```
body {
    margin:0px;
}
div {
    width:250px;
    height:100px;
    color:black;
    font-weight:bold;
    text-align:center;
}
div#d1 {
    border:5px solid #000000;
    background-color:red;
    margin:0px;
}
div#d2 {
    border-width:6px 6px 3px 6px;
    border-style:solid;
    border-color:#000000;
    background-color:green;
    margin:10px 0px 0px 15px;
}
div#d3 {
    border-width:3px 6px 6px 6px;
    border-style:solid;
    border-color:#000000;
    background-color:blue;
    margin:0px 0px 0px 15px;
}
div#d4 {
    border:5px solid #000000;
    background-color:yellow;
    margin:0px 0px 0px 265px;
}
```

A 14.4. ábra szerint a harmadik színes téglalap jobb széle némi átfedésben van a negyedik téglalap bal szélével. Mi ennek az oka? Hiszen a téglalapok 250 képpont szélesek, a harmadik téglalap bal oldali margója tizenöt képpontos, a negyedik téglalap bal oldalán pedig elvileg 265 képpontos margó terpeszkedik! Nos, igen, a 265 képpontos margó a helyén van, csak éppen egy ekkora margó kicsi, mert még a hat képpontos kerettel is

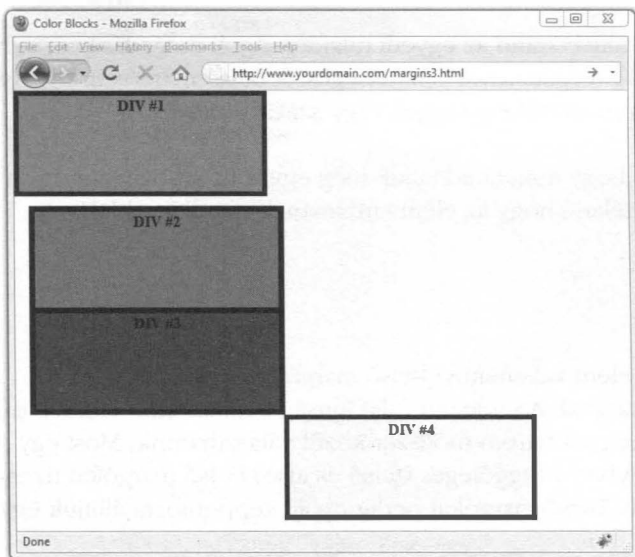
számolnunk kell. Ha a negyedik színes téglalap margin tulajdonságát a következő kódnak megfelelően módosítjuk, akkor a harmadik és a negyedik téglalap már a terveink szerint sorakozik egymás mellé – lásd a 14.5. ábrát.

```
margin:0px 0px 0px 276px;
```



14.4. ábra

A színes téglalapok a harmadik módosítás hatására közelebbi kapcsolatba kerülnek egymással



14.5. ábra

A margó megváltoztatása a tizenegy képpontnyi keretnek megfelelően

Ahogy az előző példákban láttuk, a margók beállítása igen-igen fontos az elemek elhelyezésekor, de az értékek megállapításánál érdemes fokozott figyelemmel eljárunk.

Az elemek belső margója

A belső margó vagy kitöltés hasonló a margóhoz, amennyiben ez is további hellyel bővíti az elemeket, de nagy a különbség e hely hollétét tekintve. Alighanem emlékszünk még arra, hogy margók az elemeken kívülre kerülnek. A belső margó ezzel szemben az elemet szegélyező téglalapon *belül* foglal le helyet. Ha példaként azt az esetet tekintjük, amikor egy stílusszabállyal egy elem szélességét ötven képpontra, a magasságát harminc képpontra állítjuk, és megadunk még öt képpontnyi belső margót is, akkor az elem belsejében a tartalomnak egy negyvenszer húsz képpontos hely marad. Minthogy az elem belső margója az elem tartalmának területére kerül, a stílusa az elem tartalmának stílusát fogja követni, ideértve a háttérszínt is.

A stílusszabályokban a belső margó megadása a padding tulajdonságok egyikének megadásával történik – a padding tulajdonságok működése egyébiránt igen hasonló a margin tulajdonságok működéséhez. A stílusszabályok megadásakor az alábbi tulajdonságok használhatók a belső margó értékeinek beállítására:

- padding-top: A felső belső margó beállítását végzi.
- padding-right: A jobb belső margó beállítását végzi.
- padding-bottom: Az alsó belső margó beállítását végzi.
- padding-left: A bal belső margó beállítását végzi.
- padding: A felső, a jobb, az alsó és a bal belső margókat állítja be egy tulajdonságként.

A margókhoz hasonlóan a beállítás mind az egyedi tulajdonságok megadásával, mind az önálló padding tulajdonság megadásával is lehetséges. A belső margó vastagságát szintén kifejezhetjük valamilyen mértékegységgel, vagy százalékosan.

Az alábbi példa azt ismerteti, hogy miként adhatjuk meg egy stílusszabályban úgy a bal és jobb oldali belső margó értékeit, hogy az elem tartalmának mindkét oldalán tíz képpontos hely maradjon:

```
padding-left:10px;
padding-right:10px;
```

A margókhoz hasonlóan egy elem valamennyi belső margója megadható egyedül a padding tulajdonság beállításával. A padding tulajdonság beállításakor is a margin tulajdonság beállításánál ismertetett három módszer közül választhatunk. Most egy olyan példa következik, amelyben a függőleges (felső és alsó) belső margókat tizenkettő, a vízszintes (bal és jobb) belső margókat pedig nyolc képpontosra állítjuk egy stílusszabályban:

```
padding:12px 8px;
```

A következő példa részletesebb: az előző feladatot valósítja meg ez is, de ezúttal az összes belső margó értékét megadjuk:

```
padding:12px 8px 12px 8px;
```

Az előző ábrákon láthattuk, hogy a „DIV #1”, a „DIV #2” és a többi szöveg a színes téglalapok tetejénél jelenik meg, éppen csak egy kis helyet hagyva a keret és a szöveg között. A hely nagyságát sehol nem adtuk meg padding tulajdonság értékeként, inkább valamilyen, az elem belsejére jellemző alapértelmezett értéknek tűnik. Ha az elemek belső margóit magunk szeretnénk beállítani, a 14.2. példakódban találunk néhány erre vonatkozó példát. Minden színes téglalap 250 képpont szélességű és 100 képpont magas, 5 képpont vastag fekete keretük van, és 25 képpontos margót állítottunk be – lásd a 14.6. ábrát. A téglalapok egymástól pusztán az egyes <div> címkékkel határolt részek belső margóinak beállításában különböznek.

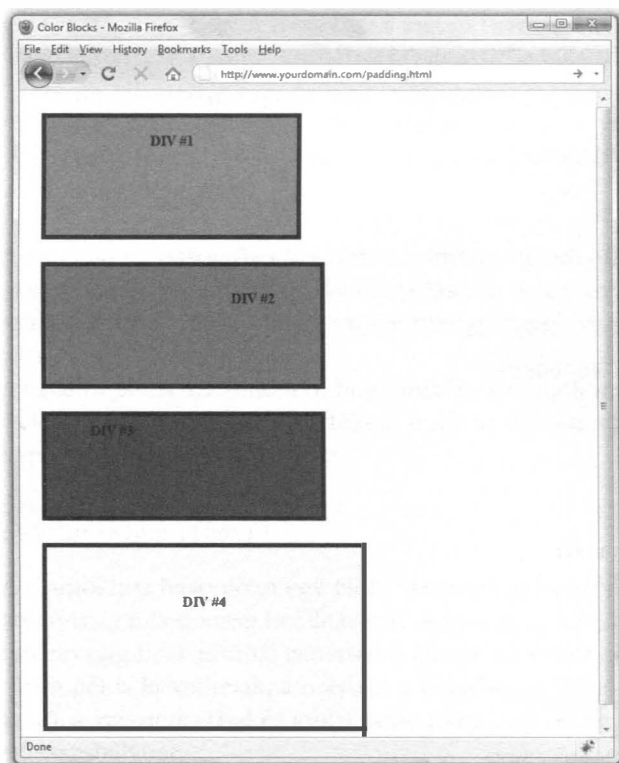
14.2. példa Négy, keretezett, színes, margókkal elválasztott és belső margókat is tartalmazó téglalapot rajzoló kód

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Color Blocks</title>
    <style type="text/css">
      body {
        margin:0px;
      }
      div {
        width:250px;
        height:100px;
        border:5px solid #000000;
        color:black;
        font-weight:bold;
        margin:25px;
      }
      div#d1 {
        background-color:red;
        text-align:center;
        padding:15px;
      }
      div#d2 {
        background-color:green;
        text-align:right;
        padding:25px 50px 6px 6px;
      }
    </style>
  </head>
  <body>
```



```
div#d3 {  
    background-color:blue;  
    text-align:left;  
    padding:6px 6px 6px 50px;  
}  
  
div#d4 {  
    background-color:yellow;  
    text-align:center;  
    padding:50px;  
}  
</style>  
</head>  
<body>  
  <div id="d1">DIV #1</div>  
  <div id="d2">DIV #2</div>  
  <div id="d3">DIV #3</div>  
  <div id="d4">DIV #4</div>  
</body>  
</html>
```

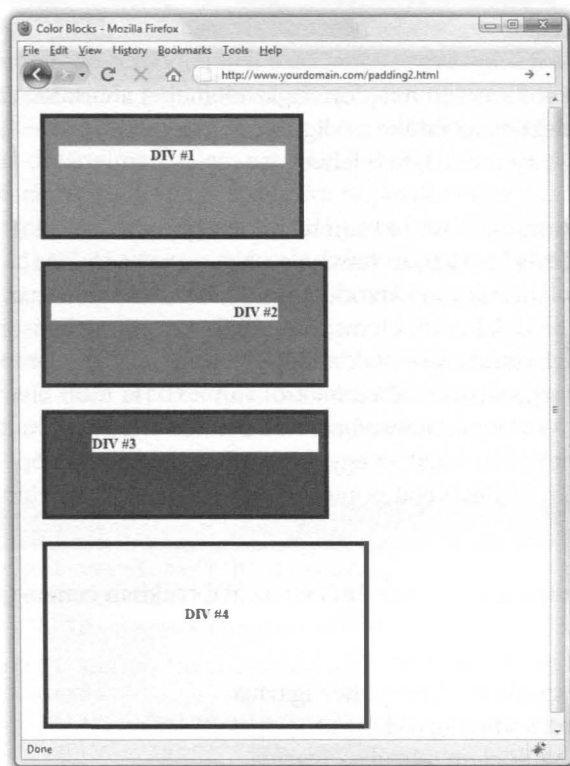


14.6. ábra

A négy téglalapot ábrázoló oldalunkon különféle belső margókat állítottunk be

Alighanem azonnal felfigyelünk rá, hogy a példával valami gond van. A téglalapok elvileg 250 képpont szélesek és 100 képpont magasak. A 14.6. ábra színes téglalapjai azért nem egyformák, mert bár igyekeztünk egyforma nagyra készíteni őket, a méretek megadását követően beállított belső margók felülbírálták az általunk beállított méretertékeket.

Ha a szöveget <p> címkék közé helyezzük, és az elemnek fehér hátteret adunk – lásd a 14.7. ábrát –, akkor láthatóvá válik a belső margó és a szöveg érintkezési pontja. Amikor egyszerűen kevés a hely a megadott belső margónak, a tartalmazó elem is megváltozik. Ezzel a jelenséggel részletesebben a 15. órán fogunk foglalkozni.



14.7. ábra

Láthatóvá válik a belső margó és a szöveg érintkezési pontja

A CSS alapú weboldalainkon végzett finomhangolások legnagyobb részében az elrendezés margóit és belső margóit fogjuk módosítani. Ne feledjük a legfontosabbat: a margók az elem körül, a belső margók pedig az elem belsejében találhatók.

Elemek igazítása

Annak ismeretében, hogy egy weboldal tartalma nem minden esetben tölti ki az őt tartalmazó téglalap teljes szélességét, gyakran bizonyul hasznosnak, ha a tartalom igazítását beállítjuk. Az igazításról már olyan egyszerű esetben is érdemes beszélnünk, amikor a téglalapban lévő szöveg több sornyira duzzad – hiszen ilyenkor is lehet értelme a szöveget balra, jobbra, vagy középre igazítanunk. Két olyan stílustulajdonság létezik, amellyel az elemek igazítása szabályozható: a `text-align` és a `vertical-align`.

A fenti tulajdonságokat már megfigyelhettük működés közben – képek igazításakor – a 11. órán, de nem árthat, ha ismét említést teszünk róluk, mivel az igazítás a teljes oldal megtervezésekor is szerepet kap.

Ismétlésként megemlítjük, hogy a `text-align` tulajdonság az elemeket a tartalmazó terület belsejében vízszintesen rendezi el, az értéke pedig `left` (balra), `right` (jobbra), `center` (középre) és `justify` (sorkizárás) lehet.

A `vertical-align` tulajdonság hasonló a `text-align` tulajdonsághoz, azzal a különbséggel, hogy az elemek függőleges igazítására használatos. A `vertical-align` tulajdonság azt adja meg, hogy az elem hogyan igazodik a szülőjéhez, illetve néhány esetben azt, hogy miként igazodik az oldal adott elemsorán belül. Az „adott elemsor” kifejezés olyan elemek függőleges elrendezésére utal, amelyek egyazon szülőelemen kaptak helyet – más szavakkal, az egy sorban lévő elemekről van szó. Ha több elem is egy sorba kerül, akkor a `vertical-align` tulajdonságaiknak azonos értéket adva igazíthatjuk egymás mellé őket. Remek példát kínál az egymás mellett megjelenő képek sora: a `vertical-align` tulajdonság beállításával pontosan egy magasságba kerülnek a soron belül.

A `vertical-align` tulajdonság elterjedten használt értékeit az alábbiakban ismertetjük:

- `top`: Az elem tetejét az adott sorhoz igazítja.
- `middle`: Az elem közepét a szülőelem közepéhez igazítja.
- `bottom`: Az elem alját az adott sorhoz igazítja.
- `text-top`: Az elem tetejét a szülőelem tetejéhez igazítja.
- `baseline`: Az elem betűvonalát a szülőelem betűvonalához igazítja.
- `text-bottom`: Az elem alját a szülőelem aljához igazítja.

Az igazítások együttesen fejtik ki hatásukat a margókkal, a belső margókkal és – ahogy a következő részből kiderül – a `float` tulajdonsággal. Így kézben tarthatjuk az oldal elrendezésének alakítását.


```
div#d1 {  
    background-color:red;  
    float:left;  
}  
  
div#d2 {  
    background-color:green;  
    float:left;  
}  
  
div#d3 {  
    background-color:blue;  
    float:left;  
}  
}</style>  
</head>  
  
<body>  
    <div id="d1">DIV #1</div>  
    <div id="d2">DIV #2</div>  
    <div id="d3">DIV #3</div>  
</body>  
</html>
```

A kód eredményeként előálló elrendezést a 14.8. ábra szemlélteti. Azonnal szembeötlik az első probléma: a téglalapoknak egymás mellé kellett volna úszniuk. Nos, az a helyzet, hogy egymás mellé úsztak, de a böngészőablak nem elég széles a 250 képpont széles és 25 képpontos margóval elválasztott téglalapok megjelenítéséhez. Úsztatott téglalapokról van szó, így a harmadik téglalap minden további nélkül átúszik a következő sorba.

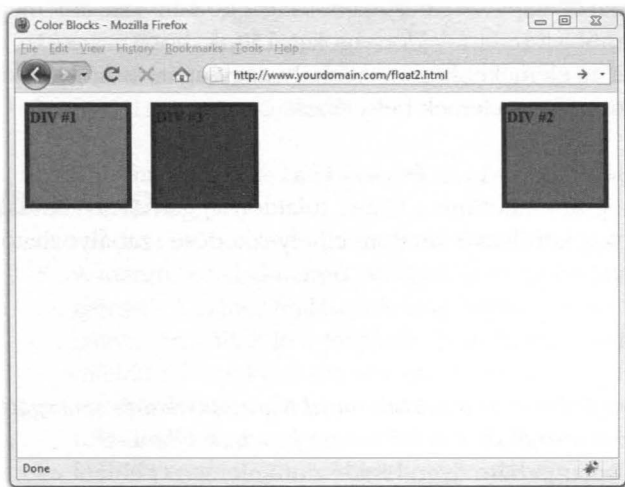


14.8. ábra

Téglalapok elhelyezése a float tulajdonság használatával

Elképzelhető, hogy a tapasztalt viselkedés problémákat okoz egy adott elrendezésnél, így nagy figyelemmel kell eljárunk a margók, belső margók, igazítások és úsztatások megadásakor, illetve amikor a böngészőablakot a célméretre állítva az oldal kipróbálását végezzük. Igaz, hogy a 14.8. ábrán látható böngészőablak elég kicsi – erre volt szükség ahhoz, hogy megfigyelhessük, ahogy az úsztatott elemek hely hiányában új sorba kerülnek –, és ha ugyanezt a HTML-fájlt egy nagyobb böngészőablakban nyitjuk meg, akkor a problémára fel sem figyelünk. Ezért illik az oldalainkat különböző felbontásban is kipróbálnunk, hiszen esetleg csak ekkor vesszük észre, hogy valamit még rendbe kell hoznunk. Esetünkben a „rendbehozás” a téglalapok margóinak és más, a mérettel kapcsolatos tulajdonságoknak az újbóli beállítását jelenti.

A 14.9. ábra egy további, az úsztatás használatakor előálló érdekes lehetőséget mutat be. A 14.3. példához képest annyit változtattunk, hogy a színes téglalapok szélességét 100 képpontra, a margókat pedig 10 képpontra csökkentettük, és a második színes téglalap `float` tulajdonságának értékét `left`-ről `right`-ra változtattuk.



14.9. ábra

Téglalapok elhelyezése a float tulajdonság használatával

A változtatás igen érdekes eredménnyel járt: a második színes téglalap most harmadiként látható, mert átúszott a jobb oldalra. A második színes téglalap `float` tulajdonságának értéke `right`, vagyis egészen a böngészőablak jobb oldalára úszott. Az első és harmadik színes téglalap annyira balra úszott, amennyire csak lehetséges volt.

Nem érvényesült az, hogy a `<div>` címkék sorrendje a HTML-fájlban az alábbi:

```
<div id="d1">DIV #1</div>
<div id="d2">DIV #2</div>
<div id="d3">DIV #3</div>
```

Az úsztatás működésének megszokása sok gyakorlást igényel, főleg akkor, ha egy oldalon néhány színes téglalapon kívül más is található. Mi történik például, ha a helyzetet egy egyszerű fénykép beillesztésével bonyolítjuk? Minden olyan elem, amely egy úsztatott elem beillesztése után kerül az oldalra, az úsztatott elem körül fog úszkálni. Ez a viselkedés a `clear` tulajdonság használatával előzhető meg.

A `clear` tulajdonság öt lehetséges értéket vehet fel: `left` (balra), `right` (jobbra), `both` (mindkettő), `none` (egyik sem), és `inherit` (öröklés). Leggyakrabban a `left`, a `right` és a `both` értéket szokás használni. A `clear:left` kód azt eredményezi, hogy más elemek már nem úszhatnak balra; a `clear:right` kód azt, hogy más elemek már nem úszhatnak jobbra, és így tovább. Az úsztatás és a `clear` tulajdonság használata leginkább gyakorlás útján sajátítható el – ehhez a lecke végén biztosítunk feladatokat.

Összefoglalás

A mai órán a CSS alapú weboldalak alapvető stílustulajdonságai közül ismertünk meg néhányat: a `margin`, a `padding` és a `float` tulajdonságokat. Láttuk, hogy a `margin` tulajdonság miként szabályozza az elemeket körbevevő hely nagyságát, és hogy miként teszi ugyanezt a `padding` tulajdonság az elemek belsejében.

Az egyik korábbi órán megismert `text-align` és `vertical-align` tulajdonságok újbóli áttekintését követően megismerkedtünk a `float` tulajdonsággal. A `float` tulajdonsággal az elemek és az azokat körülvevő tartalom elhelyezkedése szabályozható.

Kérdezz-felelek

- K:** *A margót és a belső margót bemutató példák mind téglalapokkal és szöveggel dolgoztak. Használható a margó és a belső margó képek esetében is?*
- V:** Igen, a margó és a belső margó bármilyen blokkszintű elemen (például `<p>`, `<div>` és ``), listákon (`` és ``), valamint listaelemeken (``) is használható.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Ha két, `<div>` címkével határolt elemet szeretnénk egymás mellé elhelyezni, közöttük 30 képpontos margóval, akkor milyen beállításokat kell a stíluslapon megadnunk?
2. Melyik CSS-tulajdonság, illetve érték használatával biztosítható, hogy egy úsztatott elem bal oldalára más tartalom már ne kerülhessen?
3. Milyen CSS-beállítás használható arra, hogy egy `<div>` címkével határolt elem belsejében a szöveg 12 képpont távolságra kerüljön az elem tetejétől?

Válaszok

1. Több megoldás is lehetséges. Vagy az első elemnek adjuk a `margin-right:15px`, a másodiknak pedig a `margin-left:15px` tulajdonságot, vagy mind a 30 képpontot az egyik elem margójául adjuk meg, a megfelelőt használva a `margin-right` vagy a `margin-left` tulajdonságok közül.
2. Ehhez a `clear:left` kód szükséges.
3. `padding-top:12px`

Gyakorlatok

- A margó, a belső margó, az igazítás és az úsztatás teljes megértése gyakorlást igényel. A színes téglalapok vagy saját, `<div>` címkék határolta területek használatával próbáljuk ki a térközök és az úsztatás minden lehetséges variációját, mielőtt a következő óra anyagának olvasásába foglalnánk. A következő órán egészében tárgyaljuk a mai órán látott, egyedi objektumokat tartalmazó CSS-dobozokat, vagyis a „dobozmodellt”.
- Ha már úgyis nekikezdtünk a gyakorlásnak, gyakoroljuk a margók használatát minden eddig megismert blokkszintű elemnél. Gyakoroljuk a képek elhelyezését szövegblokkokban, valamint a margóval való körbevételüket is, hogy a szöveg ne egészen a grafika szélénél kezdődjön.



15. ÓRA

A CSS dobozmodellje és az elemek elhelyezése

A lecke tartalma:

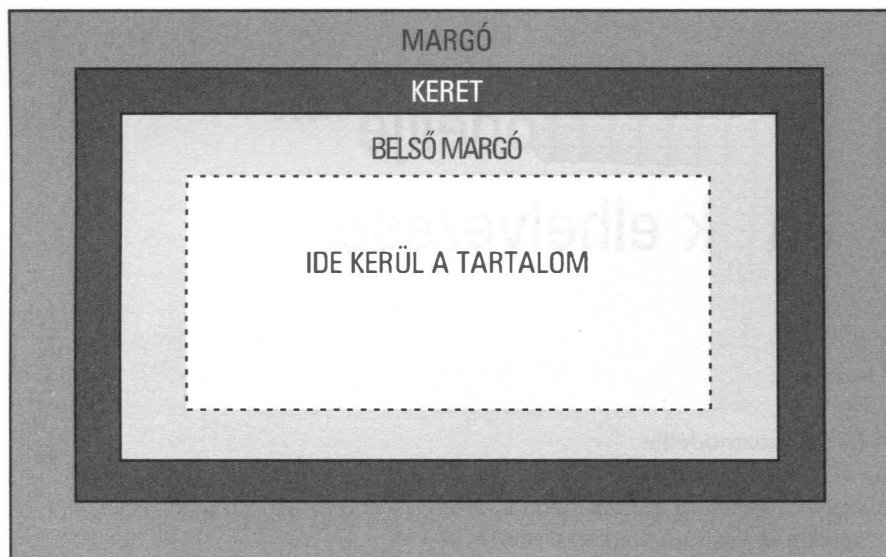
- A CSS dobozmodellje
- Az elemek elhelyezése
- Az egymás tetejére kerülő elemek elhelyezkedésének szabályozása
- A szöveg folyásirányának szabályozása

Néhány alkalommal előkerült már a „CSS dobozmodell” kifejezés. A mai óránkat a dobozmodell mibenlétének megbeszélésével kezdjük, és elmondjuk azt is, hogy az előző órán tanultak hogyan segítenek a dobozmodell megértésében. A dobozmodell megismerését követően nem fogjuk a hajunkat tépni, amikor az elemek nem akarnak a terveinknek megfelelően felsorakozni, vagy ha az elemek kicsit elcsúsznak. Tudni fogjuk már, hogy majdnem minden esetben az a megoldás, hogy valamin – a margókon, a belső margókon, a kereteken – állítunk egy keveset.

A CSS-elemek elhelyezésével kapcsolatban is új ismeretekre teszünk szert – többek között megtanuljuk, hogy miként pakolhatók az elemek egymás tetejére – de nem síkban, függőlegesen, hanem térben gondolkodva. Az óra vége felé pedig az elemeket körülvevő szöveg elhelyezkedésének a `float` tulajdonsággal való befolyásolását fogjuk elsajátítani.

A CSS dobozmodellje

A 15.1. ábrán a CSS dobozmodelljének sematikus rajza látható. A dobozmodell írja le a HTML blokk szintű elemeinek viszonyát a kerettel, a belső margóval és a külső margóval, és azt, ahogy a keret, a belső margó és a külső margó hatása érvényesül. Más szóval azt mondhatjuk, hogy minden elem bír némi belső margóval a tartalom és az elem kerete között. Ezen felül megemlítenénk, hogy a keret nem feltétlenül látható, de a számára fenntartott hely azért ott van, ugyanúgy, ahogy a margó is ott van az elem kerete és az elemet körülvevő egyéb tartalmi elemek között.



15.1. ábra

A CSS dobozmodellje a HTML valamennyi blokk szintű elemét leírja

Íme a dobozmodell egy másik magyarázata, kívülről befelé haladva:

- A (külső) *margó* az elemet körülvevő terület. Nincs színe: mindig átlátszó.
- A *keret* vagy *szegély* az elem körül kap helyet, a mindenkor belső margó külső szélénél. Többféle típusa, vastagsága és színe lehet.

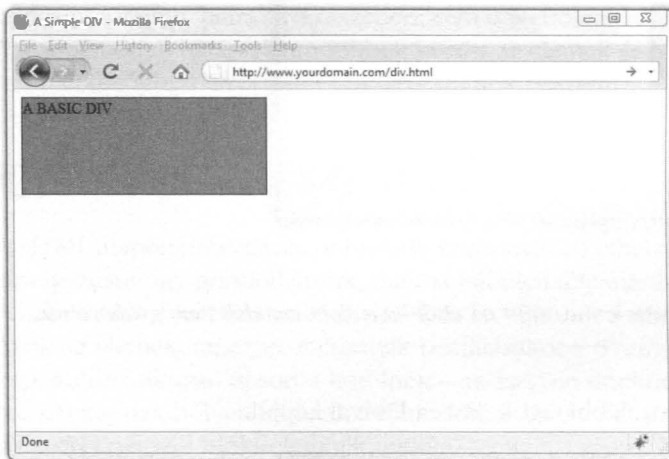
- A *belső margó* vagy *kitöltés* a tartalom körül helyezkedik el, és a tartalom háttérszínét örökli.
- A *tartalmat* a belső margó veszi körül.

Most következik az érdekes rész: ha meg akarjuk tudni egy elem valós magasságát és szélességét, a dobozmodell minden részét számításba kell vennünk. Az előző óráról emlékezhetünk még arra, amikor hiába adtuk meg, hogy a `<div>` címkék határolta terület 250 képpont széles és 100 képpont magas legyen, a téglalap nagyobb lett, különben nem fért volna el benne az alkalmazott belső margó.

Mostanra tudjuk, hogy az elem szélessége és magassága miként állítható be a `width` és a `height` tulajdonság használatával. Az alábbi példa azt mutatja be, hogy miként adható meg olyan, `<div>` címkék határolta terület, amely 250 képpont széles, 100 képpont magas, vörös háttere van, és egyetlen képpont vastagságú fekete keret veszi körül.

```
div {
  width: 250px;
  height: 100px;
  background-color: #ff0000;
  border: 1px solid #000000;
}
```

A fenti kóddal megadott téglalapot a 15.2. ábrán tekinthetjük meg.



15.2. ábra

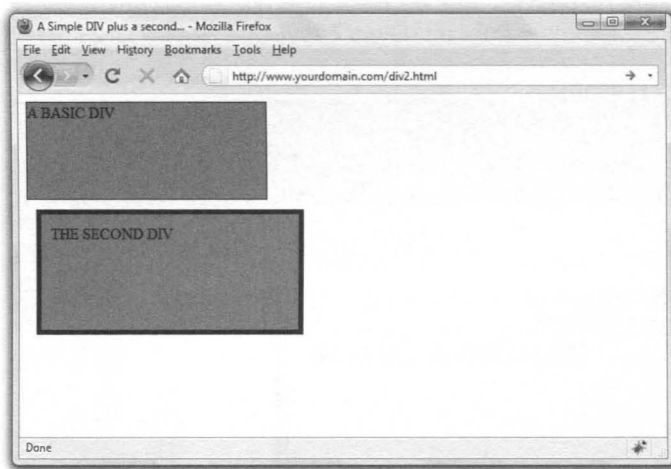
Egyszerű, `<div>` címkék határolta téglalap

Ha megadunk egy ugyanilyen tulajdonságokkal bíró második elemet is, amelynél bizonyos méretű margót, illetve belső margót is beállítunk, kezdjük érzékelni az elem méretének megváltozását. A jelenség oka a dobozmodell.

Ezt a második téglalapot az alábbi kódrészlettel adjuk meg – annyi a különbség, hogy 10 képpontnyi margót, és ugyanennyi belső margót kap az elem.

```
div#d2 {
    width: 250px;
    height: 100px;
    background-color: #ff0000;
    border: 5px solid #000000;
    margin: 10px;
    padding: 10px;
}
```

A 15.3. ábrán látható második téglalapot ugyanolyan magasságúnak és szélességűnek adtuk meg, mint az első, de az elemet körülvevő teljes doboz együttes magassága és szélessége mégis lényegesen nagyobb, mivel a margó és a belső margó is kifejti hatását.



15.3. ábra

A második téglalap ugyanolyan lenne, mint az első, ha a dobozmodell nem gyakorolna hatást a méretére

Az elem teljes *szélességét* az alábbi tagok összeadásával kapjuk:

`width + padding-left + padding-right + border-left + border-right +
➤ margin-left + margin-right`

Az elem teljes *magasságát* pedig az alábbi tagok összeadása adja meg:

`height + padding-top + padding-bottom + border-top + border-bottom +
➤ margin-top + margin-bottom`

A fentiekből következően a második téglalap valós magassága 300 képpont ($250 + 10 + 10 + 5 + 5 + 10 + 10$), a valós szélessége pedig 150 képpont ($100 + 10 + 10 + 5 + 5 + 10 + 10$).



A könyv elejétől a végéig sulykolja a DOCTYPE (dokumentumtípus) megadását – minden példánk így készült. Az Olvasónak nem csak azért tanácsos ezt a gyakorlatot folytatnia, mert a kódja csak így érvényesíthető, hanem azért is, mert van egy igen különös probléma a CSS dobozmodell működésével az Internet Explorer böngészőben: ha nem adjuk meg a dokumentum típusát, akkor az Internet Explorer az elemek szélességét és magasságát nem a szándékainknak megfelelően kezeli. Így az elrendezésünk nem lesz egyforma a különböző böngészőkben. Ne feledjük tehát a DOCTYPE megadását!

Mostanra kezdjük megérteni, ahogy a dobozmodell milyen hatással van a weboldalaink elrendezésére. Tegyük fel, hogy vízszintesen csak 250 képpontnyi helyet foglalhatunk el, és 10 képpontos margót, ugyanekkora belső margót és 5 képpontos keretet szeretnénk az elem minden oldalán. A szándékaink és a lehetőségeink egybehangolása csak akkor lehetséges, ha a `<div>` elem `width` tulajdonságául mindössze 200 képpontot adunk meg, ugyanis $200 + 10 + 10 + 5 + 5 + 10 + 10$ éppen a vízszintes helyünknek megfelelő 250 képponttá adódik össze.

Most, hogy a dobozmodellel kapcsolatosan így kiokosodtunk, ne is felejtsük el róla sem a könyv hátralévő részében, sem a weboldalterveink kialakítása során. A dobozmodell hatással van többek között az elemek és a szöveg elhelyezkedésére. A következőkben erről a két dologról fogunk beszélni.

Nagykanállal az elhelyezésről

A HTML alapértelmezés szerint relatív (viszonyított) elhelyezést alkalmaz. A relatív elhelyezésre úgy gondolhatunk, mintha bábukat állítanánk egy dámajáték táblájára. A bábuk balról jobbra sorakoznak fel, és ha eljutunk a tábla széléig, új sort kezdünk. Azok az elemek, amelyek stílusának beállításakor a `display` tulajdonság `block` értéket kap, automatikusan új sorba kerülnek – az `inline` értékűek ugyanakkor még az adott sorban kapnak helyet, közvetlenül az előttük lévő elem mellett. A `<p>` és a `<div>` elemeket például blokkelemnek tekintjük, míg a `` címke soron belüli elemnek (`inline` értékűnek) számít.

A CSS által támogatott másik elhelyezésmód az *abszolút* elhelyezés. A név onnan ered, hogy ezzel a módszerrel a HTML-tartalom adott oldalon belüli pontos helye adható meg. Bár az abszolút elhelyezés felruház bennünket azzal a szabadsággal, ami az elem kívánt helyének pontos megadását teszi lehetővé, az elem helye az oldalon lévő szülő-

elemekhez képest viszonyított. Más szóval, az abszolút elhelyezés azt teszi lehetővé, hogy a szülőelem területén belül pontosan megadjuk az elemet jelképező téglalap helyét – és ez azért eléggé eltér a relatív elhelyezéstől.

Most, hogy módunkban áll oda tennünk az elemeket az oldalon belül, ahová tenni szeretnénk őket, könnyen átfedések okozta problémákba ütközhetünk – azaz előfordulhat, hogy az egyik elem a másik által már elfoglalt helyre kerül. Semmi sem gátolja meg, hogy az elemek abszolút helyét úgy adjuk meg, hogy az elemek átfedjék egymást. Az ilyen esetekben a CSS az elemek z-indexe alapján állapítja meg, hogy melyik elem kerül előre, és melyik hátulra. A z-indexről az óra későbbi részében bővebben fogunk tanulni. Most inkább nézzük meg, hogy miként szabályozható pontosan, hogy egy stílusszabály relatív vagy abszolút elhelyezést használjon.

Az egy adott stílusszabály által használt elhelyezéstípust (relatív vagy abszolút) a `position` tulajdonság adja meg, amelynek a lehetséges értékei: `relative` és `absolute`. Az elhelyezés típusának megadását követően az alábbi tulajdonságokkal adhatjuk meg a pontos elhelyezést:

- `left`: Az elem helyének eltolása bal felé.
- `right`: Az elem helyének eltolása jobb felé.
- `top`: Az elem helyének eltolása felfelé.
- `bottom`: Az elem helyének eltolása lefelé.

Az gondolhatnánk, hogy a fenti tulajdonságok csak az abszolút elhelyezés során értelmezhetők, de az a helyzet, hogy mindkét elhelyezéstípusnál használjuk őket. A relatív elhelyezés használatakor az elem helyét az elem eredeti helyétől való eltolással adjuk meg. Azaz, amikor egy elem `left` tulajdonságát 25 képpontban adjuk meg, akkor az elem bal oldala 25 képponttal tolódik el az eredeti – viszonyított – helyéhez képest. Ezzel szemben az abszolút elhelyezés használatakor az eltolás a szóban forgó elem szülőeleméhez képest értendő. Ilyenkor a 25 képpontban megadott `left` tulajdonság hatására az elem bal széle 25 képponttal kerül jobbra a szülőelem bal szélétől. Hasonló értékkel használva a `right` tulajdonságot, az elem elhelyezése olyan lenne, hogy az elem *jobb* oldala kerülne 25 képpontnyi távolságra a szülőelem *jobb* szélétől.

15.1. példa A relatív elhelyezést szemléltető négy színes téglalap

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Positioning the Color Blocks</title>
    <style type="text/css">
      div {
```

```
position:relative;
width:250px;
height:100px;
border:5px solid #000;
color:black;
font-weight:bold;
text-align:center;
}
div#d1 {
  background-color:red;
}

div#d2 {
  background-color:green;
}

div#d3 {
  background-color:blue;
}

div#d4 {
  background-color:yellow;
}
</style>

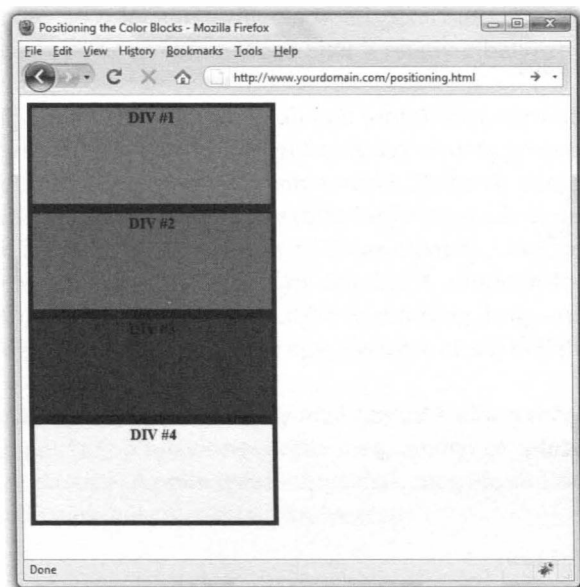
</head>
<body>
  <div id="d1">DIV #1</div>
  <div id="d2">DIV #2</div>
  <div id="d3">DIV #3</div>
  <div id="d4">DIV #4</div>
</body>
</html>
```

Az elhelyezés működését a színes téglalapos példához visszatérve tesszük meg.

A 15.1. példában a négy színes téglalap relatív elhelyezésű. Ahogy a 15.4. ábrán látható, a téglalapok egymás alá kerülnek.

A stíluslap `<div>` elemekre vonatkozó bejegyzése a `position` tulajdonság értékéül a `relative` értéket adja meg. Minthogy a további stílusszabályok a `<div>` elemekre vonatkozó beállítást öröklik, átveszik a relatív beállítást. A többi stílusszabály a különféle háttérszínek megadásán kívül valójában semmit sem tesz.

A 15.4. ábrán láthatjuk, hogy a téglalapok egymás után helyezkednek el, és a relatív elhelyezéstől éppen ezt várjuk. Tegyük érdekesebbé a dolgot – elvégre ezért vagyunk itt –, és cseréljük az elhelyezést abszolútra, pontosan megadva a színes téglalapok helyzetét. A 15.2. példakódban a stíluslap bejegyzéseit úgy változtattuk meg, hogy a színes téglalapok elhelyezése abszolút módon történjen.



15.4. ábra

A színes téglalapok függőlegesen, egymás alatt helyezkednek el

15.2. példa A színes téglalapok abszolút elhelyezéssel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Positioning the Color Blocks</title>
    <style type="text/css">
      div {
        position:absolute;
        width:250px;
        height:100px;
        border:5px solid #000;
        color:black;
        font-weight:bold;
        text-align:center;
      }
      div#d1 {
        background-color:red;
        left:0px;
        top:0px;
      }
    </style>
  </head>
  <body>
    <div id="d1">DIV #1</div>
    <div id="d2">DIV #2</div>
    <div id="d3">DIV #3</div>
    <div id="d4">DIV #4</div>
  </body>
</html>
```

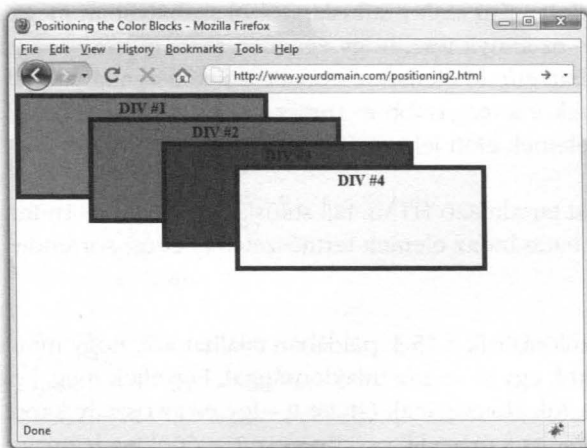


```

div#d2 {
    background-color:green;
    left:75px;
    top:25px;
}
div#d3 {
    background-color:blue;
    left:150px;
    top:50px;
}
div#d4 {
    background-color:yellow;
    left:225px;
    top:75px;
}
</style>
</head>
<body>
<div id="d1">DIV #1</div>
<div id="d2">DIV #2</div>
<div id="d3">DIV #3</div>
<div id="d4">DIV #4</div>
</body>
</html>

```

A fenti stíluslap az `absolute` kulcsszót adja a `position` tulajdonság értékéül: erre van szükség, mert azt szeretnénk, hogy a stíluslap abszolút elhelyezést használjon. Minden leszármazott `<div>`-stílusszabály beállítja a `left` és a `top` tulajdonságot. A fenti szabályok mindegyike úgy adja meg az elemek helyét, hogy azok – a 15.5. ábrán látható módon – átfedésben legyenek.



15.5. ábra

A színes téglalapok megjelenítése abszolút elhelyezéssel

Ez már igazi elrendezés! A 15.5. ábrán látható, hogy az abszolút elhelyezés alkalmazásával az elemek éppen oda kerülnek, ahová tenni szerettük volna őket. Az ábra arra is fényt derít, hogy milyen egyszerű az elemek átfedéses elrendezésének megoldása. Az Olvasóban felmerülhet a kérdés: honnan tudja a webböngésző, hogy átfedéskor melyik elem kerüljön előre, és melyik hátulra? A következő részben az átfedő elemek sorrendjéről lesz szó.

Az egymás tetejére kerülő elemek elhelyezkedésének szabályozása

Vannak olyan helyzetek, amikor pontosan szeretnénk megadni azt a sorrendet, ahogy a weboldal egyes elemei átfedik egymást. Az elemek sorrendjének a takarási sorrendet is befolyásoló kialakítása a *z-index* tulajdonság használatával válik lehetővé. Bár a *z-index* név esetleg furcsának tűnhet, mindössze a harmadik dimenzió (*Z*) fogalmára utalunk vele. Ez a dimenzió mintegy a képernyő mögé mutat, kiegészítve a képernyő szélességén (*X*), és magasságán (*Y*) alapuló dimenziókat. A *z-index*re tekinthetünk úgy is, mint arra a számra, amely egy újságköteg belsejében adja meg egy adott újság helyét. A köteg teteje felé lévő újságok *z-indexe* magasabb a köteg alján lévő újságokénál. Ehhez hasonlóan a magasabb *z-indexű* átfedő elem az alacsonyabb *z-indexű* elem előtt jelenik meg.



Az az elem, amelynek bármilyen beállított *z-index* értéke van, mindig a szülőelem előtt jelenik meg, a stílussabályban megadott *z-index* értékétől függetlenül.

A *z-index* tulajdonság arra való, hogy egy stílussabály relatív mélységi (*Z*) helyzetét megadjuk. A *z-index* értékül adott szám csak a stíluslap többi szabályában megadott értékekhez képest értelmezhető – ez annyit tesz, hogy egyetlen szabály *z-index* tulajdonsága keveset mond. Ha azonban átfedő elemekre vonatkozó stílussabályoknál adunk meg *z-index* értékeket, akkor a magasabb *z-index* értékkel bíró elemek az alacsonyabb *z-index* értékű elemek előtt jelennek meg.

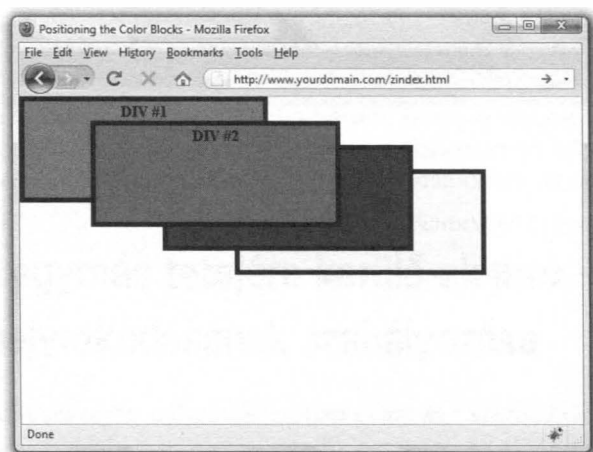
A 15.3. példa a színes téglalapokat tartalmazó HTML-fájl stíluslapján olyan *z-index* értékeket használ, amelyeknek a hatására az elemek természetes átfedési sorrendje megváltozik.

A fenti kód mindössze annyiban különbözik a 15.3. példában találhatóától, hogy minden számozott *div* stílusosztályt elláttunk egy *z-index* tulajdonsággal. Figyeljük meg, hogy az első számozott osztály *z-index* tulajdonságának értéke 0 – így ez az osztály kapja a legalacsonyabb *z-index* értéket –, és a második osztály *z-index* értéke a legmagasabb. A 15.6. ábrán a színes téglalapok a fenti stíluslapot használva jelennek meg; szépen látható, hogy a *z-index* tulajdonság értéke milyen hatással van a megjelenő tartalomra, lehetővé téve az elemek átfedésének pontos szabályozását.

15.3. példa *A színes téglalapok megjelenésének megváltoztatása a z-index tulajdonság segítségével*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Positioning the Color Blocks</title>
    <style type="text/css">
      div {
        position:absolute;
        width:250px;
        height:100px;
        border:5px solid #000;
        color:black;
        font-weight:bold;
        text-align:center;
      }
      div#d1 {
        background-color:red;
        left:0px;
        top:0px;
        z-index:0;
      }
      div#d2 {
        background-color:green;
        left:75px;
        top:25px;
        z-index:3;
      }
      div#d3 {
        background-color:blue;
        left:150px;
        top:50px;
        z-index:2;
      }
      div#d4 {
        background-color:yellow;
        left:225px;
        top:75px;
        z-index:1;
      }
    </style>
  </head>
  <body>
    <div id="d1">DIV #1</div>
    <div id="d2">DIV #2</div>
    <div id="d3">DIV #3</div>
    <div id="d4">DIV #4</div>
  </body>
</html>
```



15.6. ábra

A színes téglalapok megjelenését a z-index tulajdonsággal módosítjuk

Bár a példák színes téglalapokkal dolgoznak – mégpedig egyszerű, `<div>` címkékkel megadott téglalapokkal –, a `z-index` tulajdonság minden HTML-tartalomra hatással lehet, ideértve a képeket is.

A szöveg folyásirányának szabályozása

Most, hogy láttunk néhány olyan példát, amelyben az elemeket egymáshoz viszonyítva helyeztük el, és láttunk olyanokat is, ahol abszolút elhelyezést használtunk, itt az ideje újra elővennünk az elemeket körülvevő szöveges tartalmat. Alapvető fontosságú az *aktuális sor* fogalma, amely egy az elemek adott oldalon való elhelyezésére szolgáló, láthatatlan sort takar. Részben ez a sor felel az elemek elhelyezkedésének szabályozásáért: akkor jut szerephez, amikor az elemek vízszintes és függőleges elhelyezése egymáshoz viszonyított. Az elhelyezésre váró elemek között említhetjük meg az oldalra kerülő szöveget is. Amikor a szövegben más elemeket – például képeket – helyezünk el, lényeges annak szabályozása, ahogy a szöveg a többi elem körül elhelyezkedik.

Az alábbi stílustulajdonságok közül kettővel már a 14. órán is találkoztunk. Az alábbi lista olyan tulajdonságokat sorol fel, amelyekkel a szöveg elhelyezkedése szabályozható.

- `float`: Azt adja meg, hogy a szöveg miként folyja körbe az elemet.
- `clear`: Megakadályozza, hogy a szöveg körbefolyja az elemet.
- `overflow`: A szöveg túlfolyását szabályozza olyan esetekben, amikor az elem túl kicsi ahhoz, hogy a teljes szöveg elférjen benne.

A `float` tulajdonságot használjuk annak megadására, hogy a szöveg miként folyhatja körbe az elemeket. Értéke `left` (bal) és `right` (jobb) lehet. Ezek az értékek határozzák meg, hogy az elemet a szöveghez képest merre kell elhelyezni. Azaz egy kép `float` tulajdonságának `left` értéket adva a kép a folyó szöveg bal oldalán kap helyet.

Amint azt az előző leckében megtanultuk, a `clear` tulajdonság használatával akadályozhatjuk meg azt, hogy egy elem köré szöveg kerüljön. A `clear` tulajdonság a `none` (egyik sem), a `left`, a `right`, és a `both` (mindkettő) értéket veheti fel. A tulajdonság alapértelmezett értéke a `none`, amely azt jelzi, hogy a szöveg minden további nélkül körbefolyhatja az elemet. A tulajdonság `left` értékének hatására a szöveg nem kaphat helyet az elem bal oldalán – a `right` érték pedig a szöveget az elem jobb oldalához nem engedi oda. A `both` érték azt jelzi, hogy az elem egyik oldalára sem kerülhet szöveg.

Az `overflow` tulajdonság a szöveg túlfolyását szabályozza, vagyis azt az esetet, amikor a szöveg nem fér el a neki szánt téglalapban – ilyesmi akkor eshet meg, amikor egy elem `width` (szélesség), illetve `height` (magasság) tulajdonságának túl alacsony értéket adunk. Az `overflow` tulajdonság lehetséges értékei: `visible` (látható), `hidden` (rejtett) és `scroll` (görgetés). A `visible` beállítás hatására az elem automatikusan akkorára növekszik, hogy az egyébként túlfolyó szöveg elférjen benne – ez a tulajdonság alapértelmezett beállítása. A `hidden` érték az elem méretét változatlanul hagyja, felválalva azt, hogy a túlfolyó szöveg láthatatlan marad. A legérdekesebb érték alighanem a `scroll` – ennek hatására a szöveg körül gördítősávok jelennek meg, ami lehetővé teszi, hogy a szöveget mozgatva az egészet elolvassuk.

Összefoglalás

Az órát egy igen fontos téma tárgyalásával kezdtük: a CSS dobozmodelljéről volt szó, és arról, hogy miként számítható ki a margók, belső margók és keretek vastagságának figyelembe vételével az elemek szélessége és magassága. Ennek folytatásaként megbirkóztunk az elemek abszolút elhelyezésének megadásával, amit a `z-index` tulajdonság használatával való elhelyezés megismerése követett. Végül megismertünk néhány, az oldalon belül a szöveg folyásirányát szabályozó takaros tulajdonságot.

Kérdezz-felelek

- K: *Miként dönthető el, hogy a relatív vagy az abszolút elhelyezést érdemes használni?*
- V: Bár a relatív, illetve az abszolút elhelyezés használatát illetően nincsenek kőbe vésett irányelvek, alapvetően elmondható, hogy abszolút elhelyezést csak olyankor kell használnunk, amikor a tartalom elhelyezésének módját pontosabban szeretnénk szabályozni. Ez a kijelentés abban a tényben leli magyarázatát, hogy az abszolút elhelyezés esetében a pontos képpontot is megadhatjuk, míg a relatív elhelyezést használva a tartalom elhelyezésének eredménye lényegesen kevésbé megjósolható. Mindezzel nem azt szeretnénk mondani, hogy a relatív mód nem lehet tökéletesen elegendő, ha egy oldal elemei helyének megadása a feladat, pusztán annyit jelentünk ki, hogy az abszolút elhelyezés pontosabb. Mindennek természetes velejárója, hogy az abszolút elrendezés elvileg érzékenyebb a képernyőméret változásaira – ezen pedig nem tudunk segíteni.
- K: *Ha két átfedő elem z-index tulajdonságának nem adunk értéket, akkor miként jósolható meg, hogy melyik elem kerül előre?*
- V: Amikor az átfedő elemek z-index tulajdonsága nincs beállítva, a weboldal későbbi részében megjelenő elem kerül előre. Ezt akkor tudjuk egyszerűen megjegyezni, ha elképzeljük, amint a webböngésző a HTML-dokumentumot beolvasva kirajzolja az egyes elemeket: a később beolvasott elemeket a korábban beolvasottakra rajzolja rá.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckeiben szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Mi a különbség a relatív és az abszolút elhelyezés között?
2. Mi a neve annak a CSS-tulajdonságnak, amelyik az elemek átfedésének módját szabályozza?
3. Írjuk meg azt a HTML-kódot, amelyik pontosan a böngészőablak bal felső sarkától kezdve megjeleníti a „Where would you like to” szöveget, és pontosan a saroktól 80 képpontra lefelé és 20 képpontra balra kiírja a „GO TODAY” szavakat!

Válaszok

1. Relatív elhelyezés használatakor a tartalom az oldal alakulásának megfelelően jelenik meg, vagyis minden elem az őt a HTML-kódban fizikailag megelőzőt követi. Az abszolút elhelyezés ugyanakkor lehetőséget ad a tartalom pontos elhelyezésére az oldalon belül.
2. Az elemek átfedésének módját a z-index CSS-tulajdonság szabályozza.
3. Használhatjuk például az alábbi kódot:

```
<span style="position:absolute;left:0px;top:0px">
Where would you like to</span>
<h1 style="position:absolute;left:80px;top:20px">GO TODAY?</h1>
```

Gyakorlatok

- Gyakoroljuk a CSS-dobozmodell útvesztőiben való eligazodást: készítsünk több, különböző margójú, belső margójú és keretű elemet, és tanulmányozzuk az említett tulajdonságoknak az elem szélességére és magasságára gyakorolt hatását.
- Keressünk egy nekünk szimpatikus képsort, és abszolút elrendezés vagy akár a z-index tulajdonság használatával rendezzük őket egyfajta képgalériába. Próbálkozzunk a képek egy adott alakzatba – négyzetbe, háromszögbe, körbe – való rendezésével.

16. ÓRA



Szemrevalóbb listák és egyebek – a CSS használatával

A lecke tartalma:

- A CSS-dobozmodell hatása a listákra
- A felsorolásjel testreszabása
- A listaelemek használata és képtérkép készítése a CSS segítségével

Az 5. órán megismerkedtünk a HTML-listák három típusával, a 14.-en pedig megtanultuk a külső és belső margók használatát, valamint az elemek elrendezésének módját. A mai lecke során azt tanuljuk meg, hogy a margók, belső margók és elrendezések miként alkalmazhatók a különféle HTML-listák létrehozásakor. Így egy olyan, pusztán a HTML-re és a CSS-re támaszkodó módszer kerül a kezünkbe, amelynek a segítségével igen érdekes elemekkel gazdagíthatjuk a weboldalunkat.

Egészen pontosan a listaelemek kinézetének módosításáról lesz szó – az 5. órán megismert `list-style-type` tulajdonságtól eltérő módszerekkel –, valamint arról, hogy a CSS segítségével formázott listák használatával miként válthatók le a 11. órán

megismert ügyféloldali képtérképek. Gyakorlatban fogunk alkalmazni sok, az eddigiek során megismert CSS-tulajdonságot. A mai órán megszerzett ismeretek közvetlen előkészítői a 17. órának és az akkor elkészítendő műveinknek.

Ismétlés – a HTML listái

Amint azt az 5. órán megtanultuk, a HTML három alapvető listatípust kínál. Mindhárom lista némileg eltérően jeleníti meg a tartalmat – a listatípustól és a lista szöveggörnyezetétől függően.

- A *rendezett lista* olyan, behúzással bíró lista, amely számokat vagy betűket jelenít meg az egyes listaelemek előtt. A rendezett listát az `` és `` címkékkel fogjuk közre, a listaelemeket pedig az `` és `` címkepárral határoljuk. Ezzel a listatípussal sokszor jelenítünk meg számozott lépéseket, ezen felül a tartalom különböző szintjeinek jelzésére is használható.
- A *rendezetlen lista* olyan, behúzással bíró lista, amely korongot vagy egyéb jelet jelenít meg az egyes listaelemek előtt. A rendezetlen listát az `` és `` címkékkel fogjuk közre, a listaelemeket pedig az `` és `` címkepárral határoljuk. A listatípushoz kapcsolódó látvány rövid, pontosan megfogalmazott információelemekre hívja fel a figyelmet.
- A *meghatározás-lista* általában kifejezések és azok jelentésének megjelenítésére szolgál, így magán a listán belül alakulhatnak ki az információ rétegei. Ez a lista a rendezett listához hasonló, de számozást nem látunk. A meghatározás-listát a `<dl>` és `</dl>` címkék fogják közre. A kifejezéseket a `<dt>` és `</dt>`, a meghatározásokat a `<dd>` és `</dd>` címkék határolják.

Amikor a tartalom szükségessé teszi, a rendezett és rendezetlen listákat egymásba ágyazhatjuk. Az egymásba ágyazott listák a tartalomnak egyfajta rétegződését valósítják meg – így ajánlatos csak akkor használnunk ezeket az elemeket, amikor a tartalom valóban rendelkezik a megjeleníteni kívánt rétegződéssel. Ilyenek például a tartalmat körvonalazó vázlatok, illetve a tartalomjegyzékek. Másfelől, ahogy azt a 17. órán látjuk majd, az egymásba ágyazott listáknak olyan esetekben is hasznát vesszük, amikor a webhelyünk egymás alá szervezett oldalai között való eligazodást segítik.



Pár régebbi böngésző eltérően kezeli a (külső) margót és a belső margót (kitöltést) – főleg, ha azok listák, illetve listaelemek körül kapnak helyet. Mindazonáltal e könyv írásának idején az ebben és a többi fejezetben bemutatott HTML és CSS nyelvű kódok a jelentősebb webböngészők (Apple Safari, Google Chrome, Microsoft Internet Explorer, Mozilla Firefox és Opera) mindegyikében egységes megjelenítést eredményeznek. Természetesen – és az előbbi kijelentés dacára – a weboldalunkat az internetes közzétételt megelőzően továbbra is ellenőriznünk kell valamennyi böngészőben, de a stíluslappal történő barkácsolások kora – ami a böngészők különbözősége miatt vált egykor szükségessé – lassan a múltba vész.

A CSS-dobozmodell hatása a listákra

A listák kezelésére szolgáló stíuselemek közül a `list-style-image`, a `list-style-position` és a `list-style-type` nevűt említjük meg. Az első arra való, hogy a listaelemek jelzésére képet használjunk, a második azt adja meg, hogy a felsorolásijel hol helyezkedjen el, a harmadik pedig a felsorolásijel típusát határozza meg. Míg az előző stíuselemek a lista, illetve a listaelemek felépítését szabályozzák, a `margin` (margó), a `padding` (belső margó), a `color` (szín) és a `background-color` (hátterszín) stílusjellemző segítségével a listák megjelenésének finomabb szabályozására nyílik lehetőségünk.

A 14. órán olvashattuk, hogy a tartalom és az elem kerete között minden elem esetében van egy kevés belső margó, és azt is megtudtuk, hogy mindig van valamennyi margó az elem kerete és a szomszédos tartalom között. Ez igaz a listákra is, így amikor a listák formázását végezzük, nem feledkezhetünk meg arról, hogy a listát valójában kétféle elem alkotja: a szülőelem (`` vagy ``) és maguk az egyedi listaelemek. Ezen elemek mindegyike rendelkezik a stíluslap segítségével szabályozható margóval és belső margóval.

A mai óra példáiból kiderül, hogy a CSS különféle stílusai milyen hatással vannak a HTML-listák, illetve a listák elemeinek megjelenésére. Ha nem felejtkezünk el az előbb említett különbségekről, képessé válunk a listák megjelenésének mindenre kiterjedő szabályozására, és ahogy a 17. óra gyakorlatainál tapasztalni fogjuk, a listák segítségével a webhelyek navigációs menüjét is megszépíthetjük.

A 16.1. példa egy egyszerű, háromelemű listát alakít ki. A példában a rendezetlen lista (az `` címkék határolják) kék hátteret, fekete keretet, és 100 képpontban rögzített szélességet kap – amint az a 16.1. ábrán is látható. A listaelemek (amelyeket a `` címkék határolnak) háttere szürke, a kerete pedig sárga. A listaelemek szövege és a felsorolásijel (a korong) fekete.

16.1. példa Egyszerű színes, keretezett lista létrehozása

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

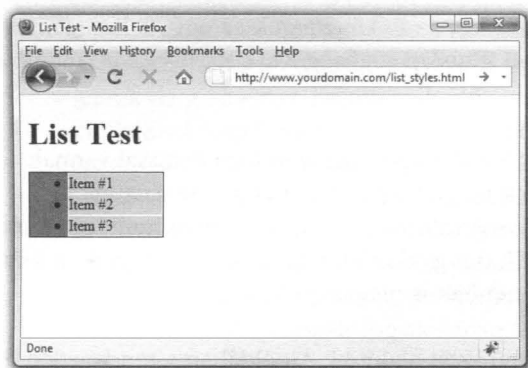
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>List Test</title>
    <style type="text/css">
      ul {
        background-color: #6666ff;
        border: 1px solid #000000;
        width:100px;
      }
```

```

    li {
        background-color: #cccccc;
        border: 1px solid #ffff00;
    }
</style>
</head>

<body>
<h1>List Test</h1>
<ul>
    <li>Item #1</li>
    <li>Item #2</li>
    <li>Item #3</li>
</ul>
</body>
</html>

```



16.1. ábra

A lista és a listaelemek színt és keretet kapnak



Azt, hogy a különböző böngészők hogyan jelenítik meg a `padding-left` (bal belső margó) tulajdonság alapértelmezett értékét, egy a 16.1. példában ismertetetthez hasonló egyszerű tesztfájl segítségével próbálhatjuk ki. Töltsük be a fájlt, majd az `ul` kijelölőnél helyezzük el a `padding-left: 40px;` bejegyzést. Ezt követően töltsük újra az oldalt, és ha a megjelenő lista nem változik, akkor tudjuk, hogy a böngészőnk a `padding-left` tulajdonság alapértelmezett értékeként 40 képpontot használ.

Ahogy a 16.1. ábrából kiderül, az `` címke hozza létre azt a dobozt, amelybe maguk a listaelemek kerülnek. Esetünkben az említett doboz teljes egészében kék háttérrel kap. Figyeljük meg azt is, hogy az egyes listaelemek (a példánkban sárgával keretezve és szürke háttérrel) nem nyúlnak el egészen az `` címkék meghatározta doboz bal oldaláig.

A jelenség oka, hogy a böngészők automatikusan elhelyeznek valamekkora helyet az `` elem bal oldalán. A hely nem a margin alapértelmezett nagyságát növeli – az a dobozon kívül jelenne meg. A helyet a belső margóhoz csapják hozzá – ez ugye a doboz belső oldalán található –, méghozzá csak a bal oldalon. A belső margó értéke így megközelítőleg 40 képpont lesz.

A belső margó alapértelmezett értéke független a lista típusától. Ha a stíluslapunkon elhelyezzük az alábbi sort – ezzel felsorolásjelek nélküli listát alakítva ki –, azt fogjuk látni, hogy a belső margó értéke nem változik (lásd a 16.2. ábrát):

```
list-style-type: none;
```



16.2. ábra

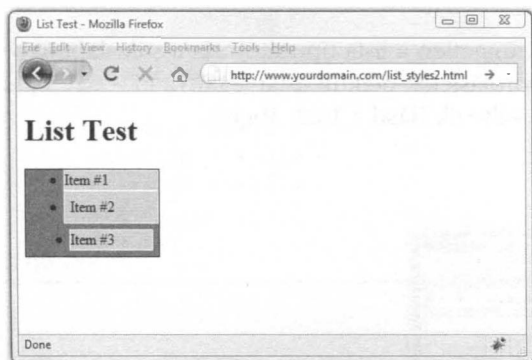
Az alapértelmezett bal belső margó a felsorolásjelek elhagyásakor is változatlan marad

Listát tartalmazó oldalelrendezés készítésekor az elemeket ide-oda helyezgetve az oldalon kísérletezgezzünk a belső margóval. Erre a célra bármelyik listatípus megfelel. Pusztán az a tény, hogy a margó nem kap alapértelmezett értéket, még nem jelenti azt, hogy mi nem adhatunk meg margót a listáinknak. Az `ul` kijelölő margin tulajdonságának értéket adva újabb olyan eszköz kerül a kezünkbe, amivel az elrendezést szabályozhatjuk.

Ne feledjük, hogy mindezidáig csak a teljes listát megadó elemekkel dolgoztunk, a listaelemeket még nem formáztuk. A 16.1. és 16.2. ábrán a szürke háttér és a sárga keret nem enged arra következtetni, hogy a margin vagy a padding tulajdonság alapértelmezett értéket kapna. A 16.3. ábra azt mutatja be, hogy milyen hatásokat érhetünk el, ha a margót és a belső margót nem a teljes listánál, hanem maguknál a listaelemeknél adjuk meg.

Az első listaelem a viszonyítási alap: sem margót, sem belső margót nem állítottunk be rá. A második listaelemben már érvényesül a `style="padding: 6px;"` kód hatása, és a 6 képpontnyi belső margó minden oldalon láthatóvá válik – a szöveg és az elemet

határoló sárga keret között. Figyeljük meg, hogy a felsorolásjel ugyanott maradt, ahova az első listaelem esetében is került. A lista harmadik eleménél a `style="margin: 6px; "` kód hatására az elem körül 6 képpontnyi margó jön létre – ennek köszönhetően válik láthatóvá az `` elem háttere.



16.3. ábra

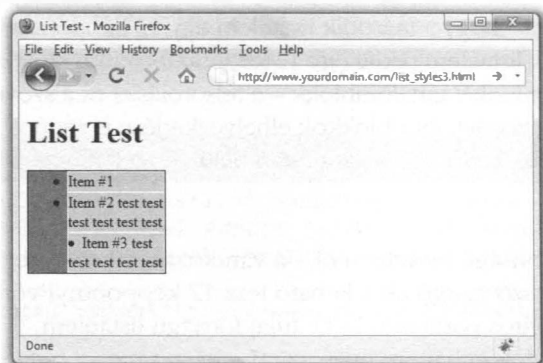
Különböző margó- és belsőmargó-beállítással bíró listaelemek

A felsorolásjelek elhelyezése

A margót és a belső margót érintő eddigi fejtegetéseink újabb kérdéseket vetnek fel, nevezetesen a – nem feltétlenül használt – felsorolásjeleket, illetve azt, hogy ha a szöveg körbeveszi a felsorolásjeleket, akkor azt miként tegye. A `list-style-position` tulajdonság alapértelmezett értéke az `outside` (kint). Ez annyit tesz, hogy a felsorolásjelek és a számok a szöveg bal oldalára kerülnek, és a `` címkepár meghatározta téglalapon kívül maradnak. Ha egy adott listaelem szövegét tördelni kell, akkor a tördelt szöveg is belül marad a téglalapon, az elem bal oldalához rendezve.

Ha azonban a `list-style-position` tulajdonság az `inside` értéket kapja, a felsorolásjelek a `` címkepár által meghatározott téglalapon belülre kerülnek. Ilyenkor nem csak a felsorolásjelek behúzása nő meg – lényegében a szöveg részévé alakulnak –, hanem a tördelt szöveg is befolyik az egyes felsorolásjelek alá.

A 16.4. ábrán mind az `outside`, mind az `inside` értékű `list-style-position` tulajdonsággal készült példát megtekinthetjük. A 16.1. példa és a 16.4. ábrán látható eredmény előállítására használt kód között – az „Item #2” és az „Item #3” bejegyzést tartalmazó sorok szövegének bővítésétől eltekintve – mindössze annyi a különbség, hogy a második listaelem kódját a `style="list-style-position: outside; "`, a harmadikét pedig a `style="list-style-position: inside; "` sorral bővítettük.

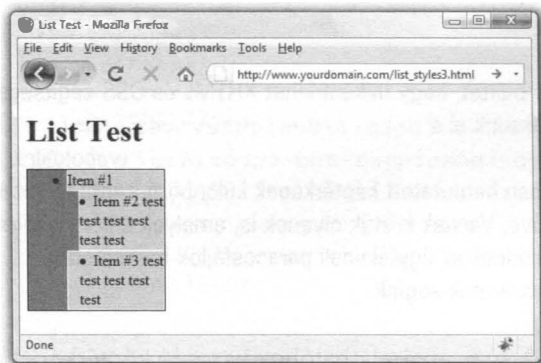


16.4. ábra

A list-style-position tulajdonság outside és inside értékét használva adódó különbség

A második listaelem bővítésére használt szöveg megmutatja a szöveg tördelését olyan esetekben, amikor a lista túl keskeny ahhoz, hogy a szöveg egyetlen sorban elférjen. Az ábrán látható eredménnyel megegyezőt értünk volna el akkor is, ha a `style="list-style-position: outside;"` kódot nem használtuk volna, mivel a `list-style-position` tulajdonság ezt az alapértelmezett értéket veszi fel, ha mi magunk nem adunk értéket neki.

Az `inside` használatakor fellépő különbség remekül megfigyelhető. A harmadik listaelem szövege és felsorolásjele egyaránt a sárgával keretezett szürke területre, azaz az elemen belülre került. A margó és a belső margó listaelemekre gyakorolt hatása más az `inside` értékre beállított `list-style-position` tulajdonság esetében – lásd a 16.5. ábrát.



16.5. ábra

A margó és a belső margó másképp jelenik meg, ha a list-style-position tulajdonság értéke inside

A 16.5. ábrán a második és a harmadik listaelem `list-style-position` tulajdonságának értéke egyaránt `inside`. A különbség az, hogy a második listaelem egy 12 képpontos `margin-left` (bal margó), a harmadik listaelem pedig egy 12 képpontos `padding-left` (bal belső margó) értéket kapott. Bár mindkét tartalomblokk – a felsorolásijel és a szöveg – esetében körbeveszi a szöveg a felsorolásijelet, és a blokkok elhelyezkedése is azonos a listaelemet jelző szürke területen belül, a változtatások a listán belüli listaelemre fejtik ki a hatásukat.

A 12 képpontos `margin-left` tulajdonságú listaelemnél – a várakozásainknak megfelelően – a listaelemet körbevevő átlátszó margó alatt látható lesz 12 képpontnyi vörös szín is. Ehhez hasonlóan, a 12 képpontos `padding-left` tulajdonságú listaelem 12 képpontnyi szürke hátteret – a listaelem háttere ilyen színű – jelenít meg a tartalmat megelőzően. A belső margó az elem belsejébe kerül, a margó pedig kívülről veszi körbe az elemet.

Ha megértjük, hogy a margó és a belső margó hogyan befolyásolja a listaelemeket, illetve a listaelemeket tartalmazó listát, képessé válunk – pusztán a CSS használatával – külső képek igénybe vétele nélkül kialakítani a webhelyünk navigációs elemeit. A 17. órán a függőleges és vízszintes navigációs menük kialakításának megismerésén túl megtanulunk lenyíló menüket is készíteni.

Képtérképek készítése listaelemek és CSS használatával

A 11. órán megtanultuk, hogy a HTML `<map/>` címkéjének használatával miként hozhatunk létre ügyféloldali képtérképeket. A képtérképek segítségével megtehetjük azt, hogy a képen belül megadott területhez rendelünk hivatkozást, és így nem kell a képet felszabdalnunk, majd újra összeállítanunk, miután a hivatkozásokat hozzárendeltük az egyes darabokhoz. Képtérképeket pusztán szabályos XHTML- és CSS-kóddal is kialakíthatunk.



Ha szívesen fogadnánk néhány ötletet, hogy miként lehet XHTML és CSS segítségével képtérképeket készíteni, látogassunk el a <http://designreviver.com/tutorials/css-image-map-techniques-and-tutorials/> weboldalra. Az itt fellelhető segédanyagokban bemutatott képtérképek különböző szintű felhasználói beavatkozást tesznek lehetővé. Vannak köztük olyanok is, amelyek a könyv kereteit meghaladó mértékben támaszkodnak az ügyféloldali parancsfájlok használatára. Az érdeklődőket kimerítő magyarázatok segítik.

A 16.2. példában ismertetett kód egy a 16.6. ábrán láthatóhoz hasonló képtérképet állít elő (a 16.2. példa kódja nem rajzolja meg az ábrán látható piros kereteket. Ezek csak azért kerültek az ábrára, hogy a szempontunkból érdekes területeket kiemeljék).

A webböngésző által a kód alapján elkészített oldal egy egyszerű, képet megjelenítő weboldalnak tűnik. A dolog akkor kezd érdekesebbé válni, amikor az egérmutatóval egy forrópont fölé érünk.

16.2. példa *Képtérkép létrehozása CSS segítségével*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

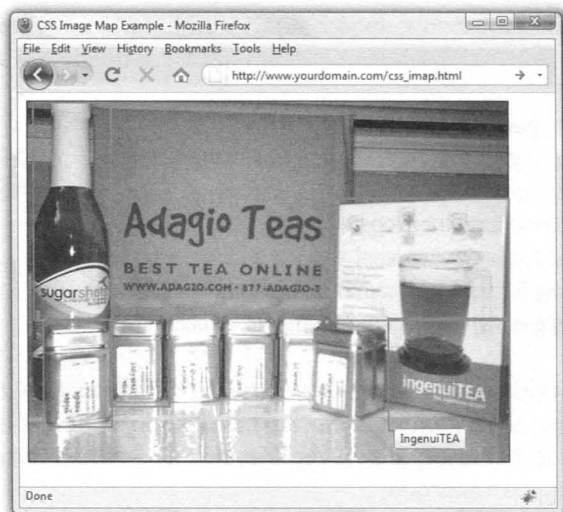
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>CSS Image Map Example</title>
    <style type="text/css">
      #theImg {
        width:500px;
        height:375px;
        background:url(tea_shipment.jpg) no-repeat;
        position:relative;
        border: 1px solid #000000;
      }
      #theImg ul {
        margin:0px;
        padding:0px;
        list-style:none;
      }
      #theImg a {
        position:absolute;
        text-indent: -1000em;
      }
      #theImg a:hover {
        border: 1px solid #ffffff;
      }
      #ss a {
        top:0px;
        left:5px;
        width:80px;
        height:225px;
      }
      #gn a {
        top:226px;
        left:15px;
        width:70px;
        height:110px;
      }
      #ib a {
        top:225px;
        left:85px;
        width:60px;
        height:90px;
      }
    </style>
  </head>
  <body>
    <div id="theImage">
      <img alt="A tea shipment box with a map showing the location of the tea plantation." data-bbox="100 100 400 400"/>
      <ul>
        <li><a href="#">Tea Shipment</a></li>
        <li><a href="#">Tea Plantation</a></li>
        <li><a href="#">Tea Processing</a></li>
        <li><a href="#">Tea Packaging</a></li>
        <li><a href="#">Tea Distribution</a></li>
      </ul>
    </div>
  </body>
</html>
```



```

#iTEA1 a {
    top:100px;
    left:320px;
    width:178px;
    height:125px;
}
#iTEA2 a {
    top:225px;
    left:375px;
    width:123px;
    height:115px;
}
</style>
</head>
<body>
    <div id="theImg">
        <ul>
            <li id="ss"><a href="[valamilyen URL]"
                title="Sugarshots">Sugarshots</a></li>
            <li id="gn"><a href="[valamilyen URL]"
                title="Golden Needle">Golden Needle</a></li>
            <li id="ib"><a href="[valamilyen URL]"
                title="Irish Breakfast">Irish Breakfast</a></li>
            <li id="iTEA1"><a href="[valamilyen URL]"
                title="IngenuiTEA">IngenuiTEA</a></li>
            <li id="iTEA2"><a href="[valamilyen URL]"
                title="IngenuiTEA">IngenuiTEA</a></li>
        </ul>
    </div>
</body>
</html>

```



16.6. ábra

A CSS segítségével forrópontokat adhatunk meg a képtérképen

Ahogy a 16.2. példából kiderül, a stíluslapnak ugyan jó néhány bejegyzése van, de maga a HTML nyelvű rész meglehetősen rövid. Listaelemek használatával alakítjuk ki azt az öt területet, amelyre kattintani lehet – ezek a „területek” tehát valójában egy a háttérben megbúvó kép előtt lévő, adott magasságú és szélességű listaelemek. Ha a listát körbevevő `<div>` elem háttéréül szolgáló képet eltávolítanánk, a listaelemek továbbra is léteznének, és továbbra is kattinthatnánk rájuk.

Ha meg szeretnénk érteni azoknak az XHTML- és CSS-kódrészleteknek a szerepét, amelyek kialakítják a – végső soron egy hivatkozásokat tartalmazó listát rejtő – képtérképet, nézzük végig a stíluslapot.

A hivatkozásokat tartalmazó lista a `theImg` nevű `<div>` elem belsejébe kerül. A stíluslap ezt a `<div>` elemet 500 képpont szélességűvé, 375 képpont magasságúvá alakítja, és 1 képpontos folytonos kerettel veszi körül. Az elem háttére a nem ismétlődő, a `tea_shipment.jpg` fájlban tárolt kép. A következő HTML nyelvű rész a rendezetlen listát kezdő `` címke. A stíluslapunk a rendezetlen listának minden oldalon 0 képpont méretű margót és belső margót ad, a `list-style` tulajdonság értéke pedig `none`, azaz a listaelemeket nem jelzi felsorolásjel.

A listaelem szövege sohasem fog a felhasználó szeme elé kerülni, mégpedig a stíluslap alábbi, ügyes sora miatt, amely egyébiránt a `<div>` elem belsejében lévő valamennyi `<a>` címkére érvényes:

```
text-indent: -1000em;
```

Az 1000 em mértékű, de *negatív* behúzás biztosítja azt, hogy a szöveg sosem válik láthatóvá. Létezik ugyan, de a böngészőablak bal oldalától 1000 em távolságra balra, egy meg nem tekinthető helyen. Más szóval, ha a bal kezünket a számítógép monitorjának szélére helyezzük, a `text-indent: -1000em;` sor hatására a szöveg valahová az ujjainktól balra fog kerülni. Persze mi éppen ezt szeretnénk, ugyanis nem kívánjuk látni a hivatkozás szövegét. Egyszerűen csak olyan területekre van szükségünk, amelyek hivatkozásként működnek, és ha a felhasználó fölérjük mozgatja az egérmutatót, akkor a mutató éppúgy megváltozik, mint ahogy a weboldalakon elhelyezett hivatkozások fölé érve egyébként szokott.

Amikor a felhasználó az egérmutatóval egy hivatkozást tartalmazó listaelem fölé ér, a listaelem körül 1 képpont vastag folytonos keret válik láthatóvá, mégpedig a stíluslap alábbi bejegyzésének köszönhetően:

```
#theImg a:hover {
    border: 1px solid #ffffff;
}
```

Ezt követően a listaelemek megadása és elhelyezése következik, mégpedig a kép azon területeire, amelyeket kattinthatóvá kívánunk alakítani.

Példának okáért a kép Sugarshots nevű részéhez tartozó, ss azonosítóval rendelkező listaelemnek a bal felső sarka a <div> elem tetejétől 0, a bal szélétől pedig 5 képpont távolságra kerül. Az elem 80 képpont széles és 225 képpont magas. A #gn, az #ib, az #iTEA1 és az #iTEA2 azonosítójú stíluselemek formázása is hasonlóképpen történik, persze úgy, hogy a megfelelő azonosítójú elemek a kép kívánt részei elé kerüljenek.

Összefoglalás

Az órát néhány példával kezdtük, amelyek a margók és belső margók listaelemekre gyakorolt hatását mutatták be. Először a listákhoz tartozó alapértelmezett belső margóról, illetve ennek megváltoztatásáról szoltunk. Ezt követően megismerkedtünk a margó és a belső margó megváltoztatásának módjával, illetve megtanultuk, hogy a felsorolási jel miként helyezhető a lista belsejébe vagy azon kívülre. Így már lehet némi fogalmunk arról, hogy a stílusok és a listák a webhelyünk kinézetének egészére milyen hatással lehetnek. Végül megismerkedtünk azzal a módszerrel, amelynek a segítségével úgy készíthetünk képtérképet, hogy kizárólag az XHTML nyelv és a CSS elemeit vesszük igénybe, így nincs szükség a <map/> címke használatára, illetve a hivatkozásokat tartalmazó kép felszeletelésére sem.

Az óra valamennyi példájával azt kívántuk elérni, hogy az Olvasó merjen a szokásostól eltérő módon gondolni a listákra, és így olyan átfogó tudásra tegyen szert a rendezetlen listák használatáról, amellyel felvértezve vízszintes és függőleges navigációs menüvel gazdagíthatja a weboldalait.

Kérdezz-felelek

- K: Borzasztóan sok weboldalon olvashatunk – különösen a listákat és listaelemeket körülvevő margóval és belső margóval kapcsolatban – egy a dobozmodellel kapcsolatos trükkéről. Biztos, hogy erre nincs szükség?
- V: Az óra elején azt mondtuk, hogy a mostani – és a többi – lecke során megismert HTML-, illetve CSS-kódok azonos eredményt adnak a fontosabb webböngészők mai változataiban. Mindez számos olyan év után alakult így, amelyek alatt a webfejlesztőknek mindenféle trükköket kellett alkalmazniuk. A modern webböngészők – túllépve a saját örölteiken – kezdik az elemeket lassanként a CSS-szabványnak megfelelően kezelni. Manapság egyre többen vannak, akik arra törekszenek, hogy az internetezőket megszabadítsák azoktól a *nagyon* régi böngészőktől, amelyek miatt egyáltalán szükség volt az említett trükkökre. A könyv írói, bár mindezek szellemében nem javasolják feltétlenül azt, hogy az Olvasó *csak* a böngészők mai változatainak megfelelő weboldalakat készítsen, azt sem ajánlják, hogy rengeteg időt szánjon olyan trükkökre, amelyeket a böngészők régebbi – mostanra az internetezők alig öt százaléka által használt – változatai tesznek szükségessé. Érdemes tehát érvényesíthető, szilárd alapokra épülő és

a tervezési alapelveket követő kódot írni, az oldalakat minél több, az oldal olvasói között elterjedt böngészőben kipróbálni, és csak ezt követően elérhetővé tenni azokat az egész világ számára.

- K: *A CSS-képtérképek készítése igen munkaigényesnek tűnik. Tényleg annyira rossz a <map/> címke?*
- V: A <map/> címke egyáltalán nem rossz, sőt, mind az XHTML, mind a HTML 5 nyelvváltozatban érvényes. Ugyanakkor az ügyféloldali képtérképek koordinátáinak megállapítása nehézségekbe ütközhet, különösen, ha nincs grafikai alkalmazásunk, illetve az ügyféloldali képtérképek készítését segítő alkalmazásunk. A CSS-változat a kattintásérzékeny területek megadását és megjelenítését tekintve több lehetőséget kínál, bár ezek közül csak egyet mutattunk be.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Mi a különbség a list-style-position tulajdonság inside és outside értékei között? Melyik az alapértelmezett?
2. Létrejön-e – akár rendezett, akár rendezetlen – lista, ha a list-style tulajdonság értékeként none szerepel?
3. Írjunk HTML-kódot, amely egy 350 képpont széles, 100 képpont magas, zöld háttérű és 2 képpontos, szaggatott fekete keretű listaelemet hoz létre, olyat, ahol a felsorolási jelet a tárolóba kerül!

Válaszok

1. A list-style-position tulajdonság inside beállítása mellett a felsorolási jelet a listaelem által meghatározott terület belsejébe kerül. Az outside érték hatására a felsorolási jelet az említett területen kívül kap helyet. Az inside érték hatására a szöveg körbeveszi a felsorolási jelet. Az alapértelmezett érték az outside.
2. Igen. Az egyetlen különbség, hogy a listaelem tartalma előtt nem jelenik meg felsorolási jelet.
3. Használjuk az alábbi kódot:

```
<ul>
<li style="width:350px; height:100px; background-color:#00ff00;
border:2px dashed #000000; list-style-position:inside;">text goes
here</li>
</ul>
```

Gyakorlatok

- Keressünk egy képet, és próbáljunk rajta forrópontokat létrehozni egyedül, a ma látott módszer használatával. Olyan képet válasszunk, amelyen a megadott forrópontok, illetve kattintásérzékeny területek más oldalakhoz vezethetnek – akár a webhelyünkön belül, akár azon kívül. Írjuk meg a kattintásérzékeny területeket megadó és az általuk hivatkozott URL-eket tartalmazó HTML-, illetve CSS-kódot.
- Előkészülve a következő órára – amikor a listák használatával fogunk navigációs menüt kialakítani – idézzük fel a weboldalunk felépítését, és vázoljuk fel a benne lévő felső szintű kapcsolatokat, illetve ezeken belül adjunk meg néhány másodlagos hivatkozást is. Gondoljuk végig, hogy az állandóan látható navigációs sávunk vízszintes vagy függőleges irányú lesz-e.



17. ÓRA

Navigációs felület kialakítása a CSS segítségével

A lecke tartalma:

- Miben különböznek a navigációs listák a hagyományos társaiktól?
- Függőleges navigációs sáv készítése CSS-kód segítségével
- Vízszintes navigációs sáv készítése CSS-kód segítségével

Az előző órán megtanultuk, hogyan befolyásolhatjuk a listák megjelenését, így túlléphetünk az egyszerű felsorolásokon, illetve számozott listákon. A következőkben arról ejtünk pár szót, hogy miként használhatjuk a listákat vízszintes vagy függőleges navigációs sávként, valamint lenyíló menük szerepében.

Az alábbiakban ismertetett módszerek mindössze egy kis részét mutatják be azoknak a navigációs lehetőségeknek, amelyeket a listák segítségével valósíthatunk meg.

Az alapelvek azonban hasonlóak – a hihetetlen változatosság a tervezők fantáziájának terméke. Az óra végén felsorolt példák, amelyek nagyszerűen mutatják a CSS alapú navigáció előnyeit, a mi kreativitásunk felébresztésére szolgálnak.

Miben különböznek a navigációs listák a hagyományos társaiktól?

Amikor navigációs elemek szerepében alkalmazott listákról beszélünk, valójában CSS-stílusok használatára gondolunk, amelyekkel elérjük, hogy a listák úgy jelenjenek meg egy weboldalon, ahogy a látogatók azt a navigációs elemektől elvárják – vagyis teljesen más alakban, mint a hagyományos felsorolások és számozott listák. Egyrésztől persze igaz az is, hogy a navigációs elemek halmaza általában nem más, mint hivatkozások listája, ezek a hivatkozások azonban olyan formában jelennek meg, hogy a felhasználó azonnal tudja, hogy itt avatkozhat be a weboldal működésébe:

- Az egérmutató változása jelzi, hogy az elemre kattinthatunk.
- Az elem környezetében található terület megváltozik, amint az egérmutatót fölé vesszük.
- Az elem tartalma képileg elválik a normál szövegtől.

A navigációs elemeket régebben javarészt képekre építették – peremet utánzó gombokkal és a háttértől jól elváló szöveggel, és ehhez jött az ügyféloldali JavaScript, amely az egérműveleteknek megfelelően cserélgette a kép változatait. Tiszta CSS használatával azonban rugalmasabb, továbbá sokkal inkább felhasználó- és keresőbarát navigációs listákat készíthetünk, amelyek könnyen elérhetők a felhasználók számára, függetlenül attól, hogy milyen eszközzel jelenítik meg a weboldalt.

Akármilyen formában is tüntetjük fel a navigációs elemeket – vízszintes vagy függőleges elrendezésben –, ezen az órán két navigációs szintről ejtünk szót. Az *elsődleges navigációs elemek* a bevezető oldalakra, valamint a webhely fontosabb részeihez viszik a látogatót, míg a *másodlagos navigációs elemek* e fő részekben belül segítenek eligazodni.



Önálló feladat

A teljes webhelyre kiterjedő navigációs rendszer kialakítása

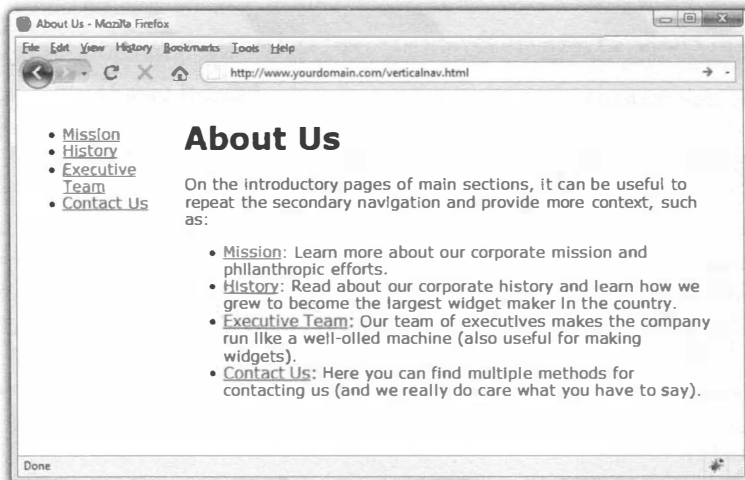
Az előző óra végén megkértük az Olvasót, hogy gondolkodjon el azon, hogy milyen navigációs szerkezetet használna a saját webhelyén. Ha ez a szerkezet – a különböző részekkel és ezek kisebb összetevőivel – felsejlik a lelki szemeink előtt, már van, amiből kiindulhatunk. Erre az alapra építve ezen az órán elkészíthetjük mind a vízszintes, mind a függőleges navigációs sáv elemeit.


```

</style>
</head>

<body>
  <div id="nav">
    <ul>
      <li><a href="#">Mission</a></li>
      <li><a href="#">History</a></li>
      <li><a href="#">Executive Team</a></li>
      <li><a href="#">Contact Us</a></li>
    </ul>
  </div>
  <div id="content">
    <h1>About Us</h1>
    <p>On the introductory pages of main sections, it can be useful
    to repeat the secondary navigation and provide more context,
    such as:</p>
    <ul>
      <li><a href="#">Mission</a>: Learn more about our corporate
      mission and philanthropic efforts.</li>
      <li><a href="#">History</a>: Read about our corporate history
      and learn how we grew to become the largest widget maker
      in the country.</li>
      <li><a href="#">Executive Team</a>: Our team of executives makes
      the company run like a well-oiled machine (also useful for
      making widgets).</li>
      <li><a href="#">Contact Us</a>: Here you can find multiple
      methods for contacting us (and we really do care what you
      have to say).</li>
    </ul>
  </div>
</body>
</html>

```



17.1. ábra

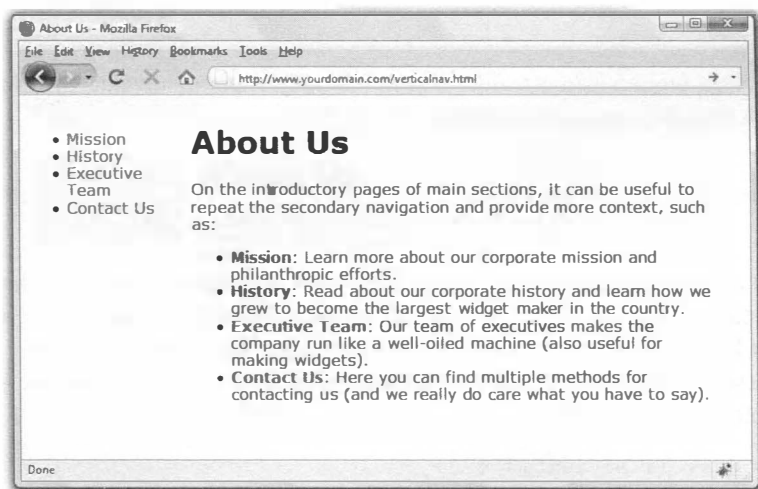
*A kiindulópont:
navigáció stílusok
nélküli listával*

Az oldal tartalma két egymás mellett elhelyezett `<div>` elemből áll össze: az egyikük `id` jellemzőjének értéke `nav`, a másiké pedig `content`. Jelenleg csak a `<div>` elemek szélességét, margóit, valamint a `float` értékét határoztuk meg stílusokkal; a listaelemek egyáltalán nem kaptak stílusokat.

A tartalom és a navigációs elemek listáinak megkülönböztetésére helyezzük el az alábbi kódot a stíluslapon:

```
#nav a {
    text-decoration: none;
}
#content a {
    text-decoration: none;
    font-weight: bold;
}
```

Ezzel mindössze annyit mondtunk, hogy amennyiben a `<div>` elem `id` jellemzőjének értéke `nav`, a benne található `<a>` hivatkozások ne kapjanak aláhúzást, ha pedig az `id` jellemző értéke `content`, az aláhúzás hiánya mellett az `<a>` hivatkozások félkövérrel jelenjenek meg. A különbségeket a 17.2. ábrán láthatjuk.



17.2. ábra

A listaelemek megkülönböztetése CSS-stílusokkal

Ha azonban az oldalsó navigációs listát valóban különlegessé szeretnénk tenni, ennél jóval részletesebben kell foglalkoznunk a stílusokkal.

Egyszintű függőleges navigációs sáv formázása

A fenti navigációs elemek esetében mindössze annyi a célunk, hogy eltüntessük a címkéket a felsorolásból, háttérrel adjunk nekik, és elérjük, hogy a szöveg színe a hivatkozás állapotától függően (egyszerű hivatkozás; már látott hivatkozás; hivatkozás, amely felett egérmutató áll; illetve hivatkozás, amelyre éppen rákattintanak) változzon. Az első lépéssel már el is készültünk: elválasztottuk a navigációt a tartalomtól. Ez abban a pillanatban megtörtént, amikor a navigációs elemeket egy `<div>` elembe helyeztük, amelynek az `id` jellemzője `nav`.

A következő lépésben módosítjuk az `` elem jellemzőit, amely a `nav <div>` elemen belül meghatározza az egyes hivatkozások megjelenését. Tüntessük el a listát jelző címkéket, és biztosítsuk, hogy a felső margón kívül ne legyen más margó vagy kitöltés. A felső margóra ugyanakkor szükség van ahhoz, hogy a navigációs lista tetejét hozzáigazítsuk a tartalomban található „About Us” címfelirat tetejéhez:

```
#nav ul {
    list-style: none;
    margin: 12px 0px 0px 0px;;
    padding: 0px;
}
```

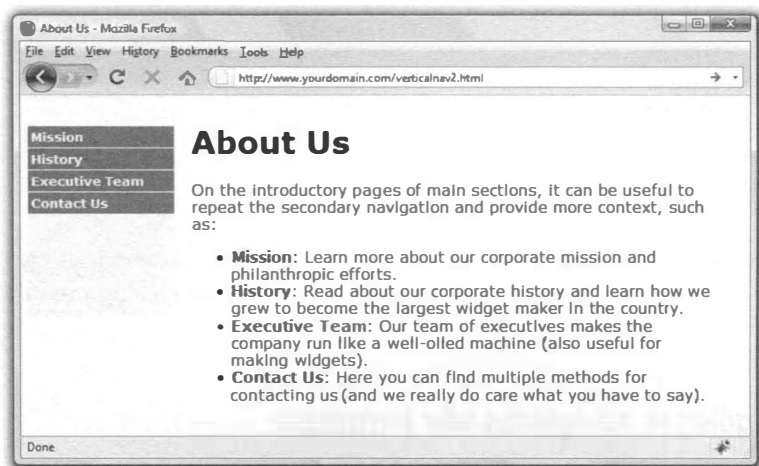
Mivel a navigációs lista elemei színes területekként jelennek meg, az elválasztásukhoz érdemes mindegyiknek alsó szegélyt adnunk:

```
#nav li {
    border-bottom: 1px solid #ffffff;
}
```

Következzen most a listaelemek stílusának meghatározása. Az elgondolásunk abban áll, hogy ha a listaelemek egyszerű hivatkozásként ülnek a helyükön, legyen háttérük egy meghatározott kék árnyalat, a szöveg pedig jelenjen meg fehérén és félkövér betűkkel (hozzátesszük, hogy a betűk mérete némiképp kisebb, mint a törzsszövegé). Mindehhez a következőt írhatjuk:

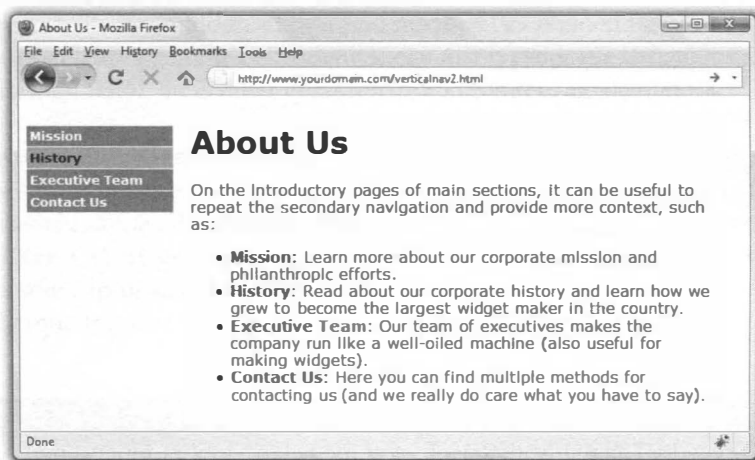
```
#nav li a:link, #nav li a:visited {
    font-size: 10pt;
    font-weight: bold;
    display: block;
    padding: 3px 0px 3px 3px;
    background-color: #628794;
    color: #ffffff;
}
```

A fentiekben minden stíluselem ismerős lehet, kivéve talán a `display: block`; kódot. Ezzel azt érjük el, hogy a teljes `` elem „érzi”, ha a felhasználó fölé viszi az egérmutatót. A függőleges navigációs listánkat a fenti stílusokkal a 17.3. ábrán láthatjuk.



17.3. ábra

A függőleges lista kezd navigációs menüre emlékeztetni



17.4. ábra

A listaelemek most már színt váltanak, ha fölérjük visszük az egérmutatót

Ha a felhasználó valamelyik navigációs elem felett tartja az egérmutatót, azt szeretnénk, hogy valamilyen látható változás menjen végbe, ami jelzi, hogy az elem reagál a kattintásra. Ez hasonló hatás ahhoz, amit a programokban is láthatunk: a menü színe megváltozik, amikor az egérmutató fölé ér. Esetünkben egyaránt megváltoztatjuk a háttér és a szöveg színét – vagyis elrugaszkodunk a megszokott fehér-kék együttestől:

```
#nav li a:hover, #nav li a:active {
    font-size: 10pt;
    font-weight: bold;
    display: block;
    padding: 3px 0px 3px 3px;
    background-color: #6cac46;
    color: #000000;
}
```

Az eddigi stílusmódosításaink eredményét a 17.4. ábra szemlélteti. Láthatjuk, hogy a stíluslap néhány bejegyzésével az egyszerű listából egy vizuálisan gazdag menühöz jutottunk.

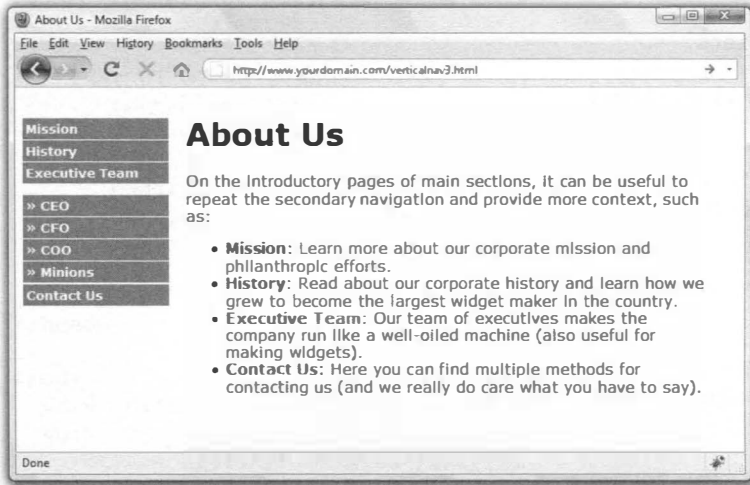
Többszintű függőleges navigációs sáv formázása

Mit tegyünk abban az esetben, ha szükség van egy második navigációs szintre is, amelyet a felhasználó bármikor elérhet? Nos, ezt megoldhatjuk beágyazott listákkal (amelyekről az előző órákon tanultunk) és további stíluslap-bejegyzésekkel. Tételezzük fel, hogy az Executive Team hivatkozás alatt további négy navigációs elemet kell feltüntetnünk. A lista HTML-kódját módosítsuk az alábbiak szerint:

```
<ul>
  <li><a href="#">Mission</a></li>
  <li><a href="#">History</a></li>
  <li><a href="#">Executive Team</a>
    <ul>
      <li><a href="#">&raquo; CEO</a>
      <li><a href="#">&raquo; CFO</a>
      <li><a href="#">&raquo; COO</a>
      <li><a href="#">&raquo; Other Minions</a>
    </ul>
  </li>
  <li><a href="#">Contact Us</a></li>
</ul>
```

Így létrehozzuk a beágyazott listát az Executive Team hivatkozás alatt (lásd a 17.5. ábrát). Az » HTML-egyeddel az új hivatkozások előtt megjelenő jobbra mutató nyilakat hozzuk létre.

Az új elemek tömbökben jelennek meg a lista szerkezetében, de a felület még nem tükrözi igazán az adatok hierarchiáját. Ahhoz, hogy némi képi segítséget is adjunk az Executive Team hivatkozás alá tartozó elemek elválasztásához, ismét módosítsuk a stíluslapot némi behúzás alkalmazásával.



17.5. ábra

*Beágyazott
navigációs lista
(amelynek
a formázása még
főmötésre szorul)*

Mielőtt azonban erre sor kerülne, meg kell változtatnunk néhány egyéb stíluslap-bejegyzést is. Az előzőekben használatba vettünk néhány elemválasztót (kijelölőt) – ezek a `#nav ul` és `#nav li`, amelyeknek a jelentése „minden `` a `nav` nevű `<div>` elemen belül”, illetve „minden `` a `nav` nevű `<div>` elemen belül”. Most azonban már két `` elem, valamint `` elemek egy újabb csoportja látható a `nav` nevezetű `<div>` elemen belül, amelyeket el szeretnénk különíteni az előzőektől.

Ahhoz, hogy biztosítsuk mindkét elemtípus megfelelő formázását, el kell érünk, hogy a stíluslapok elemválasztói pontosan tükrözzék a listahierarchia viszonyait. Ennek megvalósításához a listák első szintjén a `#nav ul` és `#nav ul li`, a második szinten pedig a `#nav ul ul` és a `#nav ul ul li` elemválasztókat kell használnunk. A 17.2. példában a stíluslap új változatát láthatjuk, valamint azt a HTML-kódot, amely a 17.6. ábrán bemutatott menüt adja.

17.2. példa Többszintű függőleges navigációs lista

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>About Us</title>
    <style type="text/css">
      body {
        font: 12pt Verdana, Arial, Georgia, sans-serif;
      }
    </style>
  </head>
  <body>
    <div id="nav">
      <ul>
        <li>Mission</li>
        <li>History</li>
        <li>Executive Team</li>
        <li>CEO</li>
        <li>CFO</li>
        <li>COO</li>
        <li>Minions</li>
        <li>Contact Us</li>
      </ul>
    </div>
  </body>
</html>
```

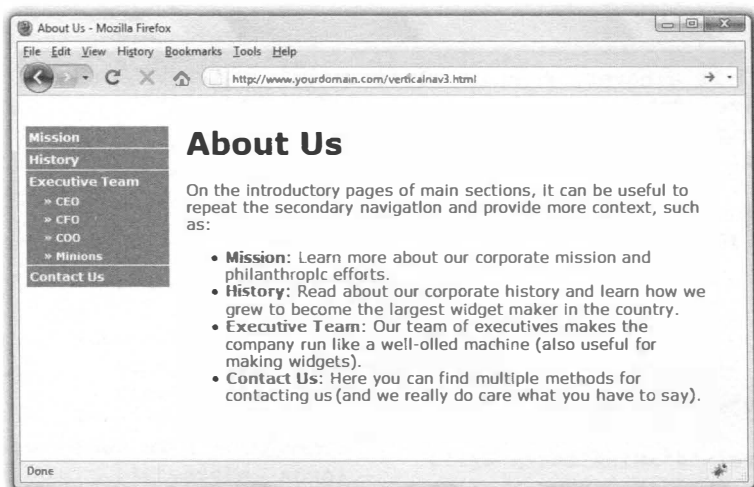
```
#nav {
    width:150px;
    float:left;
    margin-top:12px;
    margin-right:18px;
}
#content {
    width:550px;
    float:left;
}
#nav a {
    text-decoration: none;
}
#content a {
    text-decoration: none;
    font-weight: bold;
}
#nav ul {
    list-style: none;
    margin: 12px 0px 0px 0px;
    padding: 0px;
}
#nav ul li {
    border-bottom: 1px solid #ffffff;
}
#nav ul li a:link, #nav ul li a:visited {
    font-size: 10pt;
    font-weight: bold;
    display: block;
    padding: 3px 0px 3px 3px;
    background-color: #628794;
    color: #ffffff;
}
#nav ul li a:hover, #nav ul li a:active {
    font-size: 10pt;
    font-weight: bold;
    display: block;
    padding: 3px 0px 3px 3px;
    background-color: #c6a648;
    color: #000000;
}
#nav ul ul {
    margin: 0px;
    padding: 0px;
}
#nav ul ul li {
    border-bottom: none;
}
#nav ul ul li a:link, #nav ul ul li a:visited {
    font-size: 8pt;
    font-weight: bold;
    display: block;
    padding: 3px 0px 3px 18px;
```

```

        background-color: #628794;
        color: #ffffff;
    }
    #nav ul ul li a:hover, #nav ul ul li a:active {
        font-size: 8pt;
        font-weight: bold;
        display: block;
        padding: 3px 0px 3px 18px;
        background-color: #c6a648;
        color: #000000;
    }
</style>
</head>

<body>
  <div id="nav">
    <ul>
      <li><a href="#">Mission</a></li>
      <li><a href="#">History</a></li>
      <li><a href="#">Executive Team</a>
      <ul>
        <li><a href="#">&raquo; CEO</a>
        <li><a href="#">&raquo; CFO</a>
        <li><a href="#">&raquo; COO</a>
        <li><a href="#">&raquo; Other Minions</a>
      </ul>
      </li>
      <li><a href="#">Contact Us</a></li>
    </ul>
  </div>
  <div id="content">
    <h1>About Us</h1>
    <p>On the introductory pages of main sections, it can be useful
    to repeat the secondary navigation and provide more context,
    such as:</p>
    <ul>
      <li><a href="#">Mission</a>: Learn more about our corporate
      mission and philanthropic efforts.</li>
      <li><a href="#">History</a>: Read about our corporate history
      and learn how we grew to become the largest widget maker
      in the country.</li>
      <li><a href="#">Executive Team</a>: Our team of executives makes
      the company run like a well-oiled machine (also useful for
      making widgets).</li>
      <li><a href="#">Contact Us</a>: Here you can find multiple
      methods for contacting us (and we really do care what you
      have to say.</li>
    </ul>
  </div>
</body>
</html>

```

17.6. ábra

Több szintű függőleges navigációs lista létrehozása CSS-stílusokkal

A függőleges navigációs elemek stílusváltozatainak csak a fantáziánk szab határt – a lehetőségek kimeríthetetlenek. Nyugodt szívvel használhatunk színeket, margókat, kitöltést, háttérképeket és egyáltalán, a CSS összes lehetőségét – az így kapott eredmény pedig rugalmas és bármikor könnyen módosítható. Ha beírjuk a „CSS vertical navigation” (CSS függőleges navigáció) kifejezést egy internetes keresőbe, ezrenyi példát láthatunk a stíluslapok használatára – és ezek mindegyike az ezen az órán megismert egyszerű szabályokra épül.

Vízszintes navigációs sáv készítése CSS-kód segítségével

Az órát a függőleges navigáció témakörével kezdtük, mivel a lista és a navigációs elemek közötti átmenet így szemléletesebb volt – végeredményben a navigációs elemek továbbra is egy listát alkotnak, amely szemre még mindig hasonlít valamelyest a jó öreg bevásárlólistára. A vízszintes navigáció megvalósításánál továbbra is listát alkalmazunk, de ez esetben az `` és a `` elemek `display` jellemzőjénél nem a korábban megismert `block`, hanem az `inline` értéket használjuk. Ennyi az egész!

A 17.3. példában a kiindulópontként szolgáló HTML-oldal kódját láthatjuk, amelyben vízszintes navigációt szeretnénk alkalmazni. Az oldalon két `<div>` elemet találhatunk: az egyik a fejléctet, a másik pedig a tartalmat tárolja. A fejlécen belül további két, egymás mellett úszó `<div>` elemet találunk – az egyikben az embléma kap helyet, a másikban pedig a navigációs sáv. Az utóbbiban egy listát találunk, amelyben mind a lista, mind az elemek a `display` jellemző `inline` értékét hordozzák. Az elemeket és elhelyezésüket a 17.7. ábrán láthatjuk.

17.3. példa *Listára épülő egyszerű vízszintes navigáció*

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>ACME Widgets LLC</title>
    <style type="text/css">
      body {
        font: 12pt Verdana, Arial, Georgia, sans-serif;
      }
      #header {
        width: auto;
      }
      #logo {
        float:left;
      }
      #nav {
        float:left;
      }
      #nav ul {
        list-style: none;
        display: inline;
      }
      #nav li {
        display: inline;
      }
      #content {
        width: auto;
        float: left;
        clear: left;
      }
      #content a {
        text-decoration: none;
        font-weight: bold;
      }
    </style>
  </head>
  <body>
    <div id="header">
      <div id="logo">
        
      </div>
      <div id="nav">
        <ul>
          <li><a href="#">About Us</a></li>
          <li><a href="#">Products</a></li>
          <li><a href="#">Support</a></li>
          <li><a href="#">Press</a></li>
        </ul>
      </div>
    </div>
  </body>
</html>

```

```

    </div>
</div>
<div id="content">
  <p><strong>ACME Widgets LLC</strong> is the greatest widget-maker
  in all the land.</p>
  <p>Don't believe us? Read on...</p>
  <ul>
    <li><a href="#">About Us</a>: We are pretty great.</li>
    <li><a href="#">Products</a>: Our products are the best.</li>
    <li><a href="#">Support</a>: It is unlikely you will need support,
    but we provide it anyway.</li>
    <li><a href="#">Press</a>: Read what others are saying (about how
    great we are).</li>
  </ul>
</div>
</body>
</html>

```



17.7. ábra

Működő – bár nem igazán tetszetős – vízszintes navigációs sáv, amely egy sorban elhelyezkedő lista-elemekből áll

A lista megjelenésének testreszabására e ponttól csak CSS-stílusokat használunk, a HTML-tartalom már elkészült. A kívánt megjelenés eléréséhez alkalmazzuk a következő CSS-kódot. Először is, módosítsuk a nav nevű <div> elem szélességét, határozzunk meg a számára háttérszínt és szegélyt, valamint 85 pontos felső margót (így az embléma aljához közel jelenik meg):

```

#nav {
  float:left;
  margin: 85px 0px 0px 0px;
  width: 400px;
  background-color: #628794;
  border: 1px solid black;
}

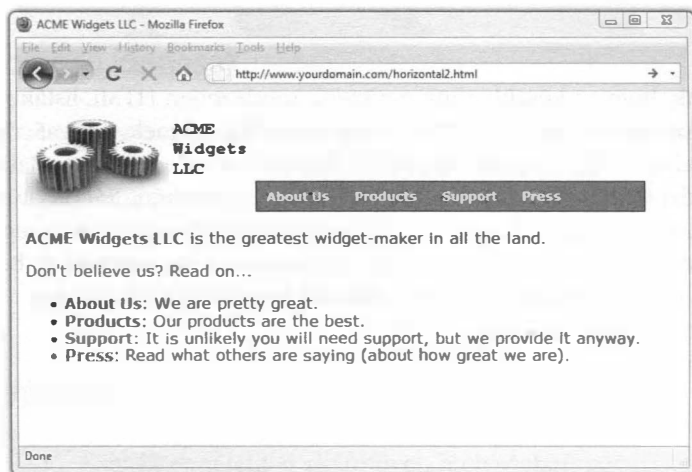
```

Az `` elem meghatározása a margóktól és a kitöltéstől eltekintve megegyezik a 17.3. példában látottakkal:

```
#nav ul {
    margin: 0px;
    padding: 0px;
    list-style: none;
    display: inline;
}
```

A `` elem esetében sincs nagy változás, mindössze a `line-height` értékét állítjuk 1.8em-re:

```
#nav li {
    display: inline;
    line-height: 1.8em;
}
```



17.8. ábra

Vízszintes navigációs sáv kialakítása stílusokkal

A hivatkozások stílusai hasonlóak a függőleges navigációnál látottakhoz – a kitöltés értékei eltérnek, de a színek és a betűméret azonos:

```
#nav ul li a:link, #nav ul li a:visited {
    font-size: 10pt;
    font-weight: bold;
    text-decoration: none;
    padding: 7px 10px 7px 10px;
    background-color: #628794;
    color: #ffffff;
}
```

```
#nav ul li a:hover, #nav ul li a:active {
    font-size: 10pt;
    font-weight: bold;
    text-decoration: none;
    padding: px 10px 7px 10px;
    background-color: #c6a648;
    color: #000000;
}
```

E módosításokat követően az oldal a 17.8. ábrán látható alakot ölti.

Ha a látogató a navigációs elemek fölé viszi az egérmutatót, a háttér és a szöveg színe ugyanúgy megváltozik, mint a függőleges navigációs menü esetében láttuk. Emellett itt is élhetünk a beágyazott listák használatának lehetőségével, így lenyíló menüket kaphatunk. Próbáljuk ki!

Összefoglalás

Ezen az órán megtanultuk, hogyan készíthetünk egyszerű, rendezetlen HTML-listákból függőleges és vízszintes navigációs sávokat. Azzal, hogy a grafikai elemek, a JavaScript, illetve más módszerek helyett a CSS használata mellett döntöttünk, jelentős rugalmasságot nyertünk mind az oldal fenntartása, mind pedig a megjelenítése terén. Ráébredtünk, hogy az egyszerű, aláhúzott szöveges hivatkozásokból néhány CSS-bejegyzéssel szegélyekkel körülvett, háttérszínnel rendelkező területeket hozhatunk létre, amelyeken belül a szöveg megjelenését is testreszabhatjuk. Minde mellett azt is megtanultuk, miként helyezzünk el beágyazott listákat a menükön belül.

Kérdezz-felelek

- K: *Használhatunk a navigációs menükben képeket a listák címkéiként?*
- V: Igen. A képet elhelyezhetjük a listaelem HTML-szövegében, de a elem háttérképeként is alkalmazhatjuk. A navigációs elemek stílusait pontosan úgy határozhatjuk meg, mint bármely más listaelemét. A rendezetlen HTML-listák és a CSS alapú függőleges vagy vízszintes navigációs sávok között csak a névben van különbség, no meg abban, hogy az utóbbiak esetében a listát a törzsszöveg kívül, különleges célra használjuk. Ennek tudatában érdemes ezeknek a listáknak valóban különleges formázást biztosítanunk – ebbe pedig pár grafikai elem alkalmazása bőven befér.
- K: *Hol találhatunk további példákat a listák alkalmazási lehetőségeire?*
- V: A „CSS navigation” keresőkifejezésre legutóbb mintegy 44 millió találatot kaptam, így hát példa van elég. Íme néhány kiindulópont:

- Az A *List Apart* CSS-stílusokkal kapcsolatos cikkei:

<http://www.alistapart.com/topics/code/>

- CSS Listamatic a Maxdesign-től:

<http://css.maxdesign.com.au/listamatic/>

- Vitaly Friedman *CSS Showcase* nevű oldala:

<http://www.alvit.de/css-showcase/>

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. A listákra épülő navigációs sávokban hány szinten ágyazhatjuk egymásba a listákat?
2. „Ha a `display` jellemzőnek az `inline` értéket adjuk, vízszintes listát kapunk.” Igaz vagy hamis?
3. Navigációs lista készítésénél megtehetjük, hogy az a elemválasztó mind a négy álosztályának ugyanazokat az értékeket adjuk?

Válaszok

1. A listák elméletileg akármilyen mélységben egymásba ágyazhatók, a használhatóság szempontjai miatt azonban van egy ésszerű határ erre nézve. Többnyire három szintig szokás elmenni – egyébként kusza szerkezetű webhelyhez jutunk, vagy a kelleténél több választási lehetőség elé állítjuk a látogatókat.
2. Igaz, amennyiben a stíluslapon mind az `ul`, mind a `li` elemválasztók esetében beállítjuk a `display` jellemzőt.
3. Természetesen megtehetjük, de ezzel azt kockáztatjuk, hogy a gyönyörű menüinkről senki nem veszi észre, hogy itt valóban menükről van szó (hiszen az egérműveletek semmilyen képi visszajelzést nem adnak).

Gyakorlatok

- A többszintű függőleges listáknál bemutatott módszerek alkalmazásával hozzunk létre alárendelt navigációs elemeket a fejezet végén készített függőleges listához.
- Tanulmányozzuk különféle webhelyek CSS alapú navigációs megoldásait, és keressünk új fogásokat. A böngésző „Oldal forrása” parancsával vizsgáljuk meg a kódot, és próbáljuk megvalósítani a látottakat a saját weboldalainkon.



18. ÓRA

Szövegmódosítás egérműveletekkel

A lecke tartalma:

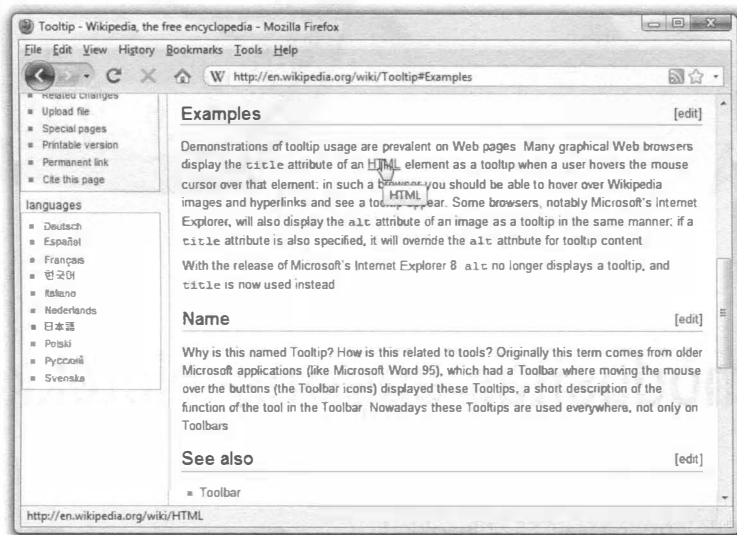
- Lebegő leírások létrehozása CSS-stílusokkal
- További váltószöveg megjelenítése CSS-stílusok segítségével
- Az események elérése
- A `<div>` elemek megjelenésének megváltoztatása az `onclick` esemény segítségével

Könyvünk eddigi részében láthattunk már példákat arra, hogy az egérműveletek miként változtathatják meg a szövegek megjelenítését. A legegyszerűbb példa talán az, amikor az `<a>` hivatkozás `hover` álosztályának stílusait úgy határozzuk meg, hogy a szöveg eltérő színnel jelenjen meg, amikor az egérmutató a hivatkozás fölé ér. Az előző órán megtanultuk, hogyan változtathatja meg a `hover` álosztály egy `` elem hátterének és szövegének színét.

Ezen az órán két módszert is tanulunk az egérműveletek és a CSS-stílusok együttes használatára: szöveget jelenítünk meg, amikor az egérmutató egy felületi elem fölé kerül, valamint egérekattintás hatására megváltoztatjuk egy tárolóelem színét. Ezek a lehetőségek önmagukban is hasznosak lehetnek, de ami még fontosabb, hogy jó kiindulópontot adnak azokhoz az összetett műveletekhez, amelyekre a webfejlesztés során később szükségünk lehet.

Lebegő leírások létrehozása CSS segítségével

A *lebegő leírás* (tooltip) a grafikus felületnek – legyen programé vagy webhelyé – egy olyan eleme, amely bővebb tájékoztatást ad, ha az egérmutatót egy adott elem fölé visszük. A 18.1. ábrán egy ilyen lebegő leírást láthatunk működés közben: az egérmutató a „HTML” hivatkozás felett áll, a lebegő leírás pedig a „HTML” szöveget jeleníti meg egy kis szövegmezőben. Ez esetben a leírás szövegét az `<a>` elem `title` jellemzője adja.



18.1. ábra

Egyszerű lebegő leírás működés közben

A lebegő leírás megjelenítését a böngésző intézi, így ennek mikéntjébe webprogramozóként nem szólhatunk bele. Arra viszont lehetőséget kapunk, hogy némi CSS-kóddal és a korábban tanultak alkalmazásával létrehozzuk a saját lebegő leírásainkat.

A 18.1. példában egy olyan weboldal rajzolódik ki előttünk, stíluslap-bejegyzésekkel, valamint szövegek és képek HTML-kódjával –, amelyben az egyik hivatkozás egyéni lebegő leírást jelenít meg.

Vegyük észre, hogy a szövegben található első hivatkozáshoz tartozik egy osztály, amelynek a neve `tip` – ez a név különbözteti meg a hivatkozásunkat a társaitól, amelyek nem rendelkeznek lebegő leírással. Mivel azonban itt a `class`, nem pedig az `id` értékét határoztuk meg, a nevet használhatjuk az oldal többi hivatkozásánál is.

18.1. példa *Egyszerű lebegő leírás létrehozása CSS-stílusokkal*

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Steptoe Butte</title>
    <style type="text/css">
      a {
        text-decoration: none;
        font-weight: bold;
      }
      a.tip {
        position: relative;
        z-index: 24;
      }
      a.tip: hover {
        z-index: 25;
      }
      a.tip span {
        display: none;
      }
      a.tip: hover span {
        font-weight: normal;
        display: block;
        position: absolute;
        top: 20px;
        left: 25px;
        width: 150px;
        padding: 3px;
        border: 1px solid #000;
        background-color: #ddd;
        color: #000;
      }
    </style>
  </head>
  <body>
    <h1>Steptoe Butte</h1>
    <p>Steptoe Butte is a quartzite island jutting out of the
silty loess of the <a class="tip"
href="http://en.wikipedia.org/wiki/Palouse">Palouse <span>Learn more
about the Palouse!</span></a> hills in Whitman County, Washington. The
rock that forms the butte is over 400 million years old, in contrast
with the 15-7 million year old
<a href="http://en.wikipedia.org/wiki/Columbia_River">Columbia River</a>
basalts that underlie the rest of the Palouse (such "islands" of
ancient rock have come to be called buttes, a butte being defined as
a small hill with a flat top, whose width at top does not exceed its
height).</p>

```

```

<p>A hotel built by Cashup Davis stood atop Steptoe Butte from 1888 to
1908, burning down several years after it closed. In 1946, Virgil
McCroskey donated 120 acres (0.49 km2) of land to form Steptoe Butte
State Park, which was later increased to over 150 acres (0.61 km2).
Steptoe Butte is currently recognized as a National Natural Landmark
because of its unique geological value. It is named in honor of
<a href="http://en.wikipedia.org/wiki/Colonel_Edward_Steptoe">Colonel
Edward Steptoe</a>.</p>
<p>Elevation: 3,612 feet (1,101 m), approximately 1,000 feet (300 m)
above the surrounding countryside.</p>
<p><em>Text from
<a href="http://en.wikipedia.org/wiki/Steptoe_Butte">Wikipedia</a>,
photo by the author.</em></p>
</body>
</html>

```

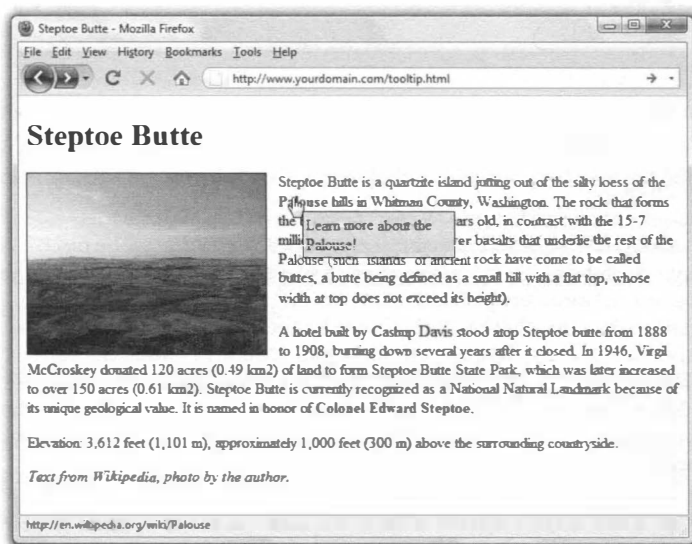
A tip osztály meghatározza a position és a z-index stílusokat. A position relative értéke biztosítja, hogy az elem az eredeti környezetében jelenjen meg a szövegben, a z-index értékének beállításával pedig azt érhetjük el, hogy a hivatkozás szövege a nagyobb z-index értékkel rendelkező elemek alatt maradjon. A tip osztályt alkalmazó hivatkozások esetében például a hover állapot magasabb z-index értéket takar.

A következő két stílus az alkalmazott elemekhez köthető. Az <a> elem belsejében található elemen belüli szöveg nem jelenik meg mindaddig, amíg meg nem jelenítjük a lebegő leírást az egérmutatót a hivatkozás fölé helyezve. Vegyük szemügyre az alábbi hivatkozást:

```

<a class="tip" href="http://en.wikipedia.org/wiki/Palouse">Palouse
<span>Learn more about the Palouse!</span></a>

```



18.2. ábra

Ha az egérmutatóval az oldal első hivatkozása fölé megyünk, egyéni lebegő leírást kapunk

Első pillantásra úgy tűnik, hogy ennek a HTML-kódnak a „Palouse Learn more about the Palouse!” szöveget kellene megjelenítenie. A „Learn more about the Palouse!” szövegrészlet azonban a `` elemen belül áll, amelynek a stílusát úgy határoztuk meg, hogy csak akkor jelenjen meg a képernyőn, ha az egérmutató a valódi hivatkozás („Palouse”) fölé ér. Az eredményt a 18.2. ábrán láthatjuk.

A „Learn more about the Palouse!” szöveg a `` elemen belül kap helyet a HTML-kódban, a stílusát azonban a stíluslap határozza meg. A hatásért egészen pontosan az a `.tip:hover span` kijelölő felel, amely egy 150 képpont széles téglalapot hoz létre, sötétzöld háttérrel és fekete szegéllyel. A téglalap szülőjétől lefelé 20, jobbra pedig 25 képponttal eltolva jelenik meg. Az eredmény nem más, mint egy lebegő leírás.

Egyéb szövegrészletek megjelenítése CSS-kód segítségével

A lebegő leírások szerepe viszonylag behatárolt: egy szöveges „tipp” megjelenítése egy adott hivatkozás mellett. A megjelenítésük módszerét azonban más területen is használhatjuk – a CSS-stílusok segítségével az egérműveletek függvényében rejtett szövegeket jeleníthetünk meg az oldal különböző részein. Vegyük példaként a 17. órán megismert ACME Widgets LLC kezdőoldalát, mégpedig azt, amelyiken a vízszintes navigációs sávot alkalmaztuk. A 17.1. példa módszereit használva megjeleníthetünk némi tájékoztatást a menü felett, amikor a felhasználó a nagyobb részek hivatkozásai fölé helyezi az egérmutatót. A 18.3. ábra ezek egyikét mutatja működés közben. A megjelenítés érdekében visszaszerezünk némi területet a navigációs sáv felett és az embléma mellett.



18.3. ábra

Egyéni szöveg egy másik felületi elem felett, amely csak az egérmutató hatására válik láthatóvá

A 18.1. példához képest mindössze annyit változtattunk a CSS-kódon, hogy beillesztettük az alábbi négy stílust. Ugyanazt teszük, mint a tip osztály stílusai, csak a megjelenítésben térnek el az ott látottaktól:

```
a.more {
    position: relative;
    z-index: 24;
}
a.more:hover {
    z-index: 25;
}
a.more span {
    display: none;
}
a.more:hover span {
    font-weight: bold;
    display: block;
    position: absolute;
    top: -35px;
    width: 400px;
    padding: 3px;
    color: #ff0000;
    line-height: 1em;
}
```

A top tulajdonság -25px értéke a beágyazott elem tartalmát 25 képponttal a szülőelem bal felső sarka fölé helyezi, ellentétben az előző példával, ahol 20 képponttal alá tettük. A HTML-kód módosításai hasonlóak az előzőekben látottakhoz: az egérmutató hatására megjelenő szöveget itt is az <a> hivatkozáson belüli elembe rejtettük el:

```
<ul>
  <li><a class="more" href="#">About Us <span>We are pretty
  great.</span></a></li>
  <li><a class="more" href="#">Products <span>Our products are
  the best.</span></a></li>
  <li><a class="more" href="#">Support <span>It is unlikely you
  will need support, but we provide it anyway.</span></a></li>
  <li><a class="more" href="#">Press <span>Read what others are
  saying (about how great we are).</span></a></li>
</ul>
```

Az itt megismert eljárások alkalmazásával tetszőleges szöveget megjeleníthetünk az oldal bármely pontján, amikor a felhasználó az egérmutatót egy <a> hivatkozás fölé viszi. A következőkben megismerkedünk az eseményekkel, és láthatjuk majd, hogy némi JavaScript-kód segítségével még komolyabb eredmény születhet az egérműveletek és a CSS együttműködéséből.

Események elérése

A felhasználó műveletei, mint a billentyűleütés vagy az egérkattintás, mind-mind *események*. Ha pedig egy parancskód egy eseményre válaszul valamilyen műveletet végez, *eseménykezelésről* beszélünk. Az eseménykezelő parancskódok és a weboldalunk elemeinek összekapcsolására különleges jellemzők állnak a rendelkezésünkre.

Az alábbiakban bemutatunk néhány fontosabb eseményjellemzőt, amelyeknek az ismerete hasznos lehet a JavaScript-parancskódok írása során. Emellett azt is eláruljuk, hogy az események bekövetkezése milyen kapcsolatban áll az oldal egyes elemeivel.

- `onload` – A böngésző betölti az elemet.
- `onkeydown` – A felhasználó lenyom egy billentyűt.
- `onkeyup` – A felhasználó felenged egy billentyűt.
- `onclick` – A felhasználó egy elemre kattint a bal egérgombbal.
- `ondblclick` – A felhasználó duplán kattint egy elemre a bal egérgombbal.
- `onmousedown` – A felhasználó lenyomja valamelyik egérgombot, miközben az egérmutató egy adott elem felett áll.
- `onmouseup` – A felhasználó felengedi valamelyik egérgombot, miközben az egérmutató egy adott elem felett áll.
- `onmouseover` – A felhasználó az elem határain belülre viszi az egérmutatót.
- `onmousemove` – A felhasználó az elem határain belül mozgatja az egérmutatót.
- `onmouseout` – A felhasználó az elem határain kívülre viszi az egérmutatót.

Láthatjuk, hogy az eseményjellemzők olyan gyakran előforduló felhasználói műveleteket jeleznek, mint az egérkattintás és a billentyűleütés. Az eseményekhez JavaScript-kódokat kapcsolhatunk; ehhez nem kell mást tennünk, mint hozzárendelni a kódot az eseményjellemzőhöz, valahogy így:

```
<h1 onclick="this.style.color = 'red';">I turn red when clicked.</h1>
```

A fenti példában a JavaScript-kódot egy `<h1>` elem `onclick` eseményjellemzőjéhez rendeltük, ami azt jelenti, hogy a kód akkor indul el, ha a felhasználó a bal egérgombbal a szövegre kattint. A kód ennek hatására a szöveg színét vörösre változtatja.

Az addig statikus szöveg így érzékennyé válik a felhasználói műveletekre, hiszen egy kattintásra megváltoztatja a színét. Ez az egyszerű példa jól mutatja, hogyan működhetnek együtt az ügyféloldali parancskódok a böngészővel.

A következőkben megmutatjuk, miként módosíthatjuk egy `<div>` elem megjelenését az eseménykezelés segítségével. A célunk egy olyan `<div>` elem létrehozása, amely egérekattintásra eltűnik, illetve megjelenik.

<div> elem megjelenésének módosítása az onclick esemény segítségével

Az onclick eseménnyel mindenféle műveletet kiválthatunk – tudjuk persze, hogy egy űrlap Submit gombjára kattintva elküldhetjük a felvett adatokat, de az egérek kattintást el is foghatjuk, és így sokkal gazdagabb szerepet adhatunk neki az oldalunk működésében. A következő példában láthatjuk, miként alkalmazhatjuk az onclick eseményt egy <div> elem tartalmának elrejtésére, illetve megjelenítésére. A 18.1. példa szöveges és képi elemekre építve bemutatjuk, hogyan jeleníthetünk meg korábban elrejtett adatokat, amennyiben a felhasználó egy szövegrészletre kattint. A „szövegrészlet” kifejezés nem pongyola szóhasználat, ugyanis szigorúan véve itt nem hivatkozásról van szó. Igaz, hogy annak néz ki, és úgy is viselkedik, de nem egy <a> elemen belül áll a kódban. A 18.2. példában az oldal teljes kódját láthatjuk, a 18.4. ábrán pedig megszemlélhetjük, hogyan jelenik meg betöltés után a böngészőablakban.

18.2. példa *Tartalom elrejtése és megjelenítése az onclick eseménnyel*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Steptoe Butte</title>
    <style type="text/css">
      a {
        text-decoration: none;
        font-weight: bold;
        color: #7a7abf;
      }
      #hide_e {
        display: none;
      }
      #elevation {
        display: none;
      }
      #hide_p {
        display: none;
      }
      #photos {
        display: none;
      }
      #show_e {
        display: block;
      }
      #show_p {
        display: block;
      }
    </style>
  </head>
  <body>
    <div id="steptoe">
      <div id="hide_e">
        <p>Steptoe Butte</p>
      </div>
      <div id="show_e">
        <p>Steptoe Butte</p>
      </div>
    </div>
    <div id="elevation">
      <div id="hide_p">
        <p>Elevation</p>
      </div>
      <div id="show_p">
        <p>Elevation</p>
      </div>
    </div>
    <div id="photos">
      <div id="hide_p">
        <p>Photos</p>
      </div>
      <div id="show_p">
        <p>Photos</p>
      </div>
    </div>
  </body>
</html>
```

```

.fakelink {
    cursor:pointer;
    text-decoration: none;
    font-weight: bold;
    color: #E03A3E;
}
</style>
</head>
<body>
<h1>Steptoe Butte</h1>
<p>Steptoe Butte is a quartzite island jutting out of the
silty loess of the <a class="tip"
href="http://en.wikipedia.org/wiki/Palouse">Palouse <span>Learn more
about the Palouse!</span></a> hills in Whitman County, Washington. The
rock that forms the butte is over 400 million years old, in contrast
with the 15-7 million year old
<a href="http://en.wikipedia.org/wiki/Columbia_River">Columbia
River</a>
basalts that underlie the rest of the Palouse (such "islands" of
ancient rock have come to be called buttes, a butte being defined as
a small hill with a flat top, whose width at top does not exceed its
height).</p>
<p>A hotel built by Cashup Davis stood atop Steptoe Butte from 1888 to
1908, burning down several years after it closed. In 1946, Virgil
McCroskey donated 120 acres (0.49 km2) of land to form Steptoe Butte
State Park, which was later increased to over 150 acres (0.61 km2).
Steptoe Butte is currently recognized as a National Natural Landmark
because of its unique geological value. It is named in honor of
<a href="http://en.wikipedia.org/wiki/Colonel_Edward_Steptoe">Colonel
Edward Steptoe</a>.</p>
<div class="fakelink"
    id="show_e"
    onclick="this.style.display='none';
    document.getElementById('hide_e').style.display='block';
    document.getElementById('elevation').style.display='inline';
">&raquo; Show Elevation</div>

<div class="fakelink"
    id="hide_e"
    onclick="this.style.display='none';
    document.getElementById('show_e').style.display='block';
    document.getElementById('elevation').style.display='none';
">&raquo; Hide Elevation</div>

<div id="elevation">3,612 feet (1,101 m), approximately 1,000 feet
(300 m) above the surrounding countryside.</div>

<div class="fakelink"
    id="show_p"
    onclick="this.style.display='none';
    document.getElementById('hide_p').style.display='block';
    document.getElementById('photos').style.display='inline';
">&raquo; Show Photos from the Top of Steptoe Butte</div>

```

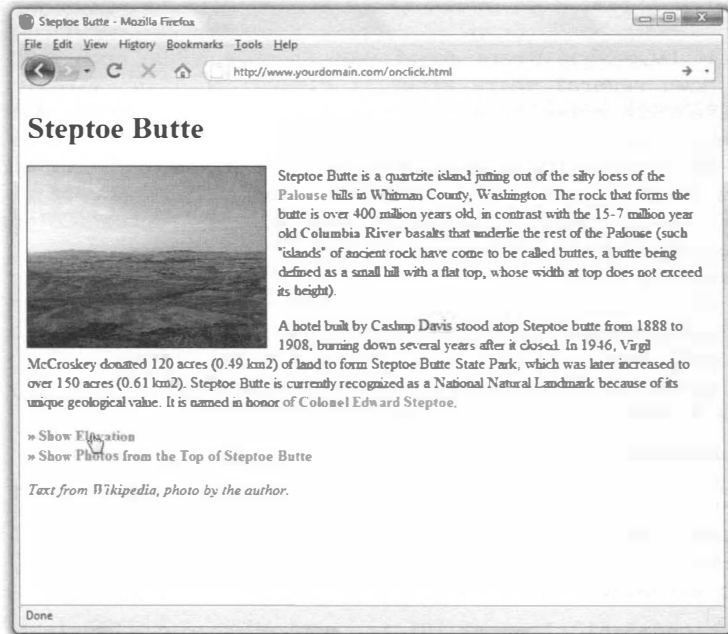


```

<div class="fakelink"
id="hide_p"
onclick="this.style.display='none';
document.getElementById('show_p').style.display='block';
document.getElementById('photos').style.display='none';
">&raquo; Hide Photos from the Top of Steptoe Butte</div>
<div id="photos"></div>

<p><em>Text from
<a href="http://en.wikipedia.org/wiki/Steptoe_Butte">Wikipedia</a>,
photos by the author.</em></p>
</body>
</html>

```



18.4. ábra

A 18.2. példa oldalának kezdeti megjelenése. Vegyük észre, hogy az egérmutató kézzé változik a piros szöveg felett, jóllehet az nem `<a>` elembeli hivatkozás

Először vizsgáljuk meg a stíluslap hat bejegyzését. Az első egyszerűen azoknak a hivatkozásoknak a stílusát határozza meg, amelyek egy `<a>` címkepáron belül helyezkednek el – ezek a hivatkozások félkövér, aláhúzott, kék színű feliratok lesznek. Ilyen hivatkozásokat találhatunk a 18.4. ábra két bekezdésében (valamint az oldal alján látható sorban).

A következő négy bejegyzés meghatározott azonosítókra vonatkozik, amelyek eszerint az oldal betöltésekor láthatatlanok (`display: none`). A következő két azonosítónál a `display: block` beállítás szerepel – igazából ezt meg sem kellett volna adnunk, ugyanis ez az alapértelmezés. A stíluslapon azért tüntettük fel mégis, hogy jól érzékelhető legyen a különbség. Ha összeszámoljuk a 18.1. példa `<div>` elemeit, hatot találunk: az oldal betöltésekor négy láthatatlan, kettő látható.

Példánkban azt szeretnénk elérni, hogy a két azonosító `display` értéke megváltozzon, ha egy másik azonosítóhoz tartozó elemre kattintunk. Először azonban arra van szükségünk, hogy a felhasználó egyáltalán észrevegye, hogy az adott szövegrészletre kattinthat – ezt rendszerint az egérmutató változása jelzi. Figyeljük meg a 18.4. ábrán, hogy az egérmutató kis kézzé alakul át a szóban forgó hivatkozás felett.

Ezt a hatást úgy érhetjük el, ha meghatározunk egy osztályt az adott szövegrészlet számára – osztályunk a `fakelink` nevet kapja, a hozzá tartozó kód pedig itt látható:

```
<div class="fakelink"
  id="show_e"
  onclick="this.style.display='none';
  document.getElementById('hide_e').style.display='block';
  document.getElementById('elevation').style.display='inline';
">&raquo; Show Elevation</div>
```

A `fakelink` osztály biztosítja, hogy a szöveg aláhúzás nélkül, félkövéren és vörös színnel jelenjen meg, a `cursor: pointer` beállítás pedig úgy változtatja meg az egérmutatót, hogy a felhasználó úgy érezze, egy `<a>` címkepáron belüli hivatkozást lát. Az igazán izgalmas dolgok azonban akkor kezdődnek, amikor összekapcsoljuk az `onclick` jellemzőt egy `<div>` elemmel. A fenti kódban az `onclick` jellemző értéke egy parancssorozat, amely módosítja a CSS-elemek aktuális értékeit.

Nézzük meg, milyen parancsokról is van szó:

```
this.style.display='none';
document.getElementById('hide_e').style.display='block';
document.getElementById('elevation').style.display='inline';
```

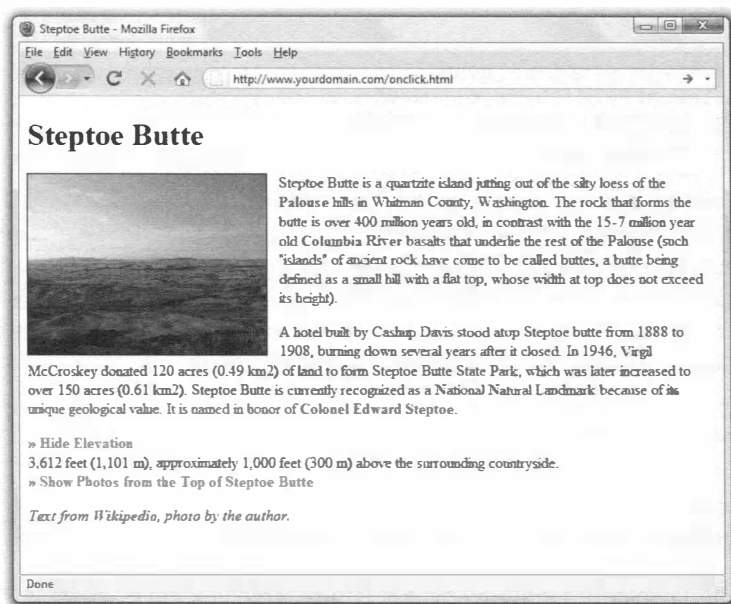
A fentiekben különféle JavaScript-tagfüggvényeket látunk, amelyek egyes elemek megváltoztatására szolgálnak. Bővebben majd a 21. órán ismerkedhetünk meg velük, jóllehet a JavaScript a maga teljességében túlmutat könyvünk keretein. Mindazonáltal a gondolatmenet követése nem okozhat nehézséget.

Az első sorban található `this` kulcsszó magára az elemre utal, más szóval a `show_e` azonosítóval jelölt `<div>` elemre. A `style` kulcsszó a stílusobjektumot adja meg, amely tartalmazza az összes, az elemhez rendelt CSS-stílust. Esetünkben a `display` stílus a legérdekesebb, nem véletlen hát, hogy a `this.style.display` jelentése „a `show_e` azonosító `display` stílusa”. A kódban pedig arról gondoskodunk, hogy ha a felhasználó a szövegre kattint, a `display` értékét `none`-ra állítsuk.

Ez azonban még nem minden, hiszen az `onclick` jellemzőn belül három műveletet találunk. A másik kettő a `document.getElementById()` függvényhívással indul, ahol a zárójelben meghatározott azonosítókat tüntetünk fel. Azért kényszerülünk ennek a függvénynek a használatára, mert a második és harmadik műveletben nem a szülő-elem `display` tulajdonságának az értékét állítjuk be. A kódból kitűnik, hogy itt a `hide_e` és az `elevation` azonosítókról van szó. Mindezek után összefoglalhatjuk, mi történik, ha a felhasználó a jelenleg látható, `show_e` nevű `<div>` elemre kattint:

- A `show_e <div>` láthatatlanná válik.
- A `hide_e <div>` láthatóvá válik, és `block` stílussal jelenik meg.
- Az `elevation <div>` láthatóvá válik, és `inline` stílussal jelenik meg.

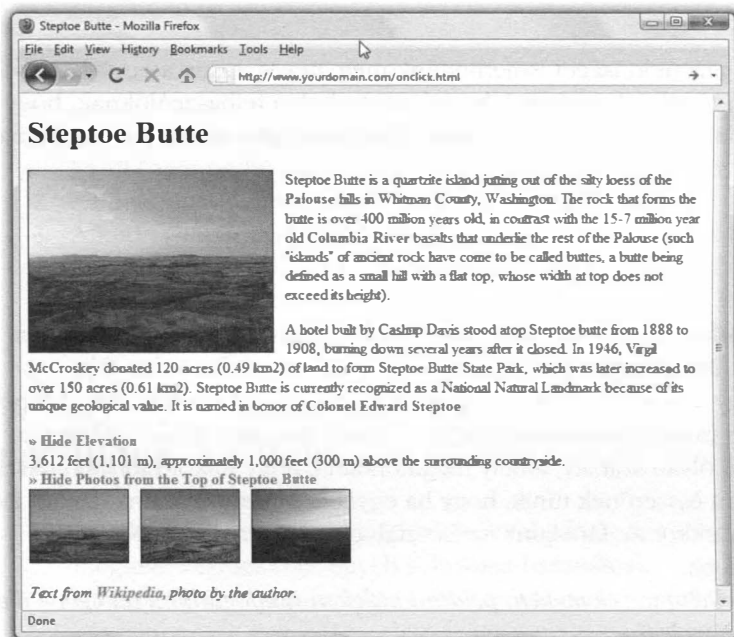
A műveletek eredményét a 18.5. ábrán figyelhetjük meg.



18.5. ábra

A `Show Elevation` elemre kattintva az aktuális és egyéb `<div>` elemek stílusa az `onclick` jellemzőben megadott parancsok szerint változik

A 18.2. példában láthatunk egy másik, `<div>` elemekből álló csoportot is, amely a további képek láthatóságát szabályozza. Ezekre az előbbiekben bemutatott onclick műveletek nincsenek hatással. Vagyis, akár a Show Elevation, akár a Hide Elevation lehetőségre kattintunk, a fotókhoz kapcsolódó `<div>` elemek nem változnak. Lehetőségünk van tehát arra, hogy megjelenítsük a magasságot (elevation), a fotókat azonban nem (lásd a 18.5. ábrát); a fotókat igen, de a magasságot nem; vagy pedig mind a fotókat, mind a magasságot (lásd a 18.6. ábrát).



18.6. ábra

Az oldal, miután a Show Elevation és a Show Photos from the Top of Steptoe Butte lehetőségekre kattintottunk

Ezzel a rövidke példával csak érzékeltetni akartuk, micsoda lehetőségek tárulnak fel előttünk, ha megtanuljuk összekapcsolni a CSS-stílusok módosítását az eseményekkel. Ennek a fegyvertárnak a birtokában olyan weboldalakat készíthetünk, amelyek stíluslapjainak elemeit a felhasználók módosíthatják, sőt akár egész más stíluslapot is használatba vehetnek, szövegrészleteket helyezhetnek át, kvízzjátékokban vehetnek részt, űrlapokat tölthetnek ki, és még sorolhatnánk a tengernyi lehetőséget.

Összefoglalás

Ezen az órán megtanultuk, hogyan befolyásolhatják az egérműveletek a szövegrészek megjelenítését. Az első két részben ezek a műveletek az `<a>` elem hover álosztályának stílusaihoz kapcsolódtak, később pedig megismerkedtünk a különböző felhasználói tevékenységeket leíró eseményekkel is. Példánkban az `onclick` eseményt használtuk, de felsoroltuk a további lehetséges eseményeket is, amilyen például az `onload` vagy az `onmouseover`.

A kódban egyetlen igazi újdonságot ismerhettünk meg, nevezetesen a `cursor` tulajdonságot. A `pointer` értéket hozzárendelve jelezni tudtuk a felhasználóknak, hogy az adott szövegrészlet valójában hivatkozásként viselkedik, jóllehet nem a megszokott `<a>` címkék között tüntettük fel a kódban.

Kérdezz-felelek

- K:** *Az órán bemutatott egyes események kísértetiesen hasonlítanak az `<a>` elem álosztályaira. Miben áll mégis a különbség?*
- V:** Való igaz, hogy az `onmousedown` esemény igencsak emlékeztet az aktív állapotra, az `onmouseup` és az `onmouseover` pedig a „felette lebegés” (hover) állapotra. Nem ismeretes olyan szabály, amely megmondaná, hogy mikor melyiket használjuk, az azonban ésszerűnek tűnik, hogy ha egyébként semmi okunk az `<a>` elem alkalmazására, akkor ne fáradjunk az álosztályokkal, és inkább forduljunk az eseményekhez.
- K:** *A szövegek mellett más elemeken, például képeken is elfoghatjuk az egér-, illetve billentyűeseményeket?*
- V:** Igen, a szövegekhez hasonlóan a képeken is észlelhetjük a kattintást, illetve az egérmutató helyzetét. Fontos tudnunk azonban, hogy más multimédia-objektumok, például a YouTube-videók vagy a Flash-fájlok, nem képesek ilyen viselkedésre, illetve más programok segítségével, eltérő módon értelmezik a billentyűzetről és az egértől származó bemenetet. Így ha például egy weboldal szerkezetébe ágyazott YouTube-videóra kattintunk, a YouTube-lejátszóval lépünk kapcsolatba, nem pedig az azt befogadó oldallal – a műveletet tehát nem foghatjuk el az itt megismert módon.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Mi történik, ha a `top` tulajdonságnak negatív értéket adunk?
2. Melyik eseményt használhatjuk, ha akkor szeretnénk valamit módosítani, amikor a felhasználó az egérmutatót az adott elem határain kívülre húzza?
3. Vajon mire következtethetünk abból, ha egy stíluslapon a `cursor: crosshair` kódot találjuk?

Válaszok

1. A `top` negatív értékének hatására a tartalom a szülőelem bal felső sarka fölé kerül (nem pedig alá).
2. Az `onmouseout` eseményt.
3. A kód egy nagy pluszjellé (vagy szálkeresztté) alakítja az egérmutatót. Arra azonban nehéz választ adni, hogy mi értelme ennek a kódnak, ugyanis ezt az alakot meglehetősen ritkán használják. A felhasználó tehát töprengeni kezd, hogy vajon milyen célt szolgál ez a változás, és tapasztalatok híján jó eséllyel kételyek maradnak benne.

Gyakorlatok

- Készítsünk egy oldalt lebegő leírásokkal, amelyek a megjelenítési sablonunknak megfelelő színeket és egyéb stílusokat használnak.
- Adjunk további parancsokat a 18.2. példa `onclick` jellemzőihez úgy, hogy egyszerre csak egy `<div>` elem (vagy a magasság, vagy a fotók) legyen látható.

19. ÓRA



Rögzített és folyékony elrendezések létrehozása

A lecke tartalma:

- A rögzített elrendezések működése
- A folyékony elrendezések működése
- Rögzített-folyékony kevert elrendezés létrehozása

Könyvünk nagyobb részében a webes tartalom stílusának meghatározásával foglalkozunk, így beállítottuk a betűméretet és -színt, a képek megjelenítését, tömbelemeket és listákat használtunk, és még megannyi más eszközt. Most azonban kissé távolabbról tekintünk a weboldalaink szerkezetére. Általánosságban véve kétféle – rögzített (fixed) és folyékony (liquid) – elrendezés ismeretes, jóllehet létezik a kettő egyvelege is, amelyben egyes elemek rögzítettek, mások viszont folyékonyak.

Ezen az órán először az említett kétféle elrendezés jellemzőivel foglalkozunk, és megtekintünk néhány példát a használatukra, az óra végén pedig bemutatunk egy egyszerű sablont, amely egyesíti a két elrendezés lehetőségeit. A döntés végeredményben a mi kezünkben van – ha követjük a HTML- és CSS-szabványokat, nagyot nem tévedhetünk.



Önálló feladat

Keressünk tetszetős elrendezéseket!

A folyékony elrendezések valóságos kincsesbányájára bukkanhatunk a Wordpress Theme Gallery-ben (<http://wordpress.org/extend/themes/>). A WordPress eredetileg webnaplómotornak készült, de egyre nagyobb népszerűsége tesz szert, mint általános webhelykezelő eszköz. A galériában több száz rögzített és folyékony elrendezést találhatunk, amelyek legalábbis jó iránymutatást adhatnak a munkánkhoz. Ha könyvünk feladatai körében nem is kellett WordPress-webnaplóval dolgoznunk, a sablongaléria nagyszerű hely arra, hogy felfedező útra induljunk az oldal-elrendezések világában.

Töltsünk némi időt a WordPress-sablonok körében, illetve keressük fel a CSS Zen Gardent (<http://www.csszengarden.com/>). Így anélkül ismerkedhetünk meg az ízlésünknek megfelelő elrendezésekkel, hogy a tartalom eltérítene a figyelmünket.

Rögzített elrendezések

A rögzített, illetve rögzített szélességű elrendezések egyetlen alapvető közös tulajdonsága, hogy az oldal törzsének szélessége rögzített érték. Ezt a szélességet többnyire egy fő, „burkoló” `<div>` szabályozza, amelyben benne foglaltatik az oldal teljes tartalma. A `<div>` width (szélesség) tulajdonságát a `style` jellemzőben, illetve egy stíluslap-bejegyzésben szabályozhatjuk, a megfelelő azonosítóra hivatkozva – alkalmazhatjuk a `main` (fő) vagy a `wrapper` (burkoló) nevet, de más elnevezés mellett is dönthetünk.



Ne feledkezzünk meg arról, hogy a böngészőablak felülete tartalmaz a weboldalon kívüli területeket is, ilyen például a gördítőszáv. Így ha 1024 képpont széles képernyőre szánjuk az oldalunkat, vegyük számításba, hogy nem használhatjuk ki a képpontok mindegyikét.

Rögzített szélességű elrendezés készítésénél a legfontosabb döntést annak meghatározása jelenti, hogy milyen képernyőfelbontással számolunk. Sokáig a 800 x 600-as felbontás volt a webtervezők közös nevezője, így nagyjából 760 képpontban rögzítették az oldalak szélességét. Mivel azonban 2007-ben már mindössze a felhasználók kevesebb mint 8 százaléka használt 800 x 600-as felbontást (jelenleg pedig ez az érték 4 százalék alatt jár), a tervezők 1024 x 768-ra emelték az általánosan elfogadott felbontásértéket, így manapság, ha egy oldal szélességét rögzíteni kívánják, jobbára valamilyen 800 és 1000 képpont közötti értéket választanak.

A rögzített szélességű oldalak használatának legfontosabb oka, hogy így pontosan szabályozhatjuk a tartalom megjelenését. Ha azonban a felhasználók a tervezettnél jóval kisebb vagy nagyobb képernyőfelbontással tekintik meg az oldalunkat, gördítősávok jelenhetnek meg (amennyiben a felbontás kisebb), vagy túl sok üres terület (ha a felbontás nagyobb).

Az ESPN.com nyitólapja nagyszerűen mutatja a jelenséget – az oldal a tartalmát 964 képpont szélességű területre korlátozza. A 19.1. ábrán a böngészőablakunk 800 képpont széles, így a jobb oldalon fontos részek kerülnek ki a látómezőnkől (ezek elérésére az alul megjelenő vízszintes gördítősáv szolgál).

A 19.2. ábra ugyanakkor azt a helyzetet mutatja, amikor a böngészőablak 1300 képpont szélességű – ilyenkor az oldal törzse mellett mindkét oldalon kiterjedt üres területeket találunk.



19.1. ábra

Rögzített szélességű weboldal kisebb képernyőfelbontásnál



19.2. ábra

Rögzített szélességű weboldal nagyobb képernyőfelbontásnál

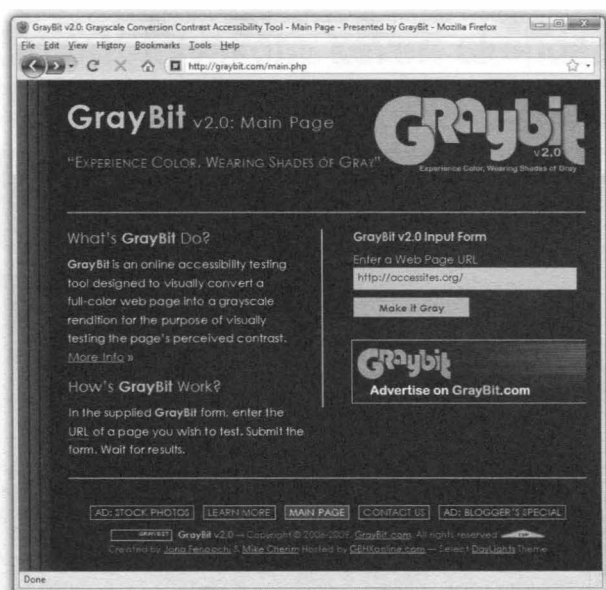
Folyékony elrendezések

A folyékony elrendezések jellemzője, hogy az oldal törzsének nincs képpontban meghatározható, rögzített szélessége, jóllehet előfordulhat, hogy egy burkoló `<div>` foglalja magába, amelynek a szélessége százalékban meghatározott. Ha folyékony elrendezést használunk, megőrizhetjük az oldal tetszetős összképét, válasszon látogatónk akár kicsi, akár nagy képernyőfelbontást.

Az eredményt jól mutatja a 19.3., a 19.4. és a 19.5. ábra.

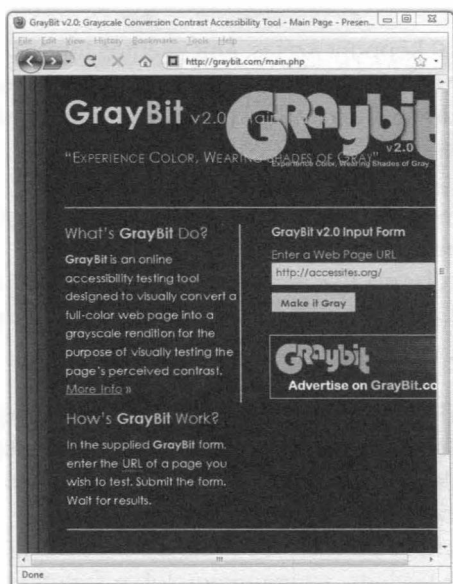
A 19.3. ábrán a böngészőablak szélessége nagyjából 770 képpont. Ez ésszerű minimális szélességérték ahhoz, hogy a gördítősáv még ne jelenjen meg – erre nem is kerül sor egészen a 735 képpontos szélességig. A hűrt persze túlfeszíthetjük, amint azt a 19.4. ábrán, 545 képpontos szélesség mellett láthatjuk is.

A 19.4. ábrán már megjelenik a vízszintes gördítősáv, a fejlécben pedig azt láthatjuk, hogy az embléma ábrája rácsúszik a szövegre. Mindazonáltal az oldal nagyobb része még mindig jól használható. A bal oldalon található lényegi tartalom tökéletesen olvasható, és sikeresen osztozik a rendelkezésre álló területen a jobb oldali beviteli úrlappal.



19.3. ábra

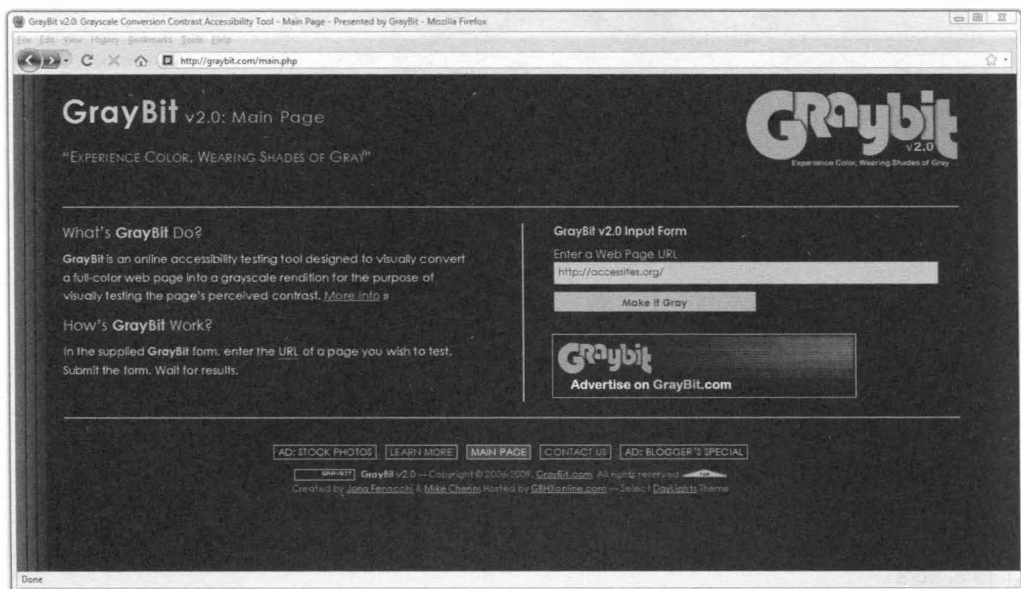
Folyékony elrendezés viszonylag kis felbontású képernyőn



19.4. ábra

Folyékony elrendezés kis képernyőn

A 19.5. ábrán láthatjuk, hogyan fest az oldal egy igen széles képernyőn.



19.5. ábra

Folyékony elrendezés széles képernyőn

A 19.5. ábrán a böngészőablak nagyjából 1330 képpont széles, ami jelentős teret ad az oldalnak a terjeszkedésre. A folyékony elrendezést az teszi lehetővé, hogy a szerkezeti elemek szélessége százalékban van megadva (nem pedig képpontban), így az oldal képes kitölteni a rendelkezésére álló területet.

A folyékony elrendezés első látásra a lehető legjobb megoldásnak tűnhet, hiszen ki ne szeretné kihasználni a képernyő összes rendelkezésre álló területét? Nos, a helyzet nem ilyen egyszerű – igen finom határvonal választja el a lehetséges terület kihasználását a tartalom „megfojtásától”. A túl sűrű tartalom nyomasztó, a túlzottan ritkás elrendezés viszont unalmassá teszi az oldalt.

A tisztán folyékony elrendezés vonzó megoldás, de rengeteg vizsgálatot igényel, hogy biztosítsuk a megfelelő megjelenést szinte bármilyen böngésző és képernyőfelbontás használata mellett. Megeshet, hogy sem időnk, sem kedvünk nincs ekkora áldozatot hozni. Ilyenkor segít a rögzített-folyékony kevert (hibrid) elrendezés.

Rögzített-folyékony kevert elrendezések

Az ilyen elrendezések mindkét típusból származó elemeket tartalmaznak. Így például alkalmazhatunk folyékony elrendezést, amely ugyanakkor tartalmaz rögzített szélességű részeket a törzsön belül, illetve lehorgonyzott elemként (ilyen lehet a bal oldali hasáb vagy a felső navigációs sáv). Arra is módunk van, hogy egy rögzített szélességű területet keretként használjunk, amely akkor is a helyén marad, ha a felhasználó görgeti a tartalmat (lásd a 13. óra anyagát).

Kiindulópontunk, egy egyszerű elrendezés

Ebben a példában egy olyan sablon készítését tanuljuk meg, amely alapján véve folyékony, ugyanakkor egy-egy rögzített szélességű hasábot tartalmaz a törzs két oldalán (ez utóbbi szintén egy hasábnak tekinthető, amely szélesebb a társainál). A sablon, amelynek a HTML-kódját a 19.1. példa mutatja, önálló fejléccel és lábléccel is rendelkezik.

19.1. példa Egyszerű rögzített-folyékony kevert elrendezés

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Layout</title>
    <link href="layout.css" rel="stylesheet" type="text/css" />
  </head>

  <body>
    <div id="header">HEADER</div>
    <div id="wrapper">
      <div id="content_area">CONTENT</div>
      <div id="left_side">LEFT SIDE</div>
      <div id="right_side">RIGHT SIDE</div>
    </div>
    <div id="footer">FOOTER</div>
  </body>
</html>
```

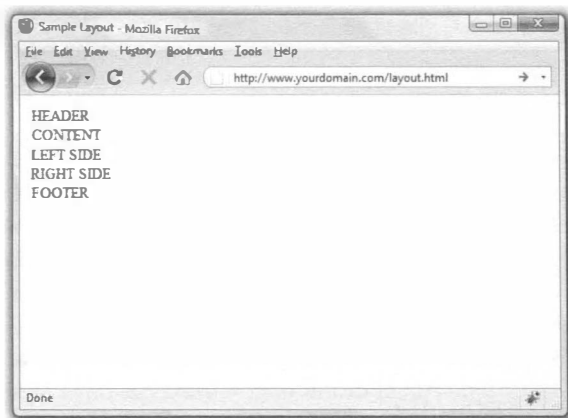
Először is vegyük észre, hogy az elrendezéshez tartozó stíluslapot a `<link>` elemmel csatoljuk a dokumentumhoz, ahelyett, hogy közvetlenül a kódban tüntetnénk fel. Mivel a sablont több oldalhoz is használjuk, az elemei megjelenítését a lehető legfegyelmezettebb módon szeretnénk szabályozni. Ezzel elérjük, hogy ha az elemek stílusainak módosítására lenne szükség, azt egy helyen – a stíluslapon – tehessük meg.

A következő, ami feltűnhet, hogy a HTML-alapkód valóban csak az alapvető elemekre szorítkozik. Ráadásul az igazat megvallva ebből a kódból kiindulva egyaránt létrehozhatunk rögzített, folyékony vagy rögzített-folyékony kevert elrendezést, mivel a megjelenést a stíluslap tartalma határozza meg.

A 19.1. példában mindössze kijelöljük a webhely tartalmának egyes területeit. Ez a tervezés fontos alapfeltétele bármiféle webes fejlesztésnek, hiszen még azelőtt tudnunk kell, hogy mit szeretnénk feltüntetni az oldalon, mielőtt az elrendezés típusát és különösképpen a stílusok részleteit meghatároznánk. Jelen pillanatban a `layout.css` tartalma mindössze ennyi:

```
body {
    margin:0;
    padding:0;
}
```

Ha most a 19.1. példára pillantunk, így kiálthatunk fel: „de hát ezek a `<div>` elemek stílusok híján egymás hegyén-hátán torlódnak!” A megállapítás helyes: amint a 19.6. ábrán is láthatjuk, elrendezésről még nem igazán beszélhetünk.



19.6. ábra

Egyszerű HTML-sablon, amelyben a `<div>` elemekre még nem hatnak stílusok

Két hasáb meghatározása egy rögzített-folyékony elrendezésben

Kezdjük az egyszerűbb feladatokkal! Mivel folyékony elrendezést szeretnénk kialakítani, tudjuk, hogy akármit is teszünk a fej- és láblécbe, az kitölti a böngészőablak teljes szélességét, bármekkora is legyen.

Az alábbi kódot a stíluslapon elhelyezve 100 százalékos szélességűre állítjuk a fej- és láblécet, továbbá azonos háttérszínt határozunk meg a számukra:

```
#header {
  float: left;
  width: 100%;
  background-color: #7152F4;
}
#footer {
  float: left;
  width: 100%;
  background-color: #7152F4;
}
```

A dolog most kezd izgalmassá válni. Meg kell határoznunk két rögzített szélességű hasábot az oldal szélein, valamint egy folyékony hasábot középen. Vegyük észre, hogy a kódban található egy wrapper névre hallgató <div> elem, amely mindhárom hasábot magában foglalja. A meghatározása így fest:

```
#wrapper {
  float: left;
  padding-left: 200px;
  padding-right: 125px;
}
```

A két kitöltésérték lényegében a jobb és bal oldali rögzített szélességű hasáboknak ad helyet. Az előbbi 200, míg az utóbbi 125 képpont széles lesz, emellett különböző háttérszín is kapnak. A hasábok elhelyezését eredeti, stílusok nélküli változataikhoz (lásd a 19.6. ábrát) képest határozzuk meg, erre szolgál a stíluslap bejegyzéseiben elhelyezett position: relative kód. Végül azt is jelezzük, hogy a <div> elemek balra ússzanak.

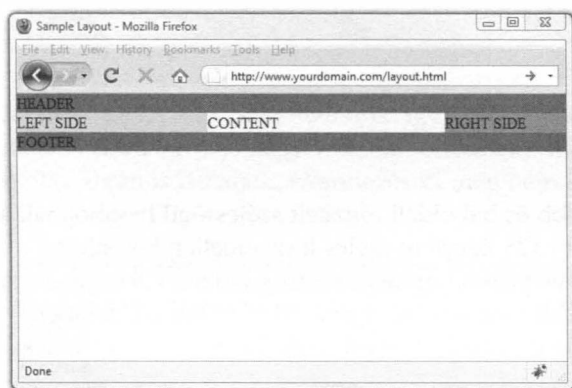
A left_side nevű <div> esetében azt is rögzítenünk kell, hogy a jobb szélső margót az elem szélétől 200 képpontra szeretnénk helyezni (ez hozzáadódik a hasáb 200 képpontos szélességéhez). A bal oldalon ugyanakkor teljes kiterjedésű negatív margót hozunk létre, amely éppen a helyére húzza az elemet (amint azt hamarosan látni is fogjuk). A right_side <div> esetében a right értéket nem határozzuk meg, de a jobb oldalon szükségünk van egy negatív margóra:

```
#left_side {
  position: relative;
  float: left;
  width: 200px;
  background-color: #52f471;
  right: 200px;
  margin-left: -100%;
}
#right_side {
  position: relative;
  float: left;
  width: 125px;
  background-color: #f452d5;
  margin-right: -125px;
}
```

Határozzuk most meg a központi területet is, mégpedig fehér háttérrel, valamint úgy, hogy kitöltse a rendelkezésre álló területet, és balra ússzon a jelenlegi helyzetéhez képest:

```
#content_area {
    position: relative;
    float: left;
    background-color: #ffffff;
    width: 100%;
}
```

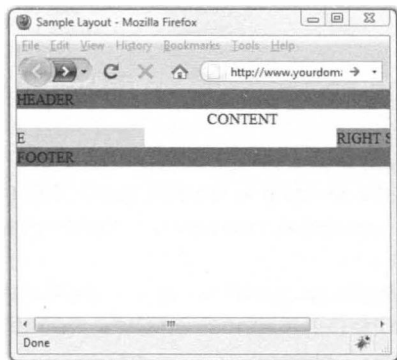
Az elrendezés most a 19.7. ábra szerint alakul. Láthatjuk, hogy a területek szépen elválnak egymástól.



19.7. ábra

Az egyszerű HTML-sablon néhány stílus meghatározása után

Akad azonban némi gond a sablonnal, ha meghatározott érték alá csökkentjük a böngészőablak szélességét. Mivel a bal oldali hasáb 200, a jobb oldali pedig 125 képpont szélességű, és valamicske szöveget azért elszeretnénk helyezni a központi területen, az oldal széttöredezik, ha a böngészőablak szélessége 350-400 képpontig csökken. A 19.8. ábrán láthatjuk, mi történik, ha az ablak szélességét kicsivel 400 képpont alá csökkentjük (a pontos érték 390 képpont).



19.8. ábra

Az egyszerű HTML-sablon 400 képpontnál keskenyebb böngészőablakban – katasztrófa!

Minimális oldalszélesség beállítása

A való életben igencsak valószínűtlen, hogy a látogatóink 400 képpontnál keskenyebb böngészőablakban nyissák meg az oldalunkat, de könyvünk keretein belül elképzelhetjük ezt a helyzetet. A példa tanulságai kiterjeszthetők és szélesebb körben értelmezhetők – könnyen belátható, hogy még rögzített-folyékony kevert elrendezések esetén is eljuthatunk egy pontra, ahol az oldal szétesik, hacsak nem teszünk ellene valamit.

Ez a „valami” nem más, mint a `min-width` tulajdonság beállítása, amellyel meghatározhatjuk egy elem legkisebb szélességét (ebbe nem számítjuk bele a belső és külső margókat vagy a szegélyeket). A 19.9. ábrán láthatjuk, milyen eredményt kapunk, ha a `min-width` tulajdonságot a `<body>` elemre alkalmazzuk.



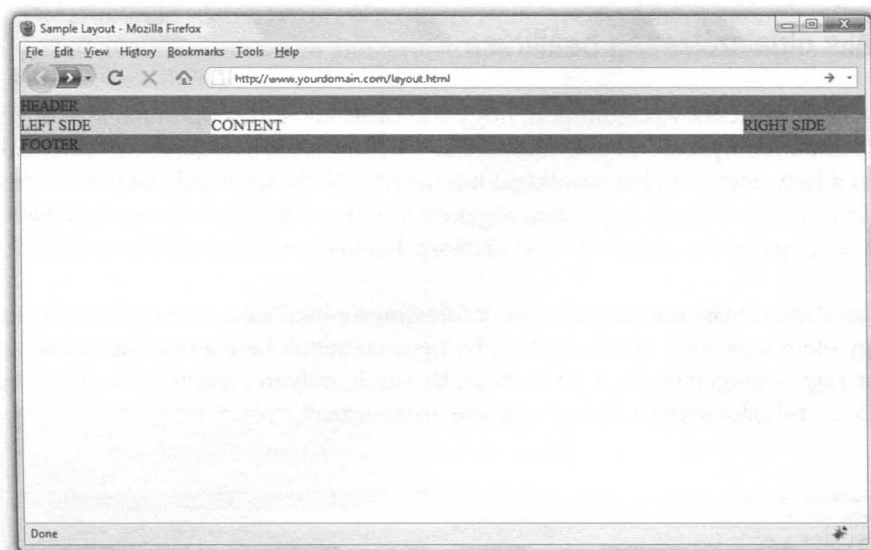
19.9. ábra

Az egyszerű HTML-sablon 400 képpont alá csökkentett mérettel – ez már tetszetősebb!

Balra görgetve az oldalt csak egy aprócska rész látszik a jobb oldali hasázból, viszont az elrendezés így nem esik szét. Ez esetben a minimális szélesség 525 képpont:

```
body {
margin: 0;
padding: 0;
min-width: 525px;
}
```

A vízszintes gördítőszávon azért jelenik meg a példában, mert maga a böngészőablak keskenyebb 500 képpontnál. Ha egy leheletnyivel 525 képpont fölé nagyítjuk, a gördítőszávon eltűnik, ha pedig a böngészőablak szélessége nagyjából 875 képpont (lásd a 19.10. ábrát), egyáltalán nem kell számítanunk a megjelenésére.



19.10. ábra

Egyszerű HTML-sablon egy 800 képpontnál szélesebb böngészőablakban

A hasábk magasságának kezelése rögzített-folyékony kevert elrendezésben

A példánk jónak tűnik – az egyetlen gond vele, hogy még nem töltöttük fel tartalommal. A különböző elemekhez adott tartalom pedig újabb kérdéseket vet fel. A 19.11. ábrán láthatjuk, hogy a hasábk magassága a bennük tárolt tartalomnak megfelelően alakul.

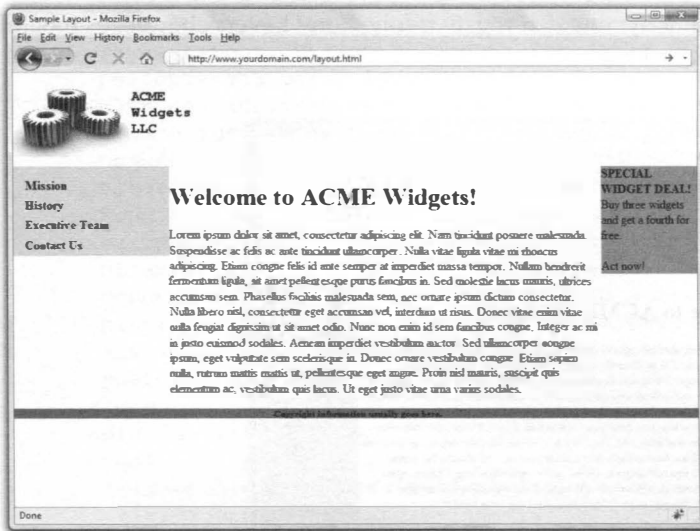
Mivel semmi sem biztosítja a felhasználók böngészőablakának azonos magasságát, illetve azt, hogy a tartalom mindig ugyanolyan méretű legyen, azt gondolhatnánk, hogy itt gondok adódnak a rögzített-folyékony kevert elrendezésekkel. Szerencsére nem ilyen rossz a helyzet. Némi gondolkodás után, pár stílus segítségével összerakhatjuk a kirakósjáték darabjait.

Először is helyezzük el az alábbi kódsorokat a `left_side`, a `right_side` és a `content_area` stíluslap-bejegyzésekben:

```
margin-bottom: -2000px;
padding-bottom: 2000px;
```

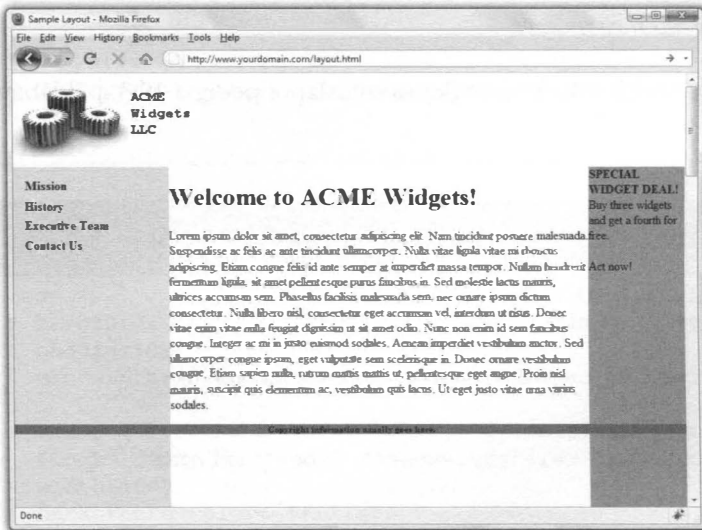
Ezzel nevetségesen nagy méretű belső és külső margót helyezünk el mindhárom elem alján. Szükség van továbbá a lábléc stíluslap-bejegyzésében a `position: relative` sorra, amely biztosítja, hogy az elem a kitöltés ellenére látható maradjon.

Ebben a pillanatban az oldal a 19.12. ábrán látható alakot ölti – ez még nem egészen az, amire vágytunk, de már alakul az összkép.



19.11. ábra

A hasábkok magasságát a tartalmuk határozza meg



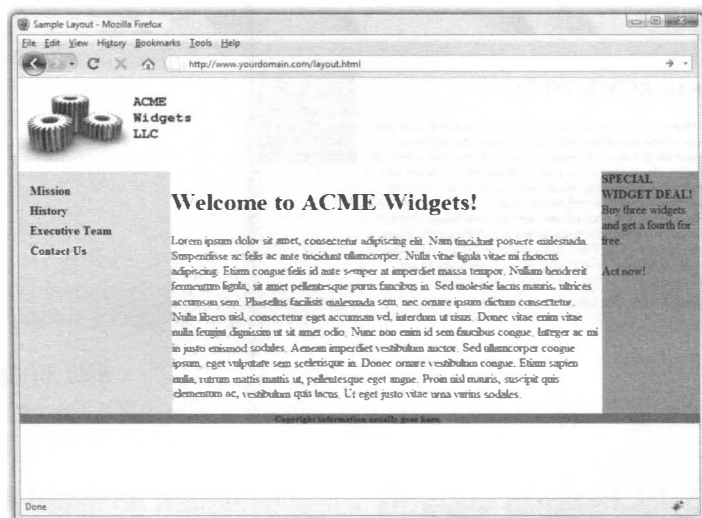
19.12. ábra

A színes mezők immár hosszan elnyúlnak a rövid tartalom ellenére

A feleslegesen túlfutó színeket lenyeshetjük, ha az alábbi beállítással élünk a wrapper `<div>` stíluslap-bejegyzésében:

```
overflow: hidden;
```

A 19.13. ábra a végeredményt mutatja: rögzített-folyékony kevert elrendezés megfelelő térközökkel ellátott hasábokkal.



19.13. ábra

A kész rögzített-folyékony kevert elrendezés

A weboldal teljes HTML-kódját a 19.2., a végleges stíluslapot pedig a 19.3. példában láthatjuk.

19.2. példa *Egyszerű rögzített-folyékony kevert elrendezés (tartalommal)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Sample Layout</title>
    <link href="layout.css" rel="stylesheet" type="text/css" />
  </head>

  <body>
    <div id="header"></div>
    <div id="wrapper">
```

```

<div id="content_area">
  <h1>Welcome to ACME Widgets!</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Nam tincidunt posuere malesuada. Suspendisse ac felis ac ante
  tincidunt ullamcorper. Nulla vitae ligula vitae mi rhoncus
  adipiscing. Etiam congue felis id ante semper at imperdiet
  massa tempor. Nullam hendrerit fermentum ligula, sit amet
  pellentesque purus faucibus in. Sed molestie lacus mauris,
  ultrices accumsan sem. Phasellus facilisis malesuada sem, nec
  ornare ipsum dictum consectetur. Nulla libero nisl,
  consectetur eget accumsan vel, interdum ut risus. Donec
  vitae enim vitae nulla feugiat dignissim ut sit amet odio.
  Nunc non enim id sem faucibus congue. Integer ac mi in justo
  euismod sodales. Aenean imperdiet vestibulum auctor. Sed
  ullamcorper congue ipsum, eget vulputate sem scelerisque in.
  Donec ornare vestibulum congue. Etiam sapien nulla, rutrum
  mattis mattis ut, pellentesque eget augue. Proin nisl mauris,
  suscipit quis elementum ac, vestibulum quis lacus. Ut eget
  justo vitae urna varius sodales. </p>
</div>
<div id="left_side">
  <ul>
    <li><a href="#">Mission</a></li>
    <li><a href="#">History</a></li>
    <li><a href="#">Executive Team</a></li>
    <li><a href="#">Contact Us</a></li>
  </ul>
</div>
<div id="right_side"><strong>SPECIAL WIDGET DEAL!</strong><br/>
  Buy three widgets and get a fourth for free.<br/><br/>
  Act now!
</div>
</div>
<div id="footer"> Copyright information usually goes here.</div>
</body>
</html>

```

19.3. példa A rögzített-folyékony kevert elrendezés teljes stíluslapja

```

body {
  margin:0;
  padding:0;
  min-width: 525px;
}
#header {
  float: left;
  width:100%;
  background-color: #ffffff;
}
#footer {
  float: left;
  width:100%;

```

```
background-color: #7152f4;
font-size: 8pt;
font-weight: bold;
text-align: center;
position: relative;
}
#wrapper {
float: left;
padding-left: 200px;
padding-right: 125px;
overflow: hidden;
}
#left_side {
position: relative;
float: left;
width: 200px;
background-color: #52f471;
right: 200px;
margin-left: -100%;
padding-bottom: 2000px;
margin-bottom: -2000px;
}
#right_side {
position: relative;
float: left;
width: 125px;
background-color: #f452d5;
margin-right: -125px;
padding-bottom: 2000px;
margin-bottom: -2000px;
}
#content_area {
position: relative;
float: left;
background-color: #ffffff;
width: 100%;
padding-bottom: 2000px;
margin-bottom: -2000px;
}
#left_side ul {
list-style: none;
margin: 12px 0px 0px 12px;
padding: 0px;
}
#left_side li a:link, #nav li a:visited {
font-size: 12pt;
font-weight: bold;
padding: 3px 0px 3px 3px;
color: #000000;
text-decoration: none;
display: block;
}
#left_side li a:hover, #nav li a:active {
```

```
font-size: 12pt;
font-weight: bold;
padding: 3px 0px 3px 3px;
color: #ffffff;
text-decoration: none;
display: block;
}
```

Összefoglalás

Ezen az órán néhány gyakorlati példát láthattunk az elrendezések három alapvető típusának – rögzített, folyékony és rögzített–folyékony kevert – bemutatására. Az óra harmadik részében lépésről lépésre bemutattuk, miként hozhatunk létre rögzített–folyékony kevert elrendezést, amelyben mind a HTML-, mind a CSS-kód érvényes. Ne feledjük, hogy az elrendezés létrehozásakor a legfontosabb feladat, hogy számba vegyük a tartalomnak azokat a részeit, amelyeknek szerkezeti elemeket kell megfeleltetnünk.

Kérdezz-felelek

- K:** *Nemcsak folyékony, hanem rugalmas elrendezésről is hallhatunk. Mi a különbség a kettő között?*
- V:** A rugalmas elrendezések jellemzője, hogy a tartalmat tároló területek mérete a szövegméret módosításának hatására megváltozik bennük. A rugalmas elrendezésben az „em” használatos mértékegységként, amely a meghatározásából eredően arányos a szöveg-, illetve a betűmérettel, hiszen megegyezik az adott betűtípus pontméretével. Ha tehát a tárolóelemek méretét em-ben adjuk meg, a szöveggel együtt tágulnak, illetve zsugorodnak, ha a felhasználó a CTRL billentyű lenyomásával és az egérgörgő mozgatásával növeli vagy csökkenti a szöveg méretét. A rugalmas elrendezések kialakítása fájdalmasan nehéz feladat, így leginkább a webtervezők bemutatóanyagaiban találkozhatunk velük, a való életben igen ritkán kerülnek elénk.
- K:** *Sokat hallhattunk a folyékony és a kevert elrendezésről – valóban jobbak lennének, mint a rögzített elrendezés?*
- V:** Nos, a „jobb” meglehetősen szubjektív fogalom, de annyit mindenesetre leszögezhetünk, hogy a szabványok követése mindenképpen jó pontokat jelent. Ha egy webtervezőt kérdeznénk, bizonyára azt mondaná, hogy a folyékony elrendezések létrehozása (és tökéletesítése) hosszabb időt vesz igénybe, de a használhatóság terén nyert előnyök bőségesen kárpótolnak a befektetésünkért. Persze ha a megbízónkat nem igazán érdekli az eredmény minősége, és nem értékeli anyagiakban is a munkára áldozott időt, felesleges energiabefektetésnek tekinthetjük az egészet. Ilyenkor, ha másért nem is, a saját képességeink csillogtatásért érdemes a göröngyösebb, de magasabbra vezető utat választanunk.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Melyik elrendezés a legjobb: a rögzített, a folyékony, vagy a rögzített-folyékony kevert?
2. Van lehetőségünk egy rögzített elrendezés tetszőleges elhelyezésére a böngészőablakon belül?
3. Mi a szerepe a `min-width` tulajdonságnak?

Válaszok

1. Ez beugratós kérdés, hiszen nincs „legjobb” elrendezés. A választás a tartalomtól és közönség igényeitől függ.
2. Persze. Jóllehet a legtöbb oldal tartalmát balra vagy középre igazítják, lehetőségünk van arra, hogy az elrendezés burkoló `<div>` elemét az X és az Y tengely mentén tetszőleges mértékben eltoljuk.
3. A `min-width` tulajdonság az adott elem lehetséges legkisebb szélességét határozza meg, amelybe nem értjük bele a kitöltés (belső margó), a szegély, valamint a (külső) margó értékét.

Gyakorlatok

- A 19.3. ábra a rögzített-folyékony kevert elrendezés „végső” változatát mutatja, de feltűnhet, hogy van még mit javítani rajta: a jobb oldali hasámban nincs térköz a szöveg körül, a törzs és a hasámbok között nem láthatunk margót, a lábléc kicsit ritkás, és így tovább. Szánjunk rá némi időt, hogy kijavítsuk ezeket az apró hiányosságokat.
- Miután az előbbi feladatban meghatároztuk a megfelelő kitöltést vagy margót, a 17. órán tanultak alapján gazdagítsuk az oldalt egy vízszintes navigációs sávval, a függőleges navigációt pedig csinosítsuk ki.



20. ÓRA

Nyomtatóbarát weboldalak készítése

A lecke tartalma:

- Mi teszi nyomtatóbaráttá a weboldalakat?
- A megjelenési formához illő stíluslap használata
- A nyomtatható oldalakhoz való stíluslap elkészítése
- A weboldalak nyomtatási előnézetének megtekintése

Ha az Olvasó használt már olyan internetes térképeket, mint a MapQuest vagy a Google Maps, kétségtelenül került már olyan helyzetbe is, amikor szeretett volna egy weboldalt kinyomtatni. A nyomtatóbarát weboldalak iránti igényt növeli továbbá az Interneten beszerezhető vásárlási kuponok tömkelege, az internetes vásárlások bizonylatai, a webes felületen elvégezhető repülőgépes helyfoglalás, illetve az otthon is kinyomtatható szállókártya. A dolog úgy áll, hogy nem minden weboldalt terveztek kizárólag monitoron való olvasásra. Lehet, hogy az Olvasó nincs is tudatában annak, hogy lehetősége van megtervezni és elérhetővé tenni a honlapja adott oldalainak nyomtatóbarát változatait is – kifejezetten azoknak a felhasználóknak a kedvéért, akik kinyomtatva, papírról szeretnék elolvasni az adott oldal tartalmát. Ilyet ajánl az olvasó-

nak a Google Maps is, miután a tartalom monitorra szánt változatát már megmutatta. A CSS technológia használatával könnyedén hozhatunk létre olyan weboldalakot, amelyek a megtekintés módjának megfelelően változtatják a kinézetüket. Ezen az órán az ilyen weboldalak létrehozását fogjuk elsajátítani.



Önálló feladat

A tartalom vizsgálata a nyomtathatóság szempontjából

Az óra anyagának olvasgatása alatt időről időre idézzük fel az egyik olyan weboldalunkat, amelyik szépen mutatna kinyomtatva is. Ilyenkor gondolkodjunk el azon, hogy mit változtatnánk az oldalon annak érdekében, hogy a nyomtatott lapon még szebb legyen. Íme pár megfontolásra érdemes ötlet:

- Elképzelhető, hogy az előző fejezetekben lévő figyelmeztetések ellenére van még olyan weboldalunk, amelynek a háttérét ismétlődő kép alkotja, illetve az, hogy valamilyen szokatlan háttérszínre írtuk a szöveget egy attól elütő színnel? Az efféle oldalakat a háttér miatt bonyolult lehet kinyomtatni, érdemes tehát elgondolkoznunk olyan nyomtatásra szánt változat kialakításán, amely sem háttérképet, sem háttérszínt nem használ, és egyszerű fekete betűkkel készül. Ha egy oldal nyomtatható változatán munkálkodunk, érdemes a fehér háttér és a fekete szöveg használatánál maradni.
- Az oldalaink sok hivatkozást tartalmaznak? Ha igen, akkor érdemes megfontolnunk a hivatkozások megváltoztatását valamilyen kevésbé kirívó formára – például távolítsuk el az aláhúzásokat. Gondoljunk csak bele: a papíron úgymint hiába kattintgatunk.
- Valóban létfontosságú az oldalon valamennyi kép? A színes képek kinyomtatása a legtöbb nyomtatón igen sokat elfogyaszt a drága tintából. Érdemes tehát elgondolkoznunk azon, hogy elhagyjunk-e pár – vagy akár az összes – képet az oldalaink nyomtatásra szánt változatából.

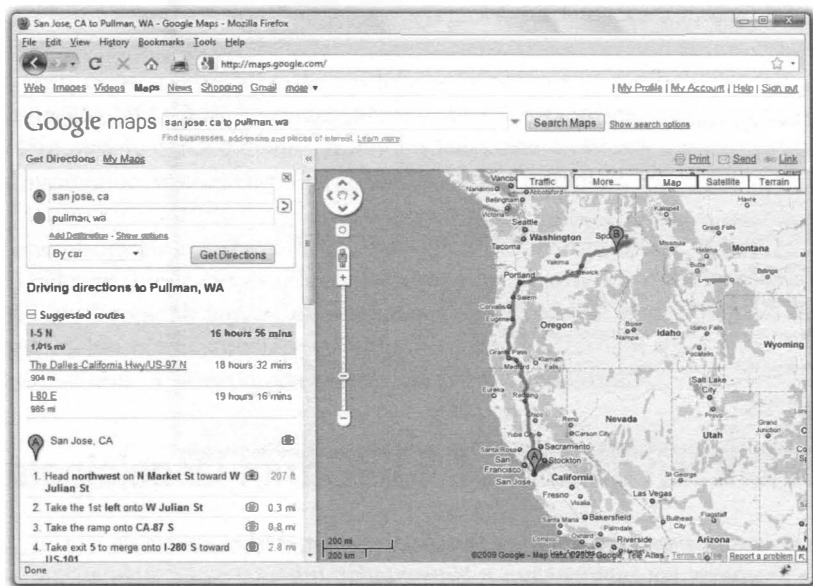
Mi teszi nyomtatóbaráttá a weboldalakot?

A fenti kérdést az előző *Önálló feladat* részben már lényegében megválasztottuk, de érdemes mélyebb magyarázatot szentelnünk annak, hogy mi az, ami a weboldalakot egyszerűen nyomtathatóvá teszi. Mindenekelőtt rá kell mutatnunk, hogy van néhány weboldal, ami máris egyszerűen nyomtathatónak ítéltető. Ha az Olvasó a weboldalain fehér háttérrel és ettől elütő, sötét szöveget, valamint kevés képet használ, akkor esetleg szükségtelen a külön elkészítendő, nyomtatásra szánt változaton törnie a fejét. A sötét háttérrel készült, dinamikus hivatkozásokat és sok képet tartalmazó oldalakkal azonban könnyen meggyűlhet a baja egy átlagos nyomtatónak.

Amikor azt firtatjuk, hogy miként tehetjük nyomtatóbaráttá a weboldalainkat, a nyomtatókban használt papír jelentette korlátokat kell szem előtt tartanunk. Más szavakkal, azt kell megfogalmaznunk, hogy mi az, ami egy nyomtatott oldalt alapvetően megkülönböztet a számítógép monitorától? A nyilvánvaló különbség a méret: egy nyomtatott oldal mérete rendszerint adott: 30,5 centiméter hosszú és 21 centiméter széles.

A monitorok mérete nagyban különbözhet egymástól. A méreten felül a nyomtatott oldalak színét sem változtathatjuk kedvünk szerint – akkor sem, ha színes nyomtatónk van. Igen kevés az olyan felhasználó, aki szívesen pocskol tintát a színes háttér kinyomtatására, holott csak az oldalon lévő szöveget szeretné nyomtatásban látni.

A legtöbb felhasználó idegenkedik az oldalnak nem lényeges részét képező szöveg kinyomtatásától is. A 20.1. ábrán a kaliforniai San José és a Washington államban lévő Pullman közötti útvonalat láthatjuk példaként.



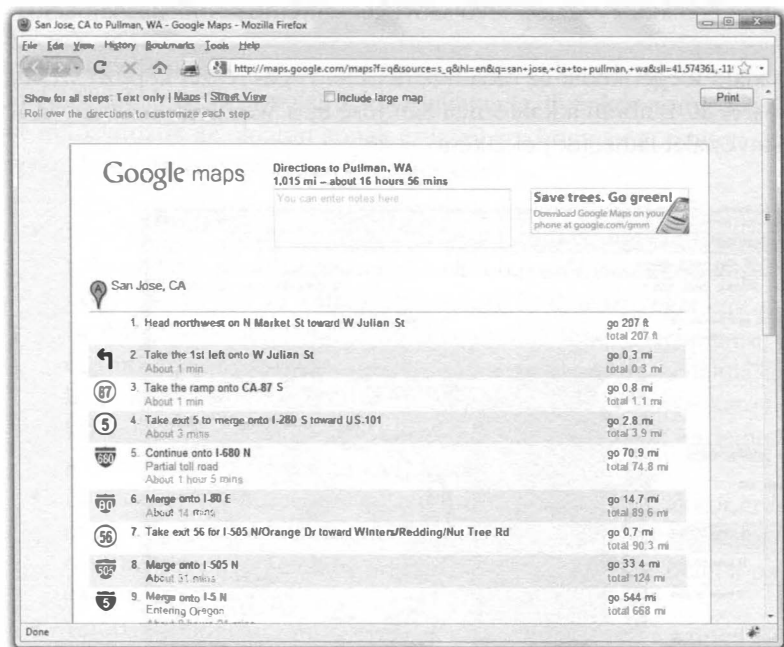
20.1. ábra

A weboldal a beviteli mezők, illetve a nagy, saját vezérlőelemekkel rendelkező kép miatt nem kifejezetten nyomtatóbarát

A 20.1 ábrán látható weboldal beviteli mezőket is tartalmaz, valamint egy nagyméretű képet, amely önállóan változtatható – mozgatható, nagyítható –, nem beszélve a weboldalon ma már megszokott segédelemről. A térkép felett kaptak helyet a végrehajtható műveletek, többek között az oldal kinyomtatását segítő hivatkozás. Lehet, hogy elgondolkodunk azon, hogy miért nem elég, ha a böngésző Nyomtatás ikonjára kattintunk. Persze, megtehetjük, de ekkor az oldalt úgy nyomtatjuk ki, ahogy

a képernyőn is látható, benne a beviteli mezőkkel és a grafikai elemekkel – holott mindössze arra vagyunk kíváncsiak, hogy vezetés közben hol kell bekanyarodnunk az úton.

A weboldal Print (Nyomtatás) hivatkozására kattintva a webböngészőnk egy olyan oldalt (20.2. ábra) jelenít meg, amelyet a Google kifejezetten az oldalt kinyomtatni kívánók igényeinek megfelelően formázott.



20.2. ábra

Az oldal nyomtatóbarát változata elkülönítve tartalmazza a vezetéshez szükséges információkat – amelyeket így önmagukban is kinyomtathatunk

Ahogy a képen látható, az oldal nyomtatóbarát változata jelentős javulást hoz az eredetihez képest – legalábbis a nyomtató szemszögéből mérlegelve a helyzetet. Minden beviteli mező és kép kimaradt.



A nyomtatásra szánt oldalakhoz egyes webtervezők a talpas betűtípusok használatát ajánlják a talp nélküli betűk helyett, ugyanis az általánosan elfogadott vélemény alapján ezek nyomtatásban könnyebben olvashatók. Ha az Olvasó a weboldalain talp nélküli betűtípust használ, neki kell eldöntenie, hogy nyomtatásban is ragaszkodik-e az oldal kinézetének megtartásához – azaz lemond-e a talpas betűk használatáról.

Annak érdekében, hogy az Olvasó minél megalapozottabb döntéseket hozhasson a nyomtatóbarát változat kialakítása során, íme egy lista azokkal a javaslatokkal, amelyeken legalább elgondolkodnunk érdemes:

- Távolítsuk el a háttérképet, így végső soron fehér hátteret kapunk nyomtatáskor.
- A szöveg színét változtassuk feketére – nem baj, ha vannak színes szövegelemek, de a fekete a javasolt szín.
- Bizonyosodjunk meg arról, hogy a betűméret kellően nagy, és nyomtatásban is olvasható marad a szöveg. Előfordulhat, hogy több méretet is ki kell próbálnunk.
- Távolítsuk el a hivatkozásokat, illetve térjünk vissza az egyszerű aláhúzásos hivatkozásokhoz. Vannak olyan tervezők, akik az aláhúzásokat szívesen megtartják: így a látogató tudja, hogy az eredeti oldal hol tartalmaz hivatkozásokat.
- Távolítsuk el valamennyi nem létfontosságú képet. Ez rendszerint az összes, az oldalon nem tartalmi elemnek számító kép eltávolítását jelenti. Ilyenek a navigációs gombok képei, a legtöbb hirdetés, és a mozgó képek.

A fenti javaslatokon túl érdemes lehet egy olyan sort is elhelyeznünk, amely az oldal szerzőjét, URL-jét és a szerzői jogi tudnivalókat tartalmazza. Mindez olyan információt jelent, amely esetleg elvész, miután a felhasználó távozik a webhelyünkről, és csak a nyomtatott változatát tartja kézben az adott oldalnak.

Lehet, hogy nem kell máris az oldalunk átírásához fogunk. A fent leírtakkal a szerzők fő célja az, hogy az Olvasónak fogalma legyen a nyomtatóbarát oldalak mibenlétéről, és így sikeresebb legyen a nyomtatásra szánt oldalak stíluslapjainak elkészítésekor.

Igen, lehet olyan stíluslapot készíteni, amelyet kizárólag a weboldal kinyomtatásakor használunk. A következő részben ezt fogjuk megtanulni.

A megjelenési formához illő stíluslap használata

A 20.1 ábrán láhattuk, amint egy hivatkozásra, illetve a kis nyomtatóikonra kattintva lehetővé válik az oldal egy különleges, nyomtatóbarát változatának megtekintése. Az efféle ikon sok híroldalon is elterjedt, és azért számít az oldalak fontos elemének, mert ha nem lenne, a felhasználók esetleg nem fáradnának az oldal cikkekhez tartozó grafikát és hirdetéseket tartalmazó változatának papír- és tintapocsékló kinyomtatásával. Bár a nyomtatót ábrázoló ikonon, illetve a hivatkozáson alapuló megközelítés magától értetődő, és remekül működik, létezik egy olyan lehetőség is, amely szükségtelemmé teszi az efféle hivatkozásokat, amikor a nyomtatóbarát változathoz kívánunk eljutni.

Az említett lehetőség egy kifejezetten nyomtatáshoz tervezett stíluslap használatát jelenti, amely akkor fejt ki a hatását, amikor a felhasználó az oldal kinyomtatása mellett dönt.

A CSS-ben megvalósították a megjelenési formához illő vagy *hordozófüggetlen stíluslapok*

elvét. Olyan stíluslapokról van szó, amelyek egy bizonyos információhordozó – monitor, nyomtató – használatát segítik. Persze a CSS nem csak az említett két hordozót támogatja. Az alábbi listában azokat az információhordozókat tüntetjük fel, amelyek használatát a CSS 2 önálló stíluslap biztosításával segíti elő.

- `all`: minden eszköz esetében használható.
- `aural`: beszédszintetizátorokhoz (a CSS 1-ben `speech` néven ismeretes).
- `braille`: a Braille-elven működő, tapogatást segítő eszközökhöz.
- `embossed`: a Braille-nyomtatók használatához.
- `handheld`: korlátozott képernyőmérettel és sávszélességgel bíró, kézben hordozott eszközök számára.
- `print`: nyomtatott anyagok, illetve a monitoron nyomtatási előnézetben megtekintett anyagokhoz.
- `projection`: kivetített bemutatókhoz.
- `screen`: számítógép-monitoron való megjelenítéshez.
- `tty`: állandó szélességű karakterekkel dolgozó eszközök (például terminálok, szöveges terminálok és korlátozott képernyőméretű kézi eszközök) számára.
- `tv`: televízió, illetve hozzá hasonló, általában alacsony felbontású és csak korlátozott mértékben görgethető képernyők számára.

Alighanem az `aural` a legérdekesebb információhordozó-típus; ezzel válik lehetővé a weboldalak felolvasása vagy egyéb módon történő meghallgatása. A CSS megalkotójának nyilvánvalóan egy lényegesen szélesebben elérhető Világháló lebegett a szemük előtt annál, amilyen kép a Webről jelenleg él bennünk – hiszen elsődlegesen számítógép-monitoron megjelenítendő weboldalakat tervezünk. Igaz, hogy a meghallgatható weboldalak tervezésével mostanság még kevés dolgunk van, a típus léte mégis hasznos figyelmeztetés lehet az egyelőre a látóhatár mögött megbúvó lehetőségeket illetően.

A más információhordozókon való használatra készített stíluslapokkal kapcsolatban azzal a jó hírrel szolgálhatunk, hogy az Olvasónak semmi újat nem kell megtanulnia. Elképzelhető ugyan, hogy a meghallgatásra szánt weboldalak esetében akad egy-két megtanulandó új trükk, de addig is elég a már megtanult stíuselemek használata a nyomtatást segítő stíluslapok elkészítéséhez. Arra kell figyelnünk, hogy miként juttassuk érvényre az adott információhordozóhoz készült stíluslapot.

Ha még emlékszünk, a `<link />` címkét használjuk arra, hogy egy weboldalon külső stíluslapra hivatkozzunk. Ez a címke támogatja a `media` nevű jellemző használatát, amellyel eddig nem találkoztunk. Ezzel a jellemzővel adhatjuk meg azt az információhordozót, amelyikre a stíluslap vonatkozik. Alapértelmezés szerint a jellemző beállított értéke az `all`, ami azt jelenti, hogy hacsak másként nem rendelkezünk, minden információhordozó esetében külső stíluslapot használunk. A többi elfogadható értéket az előző listában felsorolt információhordozók jelentik.

Ha egy weboldalhoz nyomtatóbarát stíluslapot szeretnénk megadni, két `<link />` címkét kell használnunk: az egyiket a nyomtatóhoz, a másikat a többi információhordozóhoz. A feladatot ellátó kódrészletet az alábbiakban olvashatjuk:

```
<link rel="stylesheet" type="text/css" href="standard.css" media="all" />
<link rel="stylesheet" type="text/css" href="for_print.css" media="print" />
```

Ebben a példában a weboldalunk két stíluslapra hivatkozik. Az első stíluslap esetében a `media` jellemző értékeként `all` szerepel, azaz minden információhordozóra vonatkozik. Ha nem adnánk meg másik stíluslapot, a `standard.css` stíluslap érvényesülne valamennyi információhordozó esetében. A második stíluslap jelenléte azonban azt eredményezi, hogy az oldal kinyomtatásakor a `for_print.css` stíluslapot használjuk.



Az egyes információhordozókhoz használt stíluslapokra az `@import` paranccsal is hivatkozhatunk. Az alábbi kódrészlet például a `<link />` címkék használatával azonos működést eredményez:

```
@import url(player.css) all;
@import url(player_print.css) print;
```

Egyetlen `<link />` elemben több információhordozó-típust is megadhatunk, ha a típusokat az alábbiakhoz hasonlóan vesszővel választjuk el:

```
<link rel="stylesheet" type="text/css" href="for_pp.css" media="print,
projector" />
```

A fenti kódrészlet eredményeként a `for_pp.cs` stíluslapot kizárólag a `print` és a `projector` információhordozó-típusok esetében használjuk.



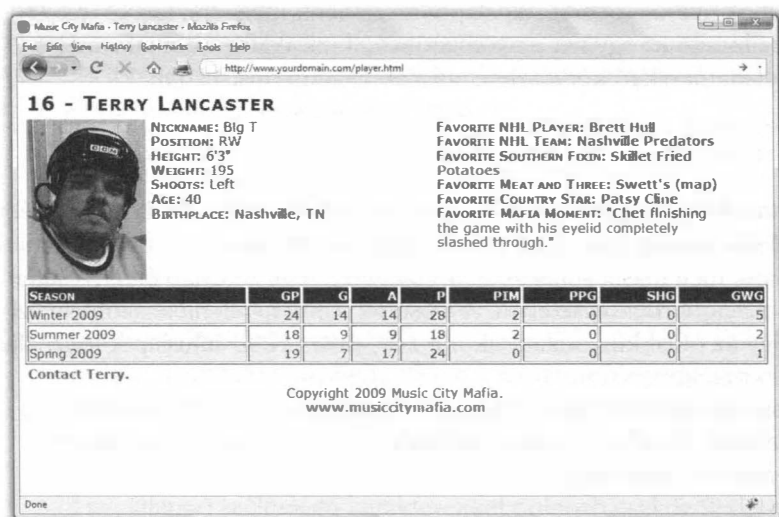
Esetleg kísértést érzünk, hogy az előző kódrészletben az elsőként hivatkozott stíluslapnál a `media` jellemző értékéül `screen`-t adjunk meg. Ha az oldalt szokásos webböngészővel nézzük meg, nem is lesz semmi baj, ha azonban a felhasználó egy kézi eszközzel vagy valamilyen más információhordozó segítségével olvassa az oldalt, a helyzet problémásabbá válik. Más szóval: a stíluslap csak a `media` jellemző értékeként felsorolt esetekben fejt ki a hatását, máskor nem.

A nyomtatható oldalakhoz való stíluslap elkészítése

A nyomtatóbarát oldalak kialakításához szükséges módosításokra vonatkozó tanácsokkal felfegyverkezve immár belevethetjük magunkat egy nyomtatóbarát stíluslap elkészítésébe. Először tekintsük meg a szokásos (monitorra érvényes) stíluslap használatával megjelenített weboldalt a 20.3. ábrán.



Az információhordozó típusát akkor is megadhatjuk, ha nem hivatkozunk külső stíluslapra. A `<style>` elem ugyanazt a `media` jellemzőt használja, mint a `<link />`.



20.3. ábra

Egy CSS használatával készült oldal szokványos böngészőben olvasva

A fenti kép megmutatja, hogy miként látszik az oldal egy szokásos webböngészőben. Az a helyzet, hogy ez az oldal nem áll túl messze attól, hogy nyomtatható legyen, de azért van még rajta mit javítani.

A weboldal az alábbi módosításokkal tehető nyomtatóbarátabbá:

- Cseréljük minden szöveg színét feketére.
- Távolítsuk el a hivatkozások félkövér és színes formázását.
- A játékoskal kapcsolatos információkat tartalmazó részeket helyezzük egymás fölé, mivel egymás mellett aligha férnek el egy nyomtatott oldalon.
- A játékoskal való kapcsolatfelvétellel szolgáló feliratot („Contact Terry”) teljes egészében távolítsuk el.



Bár az abszolút elhelyezés remekül működik a jégkorong-játékost ábrázoló mintaoldalunkon, a nyomtatásra szánt oldalak esetében nem mindig számít jó ötletnek. Kevésbé talányosan fogalmazva azt mondhatjuk, hogy ha az oldalon lévő lényegi tartalom egy oldalnál nagyobb területet igényel, érdemesebb relatív elhelyezést használnunk, és engednünk a tartalmat több oldalon át folytatódni.

Az általános célú stíluslaphoz képest megejtendő első két változtatás igazán magától értetődő – lényegében a már meglévő stíuselemek megváltoztatását, illetve megszüntetését jelentik. A harmadik változtatás mindazonáltal némi fontolgatást igényel. Annak a ténynek az ismeretében, hogy a nyomtatott oldalak mérete adott, érdemes lehet a nyomtatott változatban megjelenő valamennyi elem esetében abszolút elhelyezést

használni. Így lényegesen egyszerűbb a tartalmat hordozó elemeket pontosan a szándékaink szerint elhelyezni. Végül pedig azt említjük meg, hogy a fenti lista utolsó elemének megvalósítása igen egyszerű – egyszerűen annyi a dolgunk, hogy a `contact` elem `display` tulajdonságának a `none` értéket adjuk.

A 20.1. példa a `player_print.css` stíluslap CSS-kódját ismerteti, amely a benne található, az előzőekben ismertetett változtatásokkal tökéletesen alkalmas a jégkorong-játékosokkal foglalkozó oldalaink kinyomtatására.

20.1. példa *A jégkorong-játékosokkal foglalkozó oldalak kinyomtatását segítő CSS-kód*

```
body {
    font-family: Verdana, Arial;
    font-size: 12pt;
    color: black;
}

div {
    padding: 3px;
}

div.title {
    font-size: 18pt;
    font-weight: bold;
    font-variant: small-caps;
    letter-spacing: 2px;
    position: absolute;
    left: 0in;
    top: 0in;
}

div.image {
    position: absolute;
    left: 0in;
    top: 0.5in;
}

div.info {
    position: absolute;
    left: 1.75in;
    top: 0.5in;
}

div.favorites {
    position: absolute;
    left: 1.75in;
    top: 2in;
}

div.footer {
    position: absolute;
    text-align: left;
```

```
    left:0in;
    top:9in;
}

table.stats {
    width:100%;
    text-align:right;
    font-size:11pt;
    position:absolute;
    left:0in;
    top:3.75in;
}

div.contact {
    display:none;
}

.label {
    font-weight:bold;
    font-variant:small-caps;
}

tr.heading {
    font-variant:small-caps;
    background-color:black;
    color:white;
}

tr.light {
    background-color:white;
}

tr.dark {
    background-color:#EEEEEE;
}

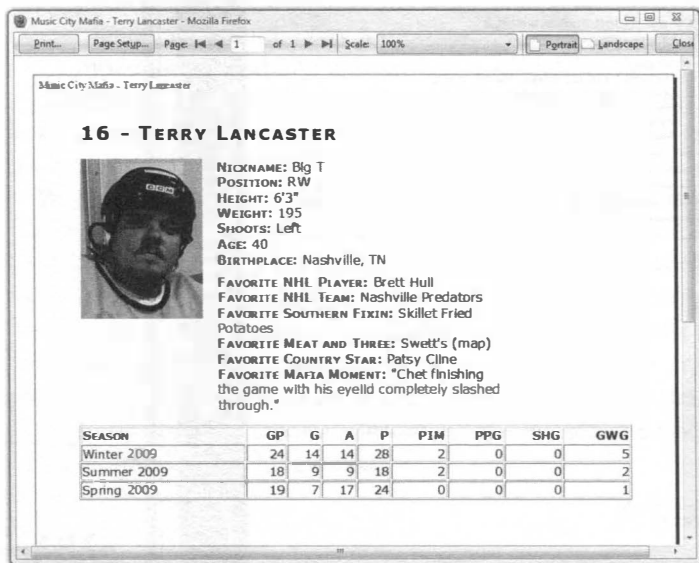
th.season, td.season {
    text-align:left;
}

a, a:link, a:visited {
    color:black;
    font-weight:normal;
    text-decoration:none;
}
```

A fenti kódban talán az a legszebb, hogy hüvelyket (angolul inch, a CSS-ben: in) használ mértékegységként minden, az abszolút elhelyezést megvalósító helyen. Ha arra gondolunk, hogy a nyomtatott oldalakat képpont helyett hüvelykben szokás mérni (mármint az USA-ban), ez igen hasznos dolog. A kódot figyelmesen tanulmányozva észrevehetjük, hogy a szöveg mindenütt fekete, a hivatkozásokról minden különleges stílusformázást eltávolítottunk, és a tartalmat képviselő részek elhelyezése most már abszolút – vagyis éppen ott lesznek, ahová szánjuk őket.

Az oldalak nyomtatási előnézetének megtekintése

A 20.4. ábra a jégkorong-játékosslal foglalkozó oldal nyomtatóbarát változatát mutatja – így látható a Mozilla Firefox Print Preview (Nyomtatási előnézet) ablakában.



20.4. ábra

A nyomtatási előnézet használatával kinyomtatás nélkül is megtekinthetjük a weboldalak nyomtatóbarát változatát

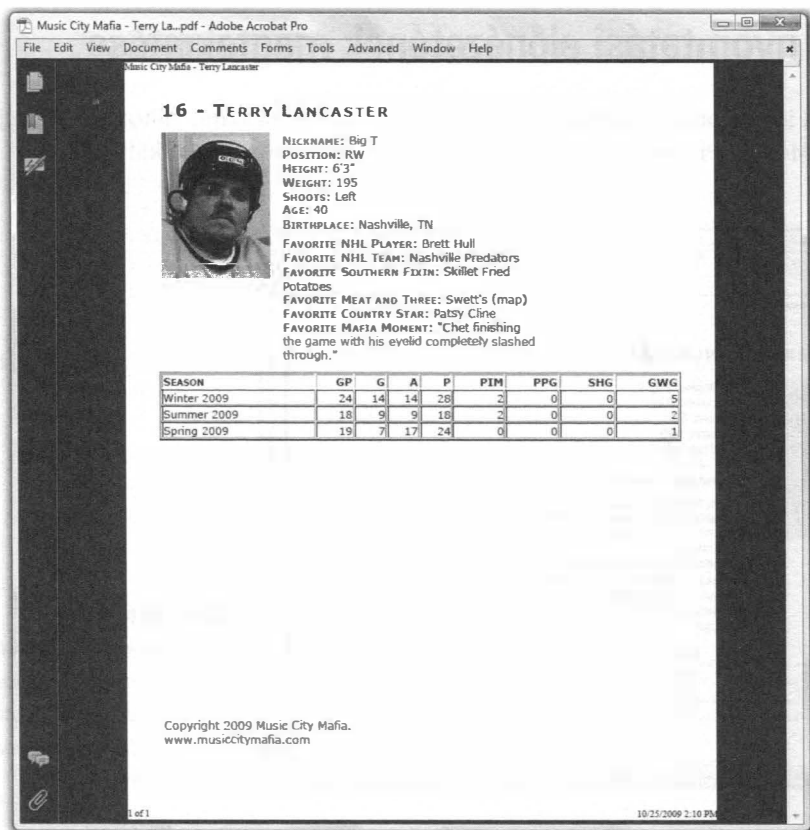
Ha a 20.4. ábra az egész oldalt megmutatná – mind a 11 hüvelyknyi magasságát, meg még egy keveset – felfigyelhetnénk rá, hogy az oldal nyomtatóbarát változata legalul mostanra a láblécet is tartalmazza – lásd a 20.5. ábrát.

A 20.5. ábra a nyomtatóbarát változatok gyakorlati használatát hivatott bemutatni. Az ábrán a jégkorong-játékost bemutató, szokásos oldalunk az Adobe Acrobat Readerrel megnyitható PDF-dokumentum formájában tűnik fel újra.

A jégkorong-játékost ábrázoló weboldalt az Adobe cég virtuális nyomtatójával „kinyomtatva” PDF-dokumentummá alakíthatjuk. Az Olvasó találkozhat olyan PDF-átalakítókkal is, mint a DoPDF (<http://www.dopdf.com>), amelyek az Adobe Acrobat alkalmazásnál alacsonyabb költséggel állíthatók üzembe. PDF-formátumba nyomtatva lényegében olyan formátumát kapjuk meg a nyomtatóbarát weboldalnak, amely kinyomtatás céljából könnyedén továbbítható elektronikusan.

Ha bővebb információra vágunk az Acrobat alkalmazást illetően, látogassunk el a <http://www.adobe.com/products/acrobat/weboldalra>.





20.5. ábra

A jégkorong-játékost bemutató oldalból úgy készítettünk PDF-dokumentumot, hogy kinyomtattuk az Adobe PDF-nyomtatójával

Összefoglalás

A mai órán a CSS-nek egy rendszeresen felmerülő igényt – a weboldalak nyomtatását – megoldó gyakorlati alkalmazásával foglalkoztunk. Az óra elején megtudtuk, hogy pontosan mit takar a nyomtatóbarát weboldal fogalma. Ezt követően megismerkedtünk a CSS-be épített, a weboldal megjelenítésére szolgáló információhordozók elkülönítését szolgáló lehetőséggel, illetve megtanultuk azt is, hogy miként választható meg a megfelelő stíluslap. Az óra végén létrehoztunk egy stíluslapot, amelynek egyetlen feladata, hogy az oldalt előkészítse nyomtatásra. Bár a legtöbb felhasználó szívesebben tekinti meg az oldalt egy nagyméretű monitoron ahelyett, hogy papírról olvasná, vannak helyzetek, amikor mindenképpen szükséges a weboldal kinyomtatása. A weboldalaink olvasóinak mindig a lehető legnagyobb rugalmasságot kell biztosítanunk, ezért a nyomtatóbarát változatok kialakítása feltétlenül szükséges.

Kérdezz-felelek

- K: A `<link />` elem *media* jellemzőjének használatával megoldható-e a kézi eszközökön való megtekintéshez szükséges stíluslap kialakítása?
- V: Igen. Ha a `<link />` elem *media* jellemzője a *handheld* értéket kapja, akkor pontosan a kézi készülékekre szánt stíluslapot adhatjuk meg. Alighanem azt fogjuk tapasztalni, hogy a mobil eszközökre szánt weboldalak is ebbe az irányba mozdulnak el, ahelyett, hogy erre a célra különleges, a WML-hez (Wireless Markup Language) hasonló jelölőnyelveket használnának.
- K: A továbbiakban is el kell helyeznem a nyomtatható változatra utaló kis nyomtatóikont az oldalaimon?
- V: Nem. A kapcsolt stíluslapok módszere, amellyel a mai órán ismerkedtünk meg, anélkül teszi lehetővé a nyomtatóbarát weboldalak kialakítását, hogy bármiféle különleges hivatkozás szerepelne az oldalon. Ugyanakkor, ha a felhasználó számára lehetővé szeretnénk tenni, hogy a nyomtatóbarát változatot a böngészőben tekintse meg, létrehozhatunk egy hivatkozást egy olyan változatra, amely fő (böngészőben megjelenő) stíluslapként is nyomtatóbarát stíluslapot használ. További lehetőség, ha az oldalon a felhasználó tudomására hozzuk, hogy a nyomtatóbarát változat megtekintéséhez a böngészőprogram nyomtatási előnézetét kell használnia.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

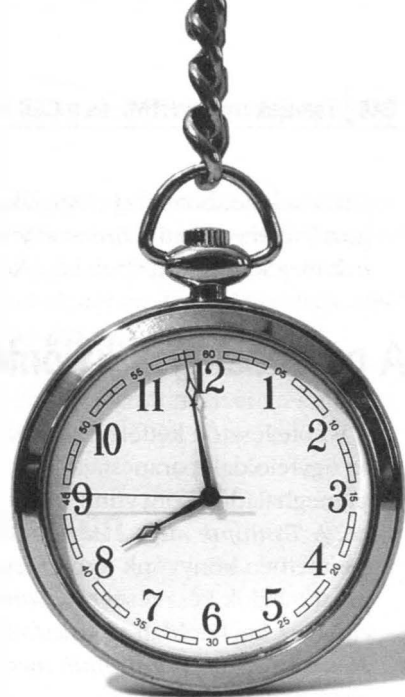
1. Ha a nyomtatóbarát változathoz vezető ikonra kattintunk, biztosan nyomtatóbarát változathoz jutunk?
2. Mi történik a külső stílussal, ha úgy hivatkozunk rá az oldalon, hogy a *media* jellemzőnek nem adunk értéket?
3. Hogyan kell hivatkoznunk a `freestyle.css` nevű stíluslapra egy weboldalon, ha azt szeretnénk, hogy a hatása csak az oldal televíziós történő megtekintésekor érvényesüljön?

Válaszok

1. Nem. Az is fontos, hogy a hivatkozás olyan különleges stíluslapot használó oldalra mutasson, amelynek hatására az oldal valóban „nyomtatóbarát” lesz.
2. A `media` alapértelmezett értéke az `all`; ennek hatására a stíluslap minden információhordozó-típus esetében érvényre jut.
3. `<link rel="stylesheet" type="text/css" href="freestyle.css" media="tv" />`

Gyakorlatok

- Hozzunk létre nyomtatóbarát stíluslapot egy olyan weboldalhoz, amelyen meglehetősen sokféle szín és sok kép található. Ne feledkezzünk meg a nyomtatóbarát stíluslapra hivatkozó `<link />` elem elhelyezéséről a weboldalon.
- Azok, akikben túlteng az önbizalom, próbálkozzanak a `<link />` elem `media` jellemzőjének `handheld` értéket adva a weboldalaikból kézi eszközökre szánt változatot kialakítani. Az elv ugyanaz, mint a nyomtatóbarát oldalak esetében, azzal a különbséggel, hogy ez esetben az olvasás nem papírlapról, hanem egy igencsak korlátozott méretű kijelzőről történik. Az oldal ellenőrzése úgy valósítható meg, hogy közzétesszük az oldalt, majd a mobiltelefonunkról vagy a kézi eszközünk böngészőjéből nyitjuk meg.



21. ÓRA

Dinamikus webhelyek

A lecke tartalma:

- A dinamikus tartalom különféle típusai
- JavaScript a HTML-kódban
- Véletlenszerű szöveg megjelenítése JavaScript használatával
- Képváltás felhasználói műveletek alapján, JavaScript segítségével

A *dinamikus* szó olyasvalamit jelent, ami mozgásban van, illetőleg mozgásra készített valamit. Webhelyekről beszélve dinamikus webhely alatt olyan webhelyet értünk, amelyet úgy terveztek, hogy a felhasználói műveletek a webhely működésének szerves részét képezik, vagyis igyekeznek a felhasználót valamilyen cselekvésre rávenni – legyen ez annyi, hogy tovább olvas egy írást, vagy hogy megvesz egy terméket, és hasonló. A mai lecke során belekóstolunk a webhelyet dinamikussá tevő felhasználói beavatkozás tárgykörébe, érintve mind a kiszolgáló, mind az ügyféloldali parancsfájlok készítését – ez utóbbiból pár gyakorlati példát is felvonultatva.

A könyv más részén is tettünk már említést az ügyféloldali parancsfájlok készítéséről, és az Olvasó is készített már ilyet a 18. óra során, amikor eseményjellemzőket és JavaScript-kódot használtunk bizonyos elemek stílusának megváltoztatására – ezt a dokumentum-objektummodell (Document Object Model, DOM) módosításának nevezzük. Ebben a fejezetben is hasonló dolgunk akad majd – ha pontosabban szeretnénk foglalmazni,

akkor a különböző technológiákkal való megismerkedés után JavaScript-kód segítségével véletlenszerű idézeteket jelenítünk meg egy oldal betöltődésekor, és a felhasználói műveletek függvényében képeket fogunk cserélni.

A parancsfájlok különféle típusai

A webfejlesztők kétféle parancsfájltípust különböztetnek meg: a kiszolgálóoldali és az ügyféloldali parancsfájlokat. Az említett két parancsfájltípus bármelyikének tárgyalása meghaladná könyvünk kereteit – ez már végső soron számítógép-programozás lenne. A *Tanuljuk meg... 24 óra alatt* sorozat két igen hasznos és népszerű kiadványa e tekintetben könyvünk természetes kiegészítőjének tekinthető: a kiszolgálóoldali parancsfájlok készítésével a *Sams Teach Yourself PHP, MySQL and Apache All-in-One* című, az ügyféloldali parancsfájlokkal pedig a *Sams Teach Yourself JavaScript in 24 Hours* (magyarul: *Tanuljuk meg a JavaScript használatát 24 óra alatt*, Kiskapu, 2006) című könyv foglalkozik.

A *kiszolgálóoldali* parancsfájlok olyan parancsfájlok, amelyek a webkiszolgálón futnak, és a futásuk eredményét a webkiszolgáló juttatja el a webböngészőhöz. Ha előfordult már olyan, hogy egy webhelyen űrlapot töltöttünk ki – ideértve bármely keresőmotor használatát –, akkor már találkoztunk is kiszolgálóoldali parancsfájl eredményeként előálló weboldallal. Az alábbi listában népszerű kiszolgálóoldali parancsfájl-értelmezőket ismertetünk – a feltüntetett webhelyeket felkeresve tudhatunk meg többet róluk.

- PHP (PHP: Hypertext Preprocessor): <http://www.php.net/>
- JSP (Java Server Pages): <http://java.sun.com/products/jsp/>
- ASP (Active Server Pages): <http://www.asp.net/>
- Perl: <http://www.perl.org/>
- Python: <http://www.python.org/>
- Ruby: <http://www.ruby-lang.org/>



Neve ellenére a JavaScript nyelv nem leszármazottja a Java nevű, objektumközpontú programozási nyelvnek, és más kapcsolatban sem áll vele. A Sun Microsystems cég által 1995-ben bevezetett Java nyelv a JSP kiszolgálóoldali parancsfájl-értelmezővel áll igen szoros rokonságban. A JavaScript nyelvet a Netscape Communications cég fejlesztette ki, szintén 1995-ben. A nyelv azért kapta ezt a nevet, mert így kívánták jelezni a külső hasonlóságát a Java nyelvvel – mindazonáltal a Java és a JavaScript nincs közvetlen rokonságban egymással.

Az előzőekkel ellentétben az *ügyféloldali parancsfájlok* olyan parancsfájlok, amelyek a webböngésző belsejében futnak – a parancsfájl futtathatóságának nem feltétele a webkiszolgálóval való kapcsolat. Az ügyféloldali parancsnyelvek közül messze a JavaScript a legnépszerűbb. Sokéves kutatások szerint a webböngészők kilencvenhárom százalékában engedélyezett a JavaScript-parancsfájlok futtatása.

A Microsoft cég által fejlesztett *VBScript (Visual Basic Scripting Edition)* egy másik ügyféloldali parancsnyelv. Ezt a nyelvet csak a Microsoft Internet Explorer webböngészővel megtekintett oldalakon használhatjuk, így aztán csak olyan esetekben érdemes ezzel a nyelvvel dolgoznunk, amikor egészen biztosak vagyunk abban, hogy a webhelyünket csak Internet Explorerrel fogják látogatni – például egy zárt céges környezetben. Minthogy szándékaink szerint a lehető legszélesebb olvasóközönséghez szeretnénk szólni, ebben a leckében az ügyféloldali parancsfájlokat JavaScript nyelven készítjük el, és az óra valamennyi példáját is ezen a nyelven írtuk.

JavaScript-kód a HTML nyelvű oldalakon

JavaScript-kód két helyen lehet a fájljainkban:

- önálló állományban, amely .js kiterjesztést kap, és
- közvetlenül a HTML nyelvű fájlok belsejében.

A külső fájlokat gyakran részesítik előnyben a parancsfájl-könyvtárak (több weboldalon is használható kódok) készítői, míg a közvetlenül a HTML-fájlba kerülő JavaScript-kódok általában az adott oldalra jellemző egyedi műveleteket valósítanak meg. Mindegy, hogy a JavaScript-kódunk hol kapott helyett, a létezéséről a `<script></script>` címkepárral kell értesítenünk a webböngészőt.

Ha a JavaScript-kódot külső fájlban helyeztük el, az alábbi módon hivatkozhatunk rá:

```
<script type="text/javascript" src="/eleresi/ut/script.js">
```

A `<script></script>` címkepár rendszerint a `<head></head>` címkék között kap helyet, mivel szigorú értelemben véve nem olyan tartalomról van szó, amelynek a `<body>` és `</body>` címkék között, azaz a szövegtörzsben kellene helyet biztosítanunk. Mindazonáltal az is elég, ha a JavaScript-függvényeket, illetve -kódrészleteket mindössze a `<script>` címkékkel határoljuk, és egyébként oda tesszük az oldalon belül, ahol éppen szükség van rájuk. A 21.1. példa egy olyan JavaScript-kódot ismertet, amelyet a HTML-dokumentum törzsében helyeztünk el.

21.1. példa Szöveg kiírása JavaScript-kóddal

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>JavaScript Example</title>
  </head>
```

```

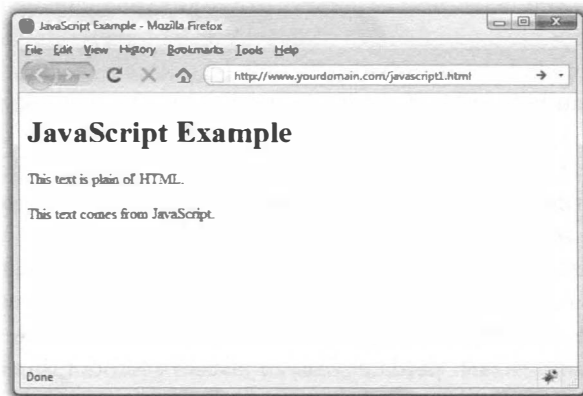
<body>
  <h1>JavaScript Example</h1>
  <p>This text is HTML.</p>
  <script type="text/javascript">
    <!-- A parancsokat elrejtjük a régi böngészők elől
    document.write('<p>This text comes from JavaScript.</p>');
    // A parancsok elrejtése eddig tartott -->
  </script>
</body>
</html>

```

A `<script>` és a `</script>` címkék közé egyetlen JavaScript-parancs került, amely az alábbi HTML-kódot állítja elő:

```
<p>This text comes from JavaScript.</p>
```

A böngészőnk a fenti HTML-oldal megjelenítése alatt a `<script></script>` címkepárhoz érve megáll, a másodperc ezredrésze alatt végrehajtja a parancsot, majd folytatja az oldal megjelenítését, immár a JavaScript-parancs kimeneteként előálló HTML-kimenetet is tartalmazva. A 21.1. ábra szerint a weboldal éppen úgy jelenik majd meg, mint bármely más HTML-oldal. Végül is HTML-oldalról van szó, annyi különbséggel, hogy a HTML-kód egy parányi részét a JavaScript-parancs állítja elő.



21.1. ábra

A JavaScript nyelvű kódrészlet kimenete semmiben sem különbözik a többitől



A 21.1. példát olvasva alighanem felfigyeltünk az alábbi két sorra:

```

<!-- A parancsokat elrejtjük a régi böngészők elől
// A parancsok elrejtése eddig tartott -->

```

Ezek valójában HTML nyelvű megjegyzések. Minden, ami a `<!--` és a `-->` karakterek közé esik, látható marad a forráskódban, de a böngésző nem jeleníti meg. Esetünkben a JavaScript-kód az, amit a HTML-kódon belül megjegyzéssé alakítottunk, arra a kevésbé valószínű esetre felkészülve, hogy a webhely látogatója túl régi böngészőt használ, vagy esetleg a böngészőjében letiltotta a JavaScript-parancsok végrehajtását.

Véletlenszerű tartalom megjelenítése

JavaScript-kóddal megoldható például az, hogy az oldal minden betöltésekor némileg eltérő tartalommal jelenjen meg. Lehet, hogy az Olvasónak is van olyan, szövegrészletekből, illetve képekből álló gyűjteménye, amelyet elég érdekesnek talál ahhoz, hogy a weboldalain is megjelenítse.

A fejezet szerzője imádja az érdekes idézeteket. Ha az Olvasó is hasonló alkat, alighanem jól elszórakozik majd azzal, hogy minden alkalommal más idézetet írhat ki a weboldalain. Ha egy minden betöltéskor más idézetet kiíró weboldalt kívánunk létrehozni, akkor az az első dolgunk, hogy összegyűjtjük az idézeteket, illetve az idézetek forrásait. Ezt követően az idézeteket elhelyezzük egy JavaScript-tömbben. A tömb a programozási nyelvek egy olyan különleges tárolóeszköze, amely remekül alkalmas adatlisták tárolására. Miután az idézeteket betöltöttük a tömbbe, az idézetek véletlenszerű kiválasztását végző JavaScript-kódot megírni már meglehetősen egyszerű. Azt a kódot, amelyik a kimenetet a HTML-oldalba illeszti, már ismerjük. A 21.2. példa az egyes betöltések alkalmával véletlenszerű idézetet megjelenítő weboldal HTML- és JavaScript-kódját egyaránt tartalmazza.

21.2. példa Véletlenszerű idézeteket megjelenítő weboldal

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Quotable Quotes</title>
    <script type="text/javascript">
      <!-- A parancsokat elrejtjük a régi böngészők elől
      function getQuote() {
        // Létrehozzuk a tömböket
        quotes = new Array(4);
        sources = new Array(4);

        // Feltöltjük a tömböket az idézetekkel
        quotes[0] = "When I was a boy of 14, my father was so " +
          "ignorant...but when I got to be 21, I was astonished " +
          "at how much he had learned in 7 years.";
        sources[0] = "Mark Twain";
        quotes[1] = "Everybody is ignorant. Only on different " +
          "subjects.";
        sources[1] = "Will Rogers";
        quotes[2] = "They say such nice things about people at " +
          "their funerals that it makes me sad that I'm going to " +
          "miss mine by just a few days.";
        sources[2] = "Garrison Keilor";
        quotes[3] = "What's another word for thesaurus?";
        sources[3] = "Steven Wright";
```

```

// Előállítunk egy véletlenszerű tömbindexet
i = Math.floor(Math.random() * quotes.length);

// Az idézetet megjelenítjük HTML nyelven
document.write("<dl style='background-color: lightpink'>\n");
document.write("<dt>" + "\"<em>" + quotes[i] + "</em>\n\n");
document.write("<dd>" + "- " + sources[i] + "\n");
document.write("<dl>\n");
}
// A parancsok elrejtése eddig tartott -->
</script>
</head>

<body>
<h1>Quotable Quotes</h1>
<p>Following is a random quotable quote. To see a new quote just
reload this page.</p>
<script type="text/javascript">
  <!-- A parancsokat elrejtjük a régi böngészők előtt
  getQuote();
  // A parancsok elrejtése eddig tartott -->
</script>
</body>
</html>

```

Bár a fenti kód némiképp hosszúnak tűnik, figyelmes tanulmányozás közben megfigyelhetjük, hogy a nagy részét a weboldalon megjeleníthető négy idézet alkotja. Ha ezen túltesszük magunkat, a kód maga már nem olyan vészesen bonyolult. Az első `<script></script>` címkepár között lévő sok-sok sor a `getQuote()` nevű függvény létrehozását végzi. Miután a függvényt megírjuk, az a tartalmazó oldal más részéről is meghívható lesz. Felhívjuk a figyelmet arra, hogy ha a függvényt egy külső fájlban helyeznénk el, akkor azt bármely oldalunkról meghívhatnánk. A kód figyelmes tanulmányozása során az alábbiakhoz hasonló sorokat veszünk észre:

```

// Létrehozzuk a tömböket
és
// Feltöltjük a tömböket az idézetekkel

```

Nos, ezek megjegyzések. A kód írója ilyen formában helyezhet el megjegyzéseket a kódban. A megjegyzések arra valók, hogy a kódot olvasók is megértsék, hogy a kód egy-egy része miért felelős. Az első, a tömbök létrehozásáról szóló megjegyzés után látható, amint létrejön két, egyenként négy elemet tartalmazó tömb – az egyik neve `quotes` (idézetek), a másiké pedig `sources` (források):

```

quotes = new Array(4);
sources = new Array(4);

```

A második megjegyzés (amely a tömbök idézetekkel való feltöltéséről szól) után négy elemet helyezünk el a tömbjeinkben. Az első, Mark Twaintől származó idézetet közelebbről is szemügyre vesszük:

```
quotes[0] = "When I was a boy of 14, my father was so " +
"ignorant...but when I got to be 21, I was astonished at " +
"how much he had learned in 7 years.";
sources[0] = "Mark Twain";
```

Azt már tudjuk, hogy a `quotes` és a `sources` szavak a tömbök nevei. Azokat a változókat azonban, amelyeknek értéket adunk, esetünkben a `quotes[0]` és a `sources[0]` névvel illetjük. Az idézeteket és a forrásokat tömbök tartalmazzák, és minden tömb-elemnek száma is van. A tömbök első elemei nem az 1., hanem a 0. helyre kerülnek. Más szóval, a számozást nem az 1, hanem a 0 sorszámnál kezdjük. Az első idézet szövegét (azaz az értéket) tehát a `quotes[0]` nevű változóban helyezzük el. Ehhez hasonlóan az első forrás szövege a `source[0]` változóba kerül. A karakterláncokat idézőjelbe tesszük. Minthogy azonban a JavaScript nyelvben a parancsok végét sortörés jelzi, az alábbi kódrészlet problémákat okozna:

```
quotes[0] = "When I was a boy of 14, my father was so
ignorant...but when I got to be 21, I was astonished at
how much he had learned in 7 years. ";
```

Így aztán azt látjuk, hogy a karakterláncot más, idézőjelbe tett karakterláncokból fűzzük össze, az összefűzés jelzésére a pluszjelet (+) használva. A következő kódrészlet az, amelyik a leghatározottabban emlékeztet programozásra. A sor egy véletlenszámot állít elő:

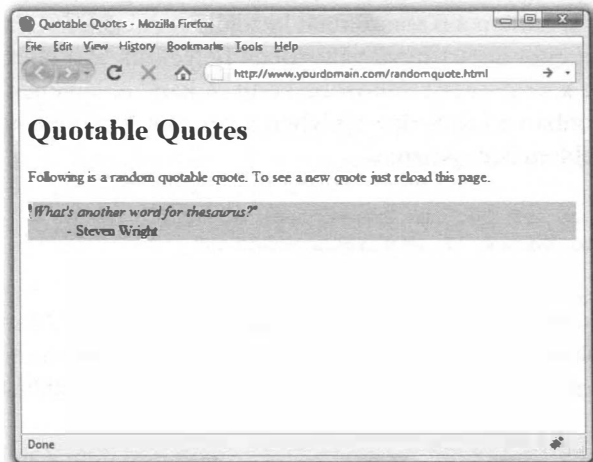
```
i = Math.floor(Math.random() * quotes.length);
```

A céljainknak azonban nem lesz megfelelő akármelyik véletlenszám, mivel a véletlenszám annak megállapítására kell, hogy melyik idézetet, illetve forrást írjuk ki – és nekünk csak négy idézetünk van. Így aztán a fenti JavaScript-sor:

- a `Math.random()` függvénnyel előállít egy 0 és 1 közötti véletlenszámot; a `Math.random()` függvény egy lehetséges kimenete lehet például a 0.5482749;
- a véletlenszámot megszorozza a `quotes` tömb hosszával, ami jelenleg 4 (a tömb hossza alatt a tömb elemszámát értjük); ha a véletlenszám az előzőeknek megfelelően 0.5482749, akkor néggyel megszorozva 2.1930996-ot kapunk;
- a `Math.floor()` függvénnyel az eredményt lefelé kerekíti a legközelebbi egész számra, más szóval a 2.1930996 számból 2 lesz;
- az `i` változó értékéül a 2 számot adja.

A függvény további része néhány kivételtől eltekintve elvileg mostanra ismerős. Ahogy azt már az óra korábbi részében láttuk, a `document.write()` parancs arra való, hogy a böngésző által megjeleníthető HTML-kódot állítsunk elő. A karakterláncokat úgy daraboljuk, hogy nyilvánvaló legyen, hogy mit kell másként kezelni – gondolunk itt arra például, hogy amikor az idézőjeleket meg kell jeleníteni, akkor egy fordított perjellel levédjük őket (`\`), de említhetnénk a változók értékének behelyettesítését is. A ténylegesen megjelenített idézet és forrás az lesz, amelyik a `quotes[i]` és a `sources[i]` változókra illeszkedik – az `i` értékét a fenti matematikai függvényekkel határozzuk meg. Mindazonáltal pusztán az, hogy a függvényt elkészítjük, még nem jelenti azt, hogy bár-

miféle kimenetet is kapunk. A HTML-fájl későbbi részében egy `<script></script>` címkepár között ismét láthatjuk a `getQuote()` szót – ott történik a függvény hívása. Minden olyan helyen, ahol függvényhívás történik, a HTML-kódba bekerül a hívás eredménye. Esetünkben ez az eredmény egy idézetet tartalmazó bekezdés. A 21.2 ábra a Quotable Quotes (Idéznivaló idézetek) című oldalt mutatja, egy webböngészőben megjelenítve. Az oldal újratöltődésekor egy a négyhez eséllyel egy másik idézet jelenik meg – végső soron tehát a véletlen dönt.



21.2. ábra

A Quotable Quotes című oldal minden betöltődésekor egy véletlenszerűen kiválasztott idézetet jelenít meg

Ne feledjük, hogy az oldalt könnyedén módosíthatjuk úgy, hogy saját idézeteket, vagy más, véletlenszerűen megjeleníthető szöveget jelenítsen meg. A kód `quotes` és `sources` tömbjeit további értékekkel feltöltve a megjeleníthető idézetek számát is növelhetjük.

A Quotable Quotes oldalt kiindulási alapként használva a parancsfájl módosításával könnyen kialakíthatjuk a saját érdekes, a megvalósított ötletre alapuló változatunkat. Ha közben hibázunk, annyi baj legyen. A hibákat akkor tudjuk kijavítani, ha türelmesek vagyunk, és figyelmesen elemezzük a begépelt kódot. Mindig törölhetünk egy keveset a kódból, amíg annyira le nem egyszerűsítjük, hogy már működőképes legyen – ezt követően pedig kis részletenként tovább bővítve a kódot ellenőrizhetjük, hogy az újonnan beírt részlet működik-e.

A dokumentum-objektummodell

Ha a JavaScript használatával szeretnénk a weboldalainkon megvalósítani a felhasználói beavatkozás lehetőségét, akkor rendszerint valamilyen értelemben a dokumentum-objektummodell (Document Object Model, DOM) kezelésével is foglalkoznunk kell. A DOM a dokumentumok láthatatlan felépítését jelenti – nem a HTML-felépítést, és nem is azt a módot, ahogy a formázás szintjeit használjuk, hanem inkább valamiféle általánosabb érvényű keretrendszert, illetve tárolót. Ha a fenti leírás ködösnek tűnik, annak az az egyetlen oka, hogy valóban az – a tárgya nem határolható be egyszerűen.

A mindent magába foglaló tárolóobjektum neve `document`. A dokumentumban előforduló, azonosítóval rendelkező további tárolókra az azonosítójuk alapján hivatkozunk. Ha van például egy olyan, `<div>` címkével megadott szakaszunk, amelynek az azonosítója `wrapper`, akkor a megfelelő DOM-elemre a `document.wrapper` hivatkozással utalhatunk.

A 17. órán egy adott elem láthatóságát úgy változtattuk meg, hogy a hozzá kapcsolódó `style` objektumon változtattunk valamit. Ha a `wrapper` azonosítójú, `<div>` címkével megadott szakasz háttérszínét szeretnénk elérni, akkor így kell rá hivatkoznunk:

```
document.wrapper.style.background-color
```

Ha a fenti stíluselem értékét szeretnénk – akár egy felhasználói művelet eredményeként létrejövő esemény alapján – megváltoztatni, akkor az alábbi sorral cserélhetjük például fehérre a színt:

```
document.wrapper.style.background-color="#ffffff"
```

A DOM az a keretrendszer, amely az elemek és a hozzájuk tartozó objektumok elérését biztosítja a számunkra. Itt nyilvánvalóan csak rövid áttekintést nyújthatunk egy a fentiekhez hasonlóan bonyolult dologról, de most már legalább kezdjük érteni, hogy ez a `document`. *akármilyen* dolog miről is szól. A DOM-ról lényegesen többet tudhatunk meg, ha felkeressük a World Wide Web Consortiumnak a témával kapcsolatos webhelyét a <http://www.w3.org/DOM/> címen.

Képváltás felhasználói műveletek alapján

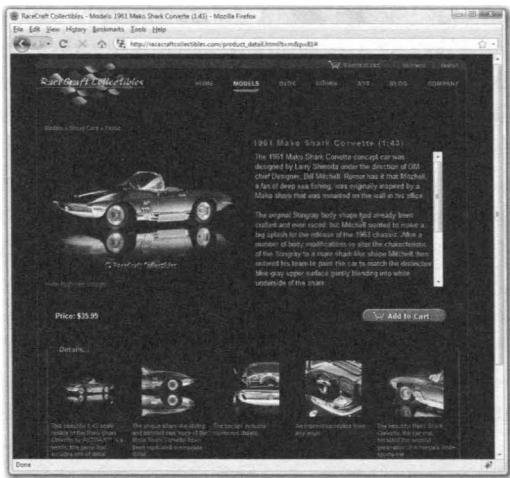
A 18. órán bemutattuk a felhasználói műveletek eseményeit: az `onclick`, az `onmouseover`, az `onmouseout` eseményt és társait. Akkor a felhasználói beavatkozásra reagálva a szövegen változtattunk; ma olyan módosításokkal ismerkedünk meg, amelyek látványosak, dinamikusak, és gyakorlati hasznuk is van.

A 21.3. ábrán egy gyűjtőknek szóló internetes katalógus oldalát látjuk. A katalógus valamennyi oldalán találunk egy nagy képet, a képen látható tárggyal kapcsolatos információkat, az oldal alján pedig további, kis képek jelennek meg. Az ilyen katalógusoknál az egyes tárgyak közeli képei is fontosak a lehetséges vásárló számára, de ha az oldalon helyeznénk el nagyobb képeket, az mind az oldal külalakjára, mind a sávszélességgel való gazdálkodásra hátrányosan hatna. Az oldalon lévő nagy képet az alábbi `` címke használatával töltjük be:

```

```

Ahogy látjuk, a kép a `product_img` nevet kapja, így aztán a DOM-ban a `document.product_img` hivatkozással érhetjük el. Ez azért fontos, mert a JavaScript segítségével dinamikusan megváltoztathatjuk a `document.product_img.src` értékét – vagyis a kép forrását, amit egyébként az `src` jellemző értékeként adunk meg.



21.3. ábra
A katalógusban lévő tárgyat tartalmazó oldal a betöltést követően

Az alábbi kód felelős a 21.3. ábra alján látható, öt képből álló csoport negyedik képének létrehozásáért. Az onmouseover esemény azt jelzi, hogy amennyiben a felhasználó az egérmutatót a kis kép fölé mozgatja, a `document.product_img.src` objektum – azaz a nagy kép forrásfájlja – értéke a megfelelő nagy képhez vezető elérési útra cserélődik.

```
<a href="#" onmouseover="javascript:document.product_img.src =  
'/path/to/large4.jpg'"></a>
```

A 21.4. ábrán is az előző oldalt látjuk – amit a felhasználó nem töltött be újra –, de itt a nagy kép helyét már egy másik képfájl tartalma tölti ki, ugyanis a felhasználó az egérmutatót az alul lévő kis képek egyike fölé mozgatta. Amikor a felhasználó az egérmutatóval az oldalon látható kis kép fölé ér, a kép nagy méretű változata megjelenik az oldal felső részében.



21.4. ábra
Amikor a felhasználó a kis kép fölé viszi az egérmutatót, a nagy képet lecseréljük

Összefoglalás

A mai órán megismertük a kiszolgálóoldali és az ügyféloldali parancsfájlok közti különbséget, és megtanultuk, hogy miként illeszthetünk JavaScript-parancsokat a HTML-fájljainkba, hogy a webhelyünkön felhasználói műveletekre adjunk lehetőséget. Tudjuk már azt is, hogy a JavaScript nyelv `document.write()` parancsa miként használható véletlenszerűen kiválasztott idézetek megjelenítésére a weboldal betöltésekor. Végül pedig kiderült, hogy mi is az a dokumentum-objektummodell, és hogy mire használhatjuk.

Az előző órákon megszerzett tudásunkat alkalmazva megtanultuk, hogy egy ügyféloldali parancsfájl segítségével miként reagálhatnak a weboldalon lévő képek az egér mozgására. A fenti feladatok egyike sem igényel különösebb programozási tudást, ugyanakkor arra sarkallhatják az Olvasót, hogy elmélyítse a tudását a JavaScript vagy a kiszolgálóoldali parancsnyelvek valamelyikének használatában, így téve alkalmassá a weboldalait a felhasználói műveletek összetettebb kezelésére.

Kérdezz-felelek

- K: *Ha az órán megismert, véletlenszerű idézeteket megjelenítő parancsfájl szerelném használni, de sok-sok idézettel szerelném feltölteni, akkor minden egyes idézetet be kell írnom minden egyes weboldal kódjába?*
- V: Igen. A tömb minden elemének szerepelnie kell az oldalon. Olyan pontra jutottunk itt, amikor az Olvasónak döntenie kell, hogy az ügyféloldali parancsfájl a megfelelő eszköz, vagy okosabb lenne a problémát a kiszolgáló oldalán megoldani. Ha valóban sok véletlen idézetünk van, és egy alkalommal mindig csak egyet szerelnénk megjeleníteni, alighanem az a legokosabb, ha az idézeteket egy adatbázis-táblában tároljuk, és egy nyúlfarknyi kiszolgálóoldali parancsfájllal kapcsolódunk az adatbázishoz, hogy kiolvassuk a szöveget, és beillesszük az oldalra. Megoldás lehet az is, ha továbbra is JavaScript-parancsok tárolják valamennyi idézetet, de ilyenkor az a legkevesebb, hogy a JavaScript-függvényt önálló fájlban helyezzük el, ugyanis így a szövegtől függetlenül történhet a karbantartása.
- K: *Láttam olyan internetes katalógusokat, amelyek a nagy képet valahogy úgy jelenítik meg, mintha egy réteg kerülne a webhely tartalma elé – továbbra is látható a webhely a kép alatt, de maga a nagy kép van az előtérben. Hogy lehet ilyet készíteni?*
- V: Így elmesélve valószínűnek tűnik, hogy a hatást a „Lightbox” nevű JavaScript-könyvtár segítségével idézték elő. Egy igen népszerű programkönyvtárról van szó, amelyet arra használhatunk, hogy a fontosnak ítélt nagy képek részleteit vagy akár egy képsorozatot mintegy „kirakatba tegyünk” az oldal tartalma elé. A programkönyvtár ingyenesen hozzáférhető a fejlesztő weboldalán, a <http://www.huddletogogether.com/projects/lightbox/> címen. A könyvtár telepítése és használata során kövessük a hozzá kapott utasításokat. A könyvünkben eddig megszerzett tudása segítségével az Olvasó képes lesz a saját webhelyén munkára fogni a programkönyvtárat.

Ismétlés

Ismétlő kérdések

1. Tegyük fel, hogy elkészítettünk egy `button.gif` nevű fájlt, benne egy gombot ábrázoló képpel. Készítettünk a képből egy egyszerű GIF-animációt is, amelyben a gomb a zöld és a fehér színt váltogatja. Ennek a fájlnak a `flashing.gif` nevet adtuk. Írjuk meg azt a HTML- és JavaScript-kódot, amely részint villogtatni kezdi a gombot, ha a felhasználó fölé viszi az egérmutatót, részint egy a gombra történő egérgattintással megnyitható, a `gohere.html` fájlra mutató hivatkozást is tartalmaz!
2. Hogyan módosítható úgy az előző feladatra adott megoldás, hogy a gomb villogni kezdjen, ha a felhasználó fölé viszi az egérmutatót, de az egérmutató elmozdulása után is folytassa a villogást?
3. Az alábbi kódrészletben mi a pluszjel szerepe?

```
document.write('This is a text string ' + 'that I have created.');
```

Válaszok

1. A kód nagyjából így nézne ki:

```
<a href="gohere.html"
onmouseover="javascript:document.flasher.src='flashing.gif'">
onmouseout="javascript:document.flasher.src='button.gif'">
</a>
```

2. A kód nagyjából így nézne ki:

```
<a href="gohere.html"
onmouseover="javascript:document.flasher.src='flashing.gif'">
</a>
```

3. A pluszjel (+) feladata a két karakterlánc összefűzése.

Gyakorlatok

- Van olyan weboldalunk, amely szebb lenne, vagy könnyebb lenne eligazodni rajta, ha a föléjük érő egérmutató hatására az ikonok, illetve egyéb képek megváltoznának? Ha igen, készítsük el a képek némileg fényesebb változatait, és módosítsuk az oldalt az ezen az órán tanultak alapján.
- Ahogy az órán tanult módon a JavaScript segítségével véletlenszerű szöveget írhatunk egy weboldalra, ugyanúgy véletlen képeket – grafikát tartalmazó sávokat, hirdetések – is megjeleníthetünk. Készítsünk el egy véletlen képeket megjelenítő weboldalt, ahol a szöveg helyére a megfelelő képeket megjelenítő `` címkét illesztjük be.

22. ÓRA



Webes űrlapok

A lecke tartalma:

- A HTML-űrlapok működése
- A HTML-űrlapok felületének létrehozása
- Az űrlapok adatalemeinek elnevezése
- Rejtett adatok elhelyezése az űrlapokon
- Hogyan választhatjuk ki a helyzetnek megfelelő beviteli vezérlőket?
- Az űrlapadatok elküldésének módja

Mindeddig javarészt azzal foglalkoztunk, hogy miként adhatunk át adatokat másoknak – a weblapjainkat azonban arra is használhatjuk, hogy magunk gyűjtsünk adatokat a látogatóinktól.

A webes űrlapok lehetővé teszik, hogy visszajelzéseket, megrendeléseket vagy más természetű adatokat fogadjunk a látogatóinktól. Ha használtunk már valamilyen keresőmotort, mint a Google, a Yahoo! vagy a Bing, máris közelebbi kapcsolatba kerültünk a HTML-űrlapokkal – ezek olyan, egyetlen beviteli mezővel rendelkező űrlapok, amelyeknek a gombját lenyomva hozzájuthatunk a keresett (és keresetlen) adatokhoz. A termékek megrendelésére szolgáló űrlapok szintén igen elterjedtek, így ha már vásároltunk valamit az Amazon.com-ról vagy az eBay-ről, szintén találkozunk kellett

űrlapokkal. Ezen az órán megtanuljuk, hogyan készítsünk saját űrlapokat – fontos azonban megjegyeznünk, hogy csak az űrlapok előfelületének jellemzőire térünk ki. A háttérfelület kiépítéséhez komolyabb programozási ismeretekre van szükség, amelyeknek a bemutatása messze meghaladná könyvünk kereteit.

A HTML-űrlap a weboldalunk része, amelynek a mezőiben a látogatók adatokat rögzíthetnek – ezek azután visszakerülhetnek hozzánk, eljuthatnak az általunk megadott elektronikus levélcímrre, egy általunk kezelt adatbázisba, vagy egy teljesen különálló rendszerbe: ilyen űrlapkezelő rendszer például a Salesforce.com.

A HTML-űrlapok működése

Mielőtt megismerkednénk azokkal a HTML-elemekkel, amelyek lehetővé teszik a saját űrlapjaink elkészítését, tekintsük át legalább fogalmi szinten, hogyan is jutnak vissza hozzánk az űrlapokon bevitt adatok. A háttérben működő folyamat (*kiszolgálóoldal*, illetve *háttérprogram*) kezeléséhez szükség van valamilyen programozási nyelv ismeretére, vagy legalábbis arra, hogy képesek legyünk a mások által erre a célra írt kiszolgálóoldali parancskód értő használatára. Ha a weboldalunk készítésében eljutunk erre a pontra, feltétlenül igénybe kell vennünk egy hozzáértő segítségét, egyébként nem marad más, mint megtanulni az alapokat. Az űrlapok egyszerű feldolgozása nem különösebben bonyolult feladat, ráadásul a webes tárhelyszolgáltatónk jó eséllyel számos kiszolgálóoldali parancsfájlt bocsát a rendelkezésünkre, amelyeket némi testreszabás után munkába állíthatunk.



A PHP a kiszolgálóoldali programnyelvek legnépszerűbbike, így minden valamit is érő webes tárhelyszolgáltató támogatja. A használatáról nagyszerű tájékoztatást kaphatunk a <http://www.php.net/> címen, de komolyabb tanulmányokat is kezdetünk az alapoktól indulva, ha beszerezzük a *Sams Teach Yourself PHP, Apache, and MySQL All-in-One* című könyvet (amelyben egyúttal az adatbázis-kezelésről is olvashatunk). Jóllehet a PHP-ről és a hozzá kapcsolódó egyéb módszerekről számos más könyvben olvashatunk, én mégis ezt ajánlanám, talán azon egyszerű okból kifolyólag, hogy én írtam. A szöveg kifejezetten azoknak szól, akik most találkoznak először a PHP-vel, és egyáltalán, bármilyen programozási nyelvvél.



Technikai értelemben módunk van arra, hogy kiszolgálóoldali parancsfájl nélkül küldjünk el űrlapadatokat – ehhez egy `mailto` hivatkozást kell használnunk a `<form>` elem `action` jellemzőjében –, ez a módszer azonban változó eredményt ad. Az egyes böngészők és az egyéni biztonsági beállítások hatására a művelet nem a várt módon mehet végbe, ami igencsak összezavarhatja a felhasználót. Amikor a felhasználó elküldi egy űrlap adatait, azt várja, hogy ez elindít egy parancsfájlt, amely a háttérben, láthatatlanul elvégzi a munkát, majd egy üzenettel tér vissza, amelyben jelzi, hogy elkészült. Nos, a `mailto` használatánál egyáltalán nem ez történik.

Az űrlapok tartalmazznak egy gombot, amellyel a felhasználó elküldheti az adatokat – a gomb lehet egy magunk választotta kép, de használhatjuk a szabványos HTML-űrlapgombot is, amelyet akkor kapunk, ha létrehozunk az űrlapon egy `<input>` elemet, és a `type` jellemzőjének a `submit` értéket adjuk. Ha a látogató a „Küldés” gombra kattint, az űrlapon rögzített adatokat a rendszer a `<form>` elem `action` jellemzőjében megadott URL-re továbbítja. Ezen a címen található az a parancsfájl, amelyik feldolgozza az adatokat – elküldheti azokat egy elektronikus levélben, de az is megeshet, hogy csak egy újabb lépést hajt végre a felhasználót kiszolgáló folyamatban (például meghív egy keresőprogramot, vagy elemeket helyez egy webáruház bevásárlókocsijába).

Ha többre vágyunk annál, minthogy elektronikus levélben elküldjük magunknak az űrlap adatait, további technikai ismeretekre lesz szükségünk. Így ha például egy webáruházat szeretnénk üzemeltetni, amely hitelkártyákat fogad el, és tranzakciókat bonyolít le, tudnunk kell, hogy erre a célra jól bevált megoldások állnak rendelkezésre, amelyek mind a felhasználó adatainak biztonságát hivatottak szolgálni. Nos, erre a harctérre nem érdemes felkészületlenül belépni, a könyvünk azonban sajnos nem biztosítja a megfelelő gyverzetet.

Mielőtt közzétennénk az űrlapunkat, fontos, hogy átnézzük a webtárhely-szolgáltatónk felhasználói útmutatóját, és tájékozódjunk arról, hogy milyen űrlapfeldolgozó parancsfájlok állnak rendelkezésre. Érdemes a *CGI* kulcsszót keresnünk (Common Gateway Interface – közös átjárófelület). A programok (így az űrlapokat kezelő parancsfájlok) igen gyakran a CGI révén lépnek kapcsolatba a webkiszolgálóval.

Űrlap létrehozása

Az űrlapok kóda minden esetben a `<form>` elemmel kezdődik, amely bárhol állhat a HTML-dokumentum törzsében. A `<form>` elem általában két jellemzővel rendelkezik, amelyeknek a neve `method` és `action`:

```
<form method="post" action="mailto:me@mysite.com">
```

A `method` a leggyakrabban a `post` értéket kapja – ilyenkor az adatokat egy dokumentum formájában küldjük el. Egyes esetekben szükség lehet a `method="get"` használatára, amikor is az adatok az URL részeként utaznak. A `get` használatára jó példát adnak a keresők webes űrlapjai. Mivel jelenleg még nem vagyunk igazán járatosak az űrlapok világában, jobb, ha a `post` beállítást használjuk, hacsak a webgazdánk útmutatója máshogy nem rendelkezik.

Az `action` jellemző azt a címet adja meg, ahová az űrlap adatait küldeni szeretnénk. Itt két lehetőségünk van:

- Megadhatjuk egy űrlapfeldolgozó program vagy parancsfájlt helyét egy webkiszolgálón, és az űrlap adatait a böngésző oda küldi el.
- A `mailto:` kulcsszó után megadhatjuk az elektronikus levélcímünket, így ha a látogató kitölti az űrlapot, az adatok közvetlenül hozzánk érkeznek meg. Ennek a megoldásnak a sikere azonban teljes mértékben azon múlik, hogy a felhasználó megfelelőképpen állította-e be a levelezőprogramját. Azok, akik olyan nyilvános számítógépről érik el a webhelyünket, amelyen nem használhatnak levelezőprogramot, zsákutcába jutnak.

A 22.1. példában létrehozott űrlapot a 22.1. ábra mutatja – itt megtalálhatunk szinte minden olyan beviteli elemet, ami a HTML-űrlapokon megjelenhet. Amikor bemutatjuk az egyes típusokat, érdemes visszatérnünk ide, hogy tisztábban lássunk.

22.1. példa *Űrlap különféle beviteli elemekkel*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Guest Book</title>

    <style type="text/css">
      .formlabel {
        font-weight:bold;
        width: 250px;
        margin-bottom: 12px;
        float: left;
        text-align: left;
        clear: left;
      }
      .formfield {
        font-weight:normal;
        margin-bottom: 12px;
        float: left;
        text-align: left;
      }
      input, textarea, select {
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <h1>My Guest Book</h1>
    <p>Please sign my guest book. Thanks!</p>
```

```

<form method="post" action="URL_to_script">

<div class="formlabel">What is your name?</div>
<div class="formfield"><input type="text" name="name"
    size="50" /></div>

<div class="formlabel">What is your e-mail address?</div>
<div class="formfield"><input type="text" name="email"
    size="50" /></div>

<div class="formlabel">Please check all that apply:</div>
<div class="formfield">
    <input type="checkbox" name="website_response[]" value="I
    really like your Web site." />I really like your Web site.<br />
    <input type="checkbox" name="website_response[]" value="One
    of the best sites I've seen." />One of the best sites I've
    seen.<br />
    <input type="checkbox" name="website_response[]" value="I sure
    wish my site looked as good as yours." />I sure wish my site
    looked as good as yours.<br />
    <input type="checkbox" name="website_response[]" value="I have
    no taste and I'm pretty dense, so your site didn't do much for
    me." />I have no taste and I'm pretty dense, so your site
    didn't do much for me.<br />
</div>

<div class="formlabel">Choose the one thing you love best about my
web site:</div>
<div class="formfield">
    <input type="radio" name="lovebest" value="me" />That gorgeous
    picture of you.<br />
    <input type="radio" name="lovebest" value="cats" />All the
    beautiful pictures of your cats.<br />
    <input type="radio" name="lovebest" value="childhood" />The
    inspiring recap of your suburban childhood.<br />
    <input type="radio" name="lovebest" value="treasures" />The
    detailed list of all your Elvis memorabilia.<br />
</div>

<div class="formlabel">If my web site were a book, how many copies
would it sell?</div>
<div class="formfield">
    <select size="3" name="sales">
        <option value="Millions, for sure." selected="selected">Millions,
            for sure.</option>
        <option value="100,000+ (would be Oprah's favorite)">100,000+
            (would be Oprah's favorite)</option>
        <option value="Thousands (an under-appreciated classic)">Thousands
            (an under-appreciated classic)</option>
        <option value="Very few: not banal enough for today's public">Very
            few: not banal enough for today's public.</option>
        <option value="Sell? None. Everyone will download it for
            ➡ free.">Sell?
            None. Everyone will download it for free.</option>
    </select>

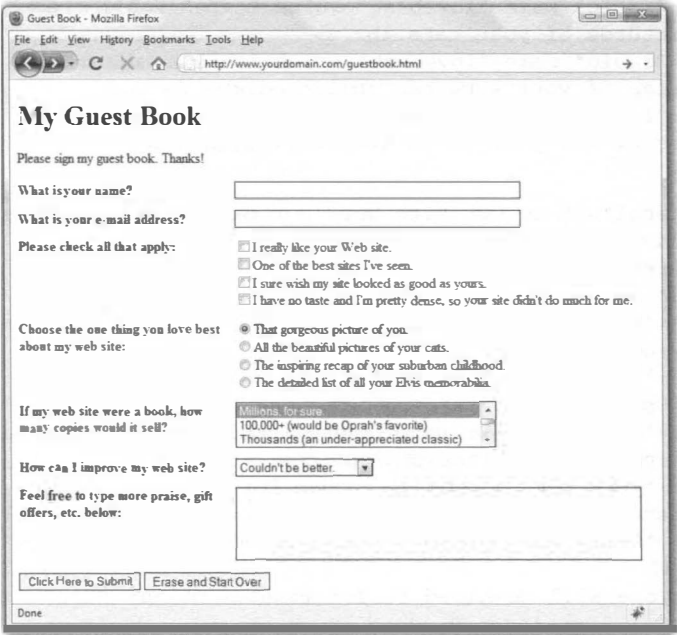
```

```
</div>

<div class="formlabel">How can I improve my web site?</div>
<div class="formfield">
  <select name="suggestion">
    <option value="Couldn't be better." selected="selected">Couldn't
      be better.</option>
    <option value="More about the cats.">More about the cats.</option>
    <option value="More about the family.">More about the
      ➡ family.</option>
    <option value="More about Elvis.">More about Elvis.</option>
  </select>
</div>

<div class="formlabel">Feel free to type more praise,
➡ gift offers, etc.
  below:</div>
<div class="formfield">
  <textarea name="comments" rows="4" cols="55"></textarea>
</div>

<div style="float:left;">
  <input type="submit" value="Click Here to Submit" />
  <input type="reset" value="Erase and Start Over" />
</div>
</form>
</body>
</html>
```



22.1. ábra
A 22.1. példában látható kód bemutatja az űrlapok szinte minden lehetséges beviteli elemét

A 22.1. példa `<form>` elemében számos `<input />` mezőt találhatunk, amelyek mindegyike egy-egy beviteli elemhez (például jelölőnégyzethez vagy választógombhoz) tartozik. Az `input`, `select` és `textarea` elemekhez a stíluslapon keret (szegély) is tartozik, így könnyen nyomon követhetjük a körvonalaikat az űrlapon belül. Ne feledjük, hogy ezekre az elemekre tetszőleges CSS-stílusokat alkalmazhatunk.

A következőkben részletesebben foglalkozunk az `<input />` elemmel, illetve az egyéb, űrlapokhoz köthető társaival.

Szöveges adatok fogadása

Ha a látogatóinktól valamilyen adatot szeretnénk fogadni, az `<input />` elemet kell használnunk. Mindössze arra kell ügyelnünk, hogy az elem a `<form>` és a `</form>` címkék közé kerüljön, egyébként a szövegekhez, képekhez és más HTML-elemekhez képest akárhogy elhelyezhetjük. Ha például a látogató nevére vagyunk kíváncsiak, a következőt írhatjuk:

```
What's your name? <input type="text" size="50" maxlength="100"
name="name" />
```

A `type` jellemzővel adhatjuk meg, hogy milyen típusú űrlapelemet szeretnénk alkalmazni – esetünkben egyszerű, egysoros szövegmezőről van szó. (Az egyes típusokról a következőkben bővebben szólnunk.)

A `size` jellemző azt határozza meg, hogy nagyjából hány karakter széles legyen a szövegmező. Ha arányos betűtípust használunk, a bevitt szöveg szélessége a tartalmától függően változó lehet. Ha a bemenet hosszabb, mint a megadott szélesség, a legtöbb böngésző automatikusan balra görgeti a szöveget.

A `maxlength` jellemző határozza meg, hogy hány karaktert írhat a felhasználó a szövegmezőbe. Ha ennél többet próbál beírni, a fölös karakterek egyszerűen nem jelennek meg. Az itt megadott hossz független a szövegmező fizikai méretétől; hosszabb vagy rövidebb is lehet annál. A `size` és a `maxlength` jellemzők csak a `type="text"` megadása esetén használatosak, ugyanis a többi beviteli elem (jelölőnégyzetek, választógombok és egyebek) rögzített méretűek.



Ha azt szeretnénk, hogy a felhasználó úgy írhasssa be az adatait, hogy ezek ne jelenjenek meg a képernyőn, az `<input type="text" />` helyett az `<input type="password" />` kódot használhatjuk. Így a beírt szöveg helyén csillagok (***) tűnnek fel. A `size`, a `maxlength` és a `name` jellemzők ugyanúgy működnek a `type` jellemző `password` és `text` értéke esetén. Mindazonáltal ne feledjük, hogy ez a védelem csak vizuális – nem jár titkosítással vagy a jelszó átvitelének bármiféle védelmével.

Az űrlapelemek elnevezése

Bármilyen típusú legyen is a beviteli elem, nevet kell adnunk neki. Tetszőleges neveket használhatunk, mindössze annyi megkötéssel, hogy az űrlapon belül mindegyikük különbözzön (kivéve a választógombok és a jelölőnégyzetek esetét, amelyekről még szót ejtünk ezen az órán). Amikor a kiszolgálóoldali parancsfájl feldolgozza az űrlapot, az egyes adatokat a nevük alapján azonosítja – ezekből a nevekből változók lesznek, a változók pedig értéket kapnak (ez lehet a felhasználó által beírt érték, illetve az az érték, amelyik a felhasználó által kiválasztott lehetőséghez tartozik).

Így, ha felhasználó például a Jane Doe nevet adja meg a korábban létrehozott szövegmezőben, egy változó kerül az űrlapot feldolgozó parancsfájllhoz. Ennek a változónak a neve `name`, az értéke pedig `Jane Doe` lesz. Az űrlapkezelő parancsfájlok ilyesfajta változónevekkel és értékekkel dolgoznak.

A név-érték párokra további példákat láthatunk a következőkben.



Az űrlapfeldolgozó parancsfájlokat jelentős mértékben leegyszerűsítettük annak érdekében, hogy a könyv keretein belül érthetőek legyenek. A parancsfájlokban szereplő változók pontos megjelenése (vagyis a nevük) az alkalmazott programozási nyelvtől függ, de nem térünk el túlzottan az igazságtól, ha azt mondjuk, hogy a beviteli elem nevéből változónév lesz, a megadott érték pedig ennek a változónak az értékéül szolgál a kiszolgálóoldali parancsfájlból.

Rejtett adatok az űrlapokon

Előfordulhat, hogy szeretnénk bizonyos adatokat átadni a feldolgozó parancsfájlnak, de nem akarjuk, hogy a felhasználó tudomást szerezzen erről. Ilyen esetekben használjuk az `<input />` elemet a `type="hidden"` jellemzővel. Ez a beállítás semmilyen hatással nincs az űrlap megjelenésére, egyszerűen elküldi az általunk megadott név-érték párt az űrlap többi adata mellett.

Ha a webtárhely-szolgáltatóunktól kapott űrlapfeldolgozó parancsfájlt használunk, ezzel a jellemzővel adhatjuk meg, hogy milyen címre küldje az elektronikus levelet a kapott adatokkal. Az alábbi kód esetében ez a cím a `me@mysite.com`:

```
<input type="hidden" name="mail_to" value="me@mysite.com" />
```

A parancsfájlok általában legalább egy-két rejtett beviteli elemet hozzácsapnak az adatfolyamhoz, amelyek hasznosak lehetnek az eredmény kézhez vételénél – ilyen lehet az elektronikus levelezési cím vagy az üzenet tárgya. Ha a szolgáltatónk parancsfájlját használjuk, nézzünk utána a leírásában, hogy milyen rejtett mezők megadását várja el tőlünk.

Az űrlapok beviteli vezérlői

A felhasználó adatainak begyűjtésére különféle beviteli vezérlők állnak a rendelkezésünkre. A szövegbevitt már láthattuk, a következőkben pedig sorra vesszük a további lehetőségeket.

Jelölőnégyzetek

A legegyszerűbb beviteli vezérlő a *jelölőnégyzet*, amely apró négyzet alakjában jelenik meg az űrlapon. A felhasználók az egy csoportba tartozó jelölőnégyzetek közül tetszőleges számút bekapcsolhatnak. Példaként nézzük meg a 22.1. példa „Please check all that apply” (Jelöld be a szerinted jellemző válaszokat) címke alatti jelölőnégyzeteket, amelyek közül valóban bármennyi, általunk jellemzőnek vélt elemet kiválaszthatunk.

A 22.1. példa jelölőnégyzetekhez tartozó részében láthatjuk, hogy a `name` tulajdonság értéke – `website_response[]` – mindegyiküknél megegyezik.

```
<input type="checkbox" name="website_response[]" value="I
  really like your Web site." /> I really like your Web site.<br />
<input type="checkbox" name="website_response[]" value="One
  of the best sites I've seen." /> One of the best sites
  I've seen.<br />
<input type="checkbox" name="website_response[]" value="I sure
  wish my site looked as good as yours." /> I sure wish my site
  looked as good as yours.<br />
<input type="checkbox" name="website_response[]" value="I have
  no taste and I'm pretty dense, so your site didn't do much for
  me." /> I have no taste and I'm pretty dense, so your site
  didn't do much for me.<br />
```

A szögletes zárójelek (`[]`) jelenléte azt üzeni a feldolgozó parancsfájlnak, hogy ezúttal egyetlen változóba értékek egész sorozata kerül, nem csak egy érték (persze, ha a felhasználó csak egy jelölőnégyzetet választ, még ez is előfordulhat). Ha a látogatónk az első jelölőnégyzetet választja, az „I really like your Web site.” karakterlánc kerül a `website_response[]` nevű vödörbe. Ha a harmadik jelölőnégyzetet is kiválasztja, az „I sure wish my site looked as good as yours.” is idekerül. A feldolgozó parancsfájl adatok tömbjeként tekint a kapott eredményre, ahelyett, hogy egyetlen adategységet látna benne.



Ha úgy látjuk, hogy egy beviteli elem felirata túlzottan közel áll magához az elemhez, egyszerűen helyezzünk el pár szóközt az `<input />` elem vége és a felirat eleje között, valahogy így:

```
<input type="checkbox" name="mini" /> Mini Piano Stool
```

Mindazonáltal az is előfordulhat, hogy egyes jelölőnégyzetek csoportban állnak, mégis különböző változóneveket alkalmaznak. Az alábbi kód is egy ilyen csoportot mutat be:

```
<input type="checkbox" name="liked_site" value="yes" /> I really like
  your Web site.<br />
<input type="checkbox" name="best_site" value="yes" /> One of the best
  Sites I've seen.<br />
<input type="checkbox" name="my_site_sucks" value="yes" />I sure wish my
  site looked as good as yours.<br />
<input type="checkbox" name="am_dense" value="yes" />I have no taste and
  I'm pretty dense, so your site didn't do much for me.<br />
```

A kódban például az első jelölőnégyzethez tartozó változónév a „liked_site”, az értéke pedig (a bekapcsolása esetén) „yes”.

Ha azt szeretnénk, hogy a jelölőnégyzet kezdetben bekapcsolt állapotban jelenjen meg a böngészőben, tüntessük fel a kódban a `checked` jellemzőt. Az alábbi kódban két jelölőnégyzet szerepel – az első alapértelmezésben bekapcsolt állapotban:

```
<input type="checkbox" name="website_response[]" value="I
  really like your site." checked="checked"/> I really like your
  site.<br />
<input type="checkbox" name="website_response[]" value="One
  of the best sites I've seen." /> One of the best sites I've
  seen.<br />
```



Az XHTML megköveteli, hogy minden jellemző mellett tüntessünk fel egy egyenlőségjelet, majd ezt követően egy értéket. Mindez világossá teszi, hogy miért használtuk a `checked="checked"` alakot az egyszerű `checked` helyett. Ez a szabály vonatkozik mindenféle logikai értékre (true/false, on/off, yes/no stb.), amellyel a HTML-programozás során találkozhatunk.

Példánkban az „I really like your site.” (Nagyon tetszik a webhelyed) jelölőnégyzet szerepel kezdetben bekapcsolt állapotban, így ha a látogatóknak más a véleménye, ennek közléséhez a jelölőnégyzetre kell kattintania. A „One of the best I've seen.” (Az egyik legjobb, amit eddig láttam) ugyanakkor kezdetben nincs bekapcsolva, így a látogatónak rá kell kattintania, ha közölni szeretné velünk ebbéli megjegyzését.

Azok a jelölőnégyzetek, amelyek kikapcsolt állapotban maradnak, egyáltalán nem jelennek meg az űrlap kimenetében.



Ha mások webes űrlapjait tanulmányozzuk, könnyen bukkanhatunk olyan jelölőnégyzetekre, amelyeknek a neve megegyezik, az értékeik viszont eltérnek, mint az alábbi kódban:

```
<input type="checkbox" name="pet" value="dog"> dog<br />
<input type="checkbox" name="pet" value="cat"> cat<br />
<input type="checkbox" name="pet" value="iguana"> iguana<br />
```

Ha a felhasználó egynél több jelölőnégyzetet kapcsol be, több mint valószínű, hogy a parancsfájl csak az utolsó értéket dolgozza fel, ezért érdemes jobban átgondolnunk a jelölőnégyzetek csoportjainak kialakítását, a csoport nevét és az értékek halmazát.

Választógombok

A *választógombok* – ahol a csoporton belül csak egy lehetőséget választhatunk ki – majdnem olyan egyszerűen megvalósíthatók, mint a jelölőnégyzetek. A választógombok legegyszerűbb alkalmazási területét az igen/nem választási lehetőségek, valamint az olyan szavazások jelentik, ahol csak egy jelöltre adhatjuk le a voksunkat.

Választógomb létrehozásához alkalmazzuk a `type="radio"` beállítást, és hozunk létre minden lehetőség számára egy-egy `<input />` elemet. A `name` jellemzőnek egy csoporton belül adjunk azonos értéket, de ne használjuk a jelölőnégyzeteknél megismert szögletes zárójelpárt:

```
<input type="radio" name="vote" value="yes" checked="checked" /> Yes<br />
<input type="radio" name="vote" value="no" /> No <br/>
```

A `value` tetszőleges értéket vagy kódot tartalmazhat. Ha használatba vesszük a `checked` jellemzőt, a böngészőben az adott lehetőség eleve kiválasztva jelenik meg. Azonos `name` jellemzővel rendelkező választógombok közül legfeljebb egyet kapcsolhatunk be.

Ha egy űrlap tervezése során döntenünk kell, hogy jelölőnégyzeteket vagy választógombokat használjunk, tegyük fel magunknak a következő kérdést: az űrlapon feltett kérdésre egyértelmű válasz adható? Ha igen, döntsünk a választógombok használatára mellett.

Választólisták

A *görgethető listák* és a *lenyíló listák* létrehozására egyaránt a `<select>` elem szolgál, amelyet az `<option>` elemmel együtt kell használnunk, amint azt az alábbi példa mutatja:

```
<select name="extras" size="3" multiple="multiple">
  <option value="Electric windows" selected="selected">Electric
  windows</option>
  <option value="Sunroof">Sunroof</option>
  <option value="AM/FM Radio">AM/FM Radio</option>
  <option value="CD Player">CD Player</option>
  <option value="GPS">GPS</option>
</select>
```

A `<select>` és `</select>` címkék között kizárólag `<option>` és `</option>` elemek állhatnak.

A text beviteli típustól eltérően a `size` jellemző ezúttal azt határozza meg, hogy hány elem jelenjen meg egyszerre a választólistában. Ha például a `size="2"` beállítással élnénk a fenti kódban, csak az első két lehetőség jelenne meg, és egy gördítősávot kapnánk, amellyel a felhasználó elérhetné a harmadikat is.

Ha a `multiple` jellemzőt is használatba vesszük, a felhasználó egyszerre több lehetőséget is kiválaszthat, a `selected` jellemzővel pedig eleve kiválaszthatjuk valamelyik lehetőséget. Az űrlap adatainak elküldésekor a kiválasztott lehetőség `value` jellemző-jének értéke kerül a parancsfájlhoz.



Ha kihagyjuk a `size` jellemzőt, vagy a `size="1"` beállítással élünk, egy lenyíló listát kapunk. Itt nincs lehetőségünk több elem kiválasztására – a lista logikailag egy választógomb-csoporttal egyenértékű. Az alábbi példában bemutatjuk az igen/nem választás egy újabb módját:

```
<select name="vote">
  <option value="yes">Yes</option>
  <option value="no">No</option>
</select>
```

Többsoros szövegmezők

A korábban bemutatott `<input type="text">` jellemzővel egyetlen sornyi szöveget vihetünk be. Ha egyetlen elemmel többsoros szöveget szeretnénk fogadni, használjuk a `<textarea>` és `</textarea>` címkéket. Bármit is írjunk a kódban e két címke közé, az alapértelmezett szöveggént jelenik meg a szövegmezőben. Íme egy példa:

```
<textarea name="comments" rows="4" cols="20">Please send more
information.
</textarea>
```

Amint azt talán ki is találtuk, a `rows` és a `cols` jellemzők a szövegmezőben helyet kapó sorok és oszlopok számát adják meg. A `cols` jellemző kissé kevésbé pontos, mint a `rows`, mivel az egy soron belüli karakterek számára ad becslést. A többsoros szövegmezők gördítősávval is rendelkeznek, így a felhasználó a látható méretüknél hosszabb szöveget is elhelyezhet bennük.

Az űrlapadatok elküldése

Minden űrlapon szerepelnie kell egy gombnak, amellyel elküldhetjük a kapott adatokat a kiszolgálónak. A `value` jellemző segítségével tetszőleges feliratot elhelyezhetünk a gombon:

```
<input type="submit" value="Place My Order Now!" />
```

A kapott szürke gomb mérete megfelel majd a `value` jellemzőben megadott szövegnek. Ha a felhasználó rákattint, a böngésző elküldi az űrlap adatait az `action` jellemzőben megadott elektronikus levélcímrre, illetve parancsfájlnak.

Feltüntethetünk egy `Reset` (Alaphelyzet) gombot is, amely törli az űrlap összes bejegyzését, így a felhasználó újratekesheti a kitöltést, ha meggondolta magát, vagy elrontott valamit. Ehhez alkalmazzuk az alábbi kódot:

```
<input type="reset" value="Clear This Form and Start Over" />
```

Ha a szokásos `Submit` (Küldés) és `Reset` gombok túlságosan szürkének tűnnek, jusson eszünkbe, hogy a CSS segítségével testreszabhatjuk őket. Ha még ez sem elég, nyilván örömmel fogadjuk a hírt, hogy akár saját képeket is elhelyezhetünk a gombokon. Ha ezt a `Submit` gombbal szeretnénk megtenni, az alábbi kóddal érhetünk célt:

```
<input type="image" src="button.gif" alt="Order Now!" />
```

Így a `button.gif` kép jelenik meg az oldalon, és a felhasználó erre kattintva küldheti el az űrlap adatait. Itt alkalmazhatjuk az `` elemnél megszokott jellemzőket is, mint az `alt` vagy a `style`.

Összefoglalás

Ezen az órán bemutattuk, miként készíthetünk HTML-űrlapokat, amelyek révén a webhelyünk látogatói adatokat közölhetnek velünk. A kapott adatok kezelésére nem vállalkoztunk, ugyanis ehhez külső parancsfájltra van szükség.

Megismerkedtünk az űrlapok legfontosabb elemeivel, valamint megtanultuk azt is, hogy miként értelmezik a parancsfájlok ezeknek az elemeknek a neveit és értékeit. Ha a későbbiekben eljutunk odáig, hogy elkészítsük a háttérben futó parancsfájlt, a felület részleteit már ismertnek vehetjük.

A 22.1. táblázatban sorra vesszük az ezen az órán bemutatott HTML-elemeket és -jellemzőket.

22.1. táblázat A 22. órán megismert HTML-elemek és -jellemzők

Elem/jellemző	Leírás
<code><form>...</form></code>	Egy beviteli űrlapot jelöl.
<code>action="parancsfájl_url"</code> <code>method="post/get"</code>	A bevitt adatokat feldolgozó parancsfájl címe. Az adatok elküldésének módja. Általában inkább a post, mint a get értéket használjuk.
<code><input /></code>	Az űrlap egy beviteli eleme.
<code>type="vezérlő_típusa"</code>	A beviteli elem típusa. A lehetséges értékek: checkbox, hidden, radio, reset, submit, text és image.
<code>name="név"</code>	Az elem egyedi azonosítója, amelyet az adatokkal együtt elküldünk a kiszolgálónak.
<code>value="érték"</code>	Szöveg vagy rejtett elem alapértelmezett értéke. Jelölőnégyzetek és választógombok esetén a kiszolgálónak elküldött értéket jelöli. A Reset és a Submit gomboknál a gomb feliratát tartalmazza.
<code>src="kép_url"</code>	A felhasznált kép forrásfájlja.
<code>checked="checked"</code>	Jelölőnégyzetek és választógombok esetén jelzi, hogy az adott elemet bekapcsoltuk.
<code>size="szélesség"</code> <code>maxlength="maximális_hossz"</code>	Szöveges beviteli mező szélessége, karakterben mérve. A szövegmezőbe beírható karakterek legnagyobb száma.
<code><textarea>...</textarea></code>	Többsoros szövegbeviteli elem, amelyben alapértelmezett szöveget is elhelyezhetünk.
<code>name="név"</code>	A parancsfájlnak átadott név.
<code>rows="sorok_száma"</code>	A szövegmezőben megjelenő sorok száma.
<code>cols="karakterek_száma"</code>	A szövegmezőben megjelenő oszlopok (karakterek) száma.
<code><select>...</select></code>	Menüt, illetve gördíthető listát jelenít meg.
<code>name="név"</code>	A parancsfájlnak átadott név.
<code>size="elemszám"</code>	A megjelenítendő elemek száma. Ha alkalmazzuk a size jellemzőt, gördíthető listát kapunk, ellenkező esetben pedig lenyíló listához jutunk.
<code>multiple="multiple"</code>	Lehetővé teszi, hogy a lista több elemét is kiválasszunk.
<code><option>...</option></code>	A <code><select></code> elem egy választási lehetőségét jelöli.
<code>selected="selected"</code>	Ha alkalmazzuk, az adott <code><option></code> elemet alapértelmezés szerint kiválasztjuk a listában.
<code>value="érték"</code>	Ezt az értéket küldi el a böngésző a parancsfájlnak, ha a felhasználó ezt a lehetőséget választja.

Kérdezz-felelek

- K: *Úgy hallottam, veszélyes lehet a hitelkártyák számát az Interneten keresztül elküldeni. Valós veszély, hogy valaki hozzájut az adatokhoz, amíg eljutnak az űrlaptól a kiszolgálóig?*
- V: Valóban lehetséges, hogy valaki útközben elfogja az űrlap (hasonlóképpen egy weboldal vagy elektronikus levél) adatait. Ha tehát hitelkártyaszámot vagy más érzékeny adatokat kérünk, lehetővé kell tennünk a biztonságos böngészést egy biztonságos kiszolgálón egy SSL- (Secure Sockets Layer) tanúsítvány révén. Ezekhez a tanúsítványokhoz közvetlenül is hozzájuthatunk a megfelelő cégektől, amilyen például a VeriSign (<http://www.verisign.com/>), de jobb, ha a webtárhely-szolgáltatónknál nézünk utána, hogy milyen típusú SSL-tanúsítványokat fogad el és értékesít viszonteladóként.
- A veszély valós nagyságának felméréséhez azonban el kell mondanunk, hogy messze nehezebb az Interneten haladó adatokat elfogni, mint kilesni a hitelkártya adatait egy gyanútlan vásárló válla felett. Persze ennek tudatában is fontos, hogy érzékeny üzleti adatok – például hitelkártyaszámok – kezelésénél biztonságos oldalakat használjunk, különösen, ha valaki megtisztel azzal a bizalommal, hogy ránk bízza ezeket az adatokat.
- K: *Elhelyezhetjük az űrlapokat CD/DVD lemezen is, vagy feltétlenül az Interneten kell közzétenni őket?*
- V: Az űrlapot bárhol elhelyezhetjük, ahol egyébként weboldalakat tárolhatnánk. Ha nem webkiszolgálót, hanem merevlemezt vagy CD-t választunk a tárolásra, a felhasználók nyugodtan kitölthetik az adatokat, akár él az internetkapcsolat, akár nem. Persze, ha a Submit gomb megnyomására kerül a sor, már szükség van hálózati (akár helyi hálózati) kapcsolatra, ellenkező esetben az adatok nem jutnak el a feldolgozásért felelős parancsfájlhoz.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Milyen HTML-kóddal hozhatunk létre egy űrlapot egy vendégkönyv számára, amely elkéri a látogató nevét, nemét, életkorát és elektronikus levélcímét? Tegyük fel, hogy az űrlapot feldolgozó parancsfájl a /cgi/generic címen található, továbbá az alábbi rejtett beviteli elemet is fel kell tüntetnünk, amely jelzi a parancsfájlnak, hogy hová küldje az eredményt:

```
<input type="hidden" name="mailto" value="you@yoursite.com" />
```

2. Tételezzük fel, hogy készítettünk egy `sign-in.gif` nevezetű képet. Miként használnánk ezt az előbbieken ismertetett vendégkönyv Submit gombjaként?

Válaszok

1. Az alábbihoz hasonló HTML-kóddal célt érünk (természetesen a megfelelő DOCTYPE meghatározással együtt):

```
<html>
  <head>
    <title>My Guestbook</title>
  </head>

  <body>
    <h1>My Guestbook: Please Sign In</h1>
    <form method="post" action="/cgi/generic">
      <p>
        <input type="hidden" name="mailto" value="you@yoursite.com" />
        Your name: <input type="text" name="name" size="20" /><br />
        Your sex: <input type="radio" name="sex" value="male" /> male
        <input type="radio" name="sex" value="female" /> female<br />
        Your age: <input type="text" name="age" size="4" /><br />
        Your e-mail address: <input type="text" name="email" size="30" />
        <br />
        <input type="submit" value="sign in" />
        <input type="reset" value="erase" />
      </p>
    </form>
  </body>
</html>
```

2. Cseréljük a következő kódsort...

```
<input type="submit" value="Sign In" />

...erre:

<input type="image" src="sign-in.gif" alt="Sign In" />
```

Gyakorlatok

- Készítsünk egy űrlapot, amelyen szerepel a beviteli elemek és választólisták minden típusa – így meggyőződhetünk arról, hogy valóban elsajátítottuk-e a használatukat.
- Derítsük ki, hogy a webtárhely-szolgáltatónk milyen lehetőségeket nyújt az űrlapok adatainak kezelésére. Válasszunk egyet a rendelkezésre álló parancsfájlok közül, és alkalmazzuk az előzőekben elkészített űrlap adatainak feldolgozására.



23. ÓRA

Webhelyek összeállítása és kezelése

A lecke tartalma:

- Hogyan határozhatjuk meg, hogy elég-e egyetlen weblap a kívánt tartalom megjelenítéséhez?
- Hogyan állítsunk össze egy egyszerű webhelyet?
- A nagyobb webhelyek összeállításának fortélyai
- Hogyan írhatunk fenntartható HTML-kódot?

Könyvünk nagy részében azzal foglalkoztunk, hogy miként építhetjük fel a saját webes tartalmunkat a szöveges részektől a grafikán át a multimédiás elemekig. Időközben tettünk pár megjegyzést arra vonatkozóan, hogy miként érdemes gondolkodnunk a tartalom életciklusáról – ezt az órát azonban teljes mértékben annak szenteljük, hogy a munkánkat egészében tekintsük át.

Bemutatjuk, miként szervezhetünk és jeleníthetünk meg több weblapot úgy, hogy a látogatóink könnyedén eligazodjanak közöttük, anélkül, hogy elvesztenék a fonalat. Tanulunk arról is, hogyan tehetjük emlékezetesebbé a webhelyünket, hogy az egyszeri látogatóinkból visszatérő vendégek váljanak. A webfejlesztők „ragadoárnak” nevezik

azokat a weboldalakat, amelyeket a látogatók nem szívesen hagynak el. Fejezetünk remélhetőleg segít abban, hogy valódi ragacsot keverjünk ki webes boszorkánykonyhánkban.

Mivel a webhelyek frissítésére gyakorta szükség van (és ez rendszerint elemi elvárás is velük szemben), alapvető fontosságú, hogy olyan weboldalakat készítsünk, amelyeknek a fenntartása egyszerű. Ezen az órán megtanuljuk, hogyan mellékeljünk megjegyzéseket és egyéb leírásokat az oldalainkhoz, így a későbbiekben megkönnyíthetjük a módosításukat a magunk és munkatársaink számára.



Önálló feladat

Vizsgáljuk felül az oldalaink szerkezetét!

Ha idáig eljutottunk a könyv olvasásában, már bizonyára rendelkezünk elegendő tudással a HTML- és CSS-programozásról ahhoz, hogy a webhelyünk tartalmának java részét felépítsük. Minden bizonnyal több weboldalt is elkészítettünk, sőt némelyiküket közzé is tettük.

Óránk anyagának olvasása során gondolkodjunk azon, hogy miként rendeztük el az oldalaink anyagát, és hogyan tudnánk ezen a rendezésen javítani. Alkalmaztunk megjegyzéseket a HTML-kódban, illetve készítettünk valamiféle leírást a kód szerkezetéről a webhely későbbi karbantartói számára? Ha nem, itt az idő, hogy elkezdjük! Ha eközben szükség lesz akár az összes oldal átalakítására, ne lepődjünk meg, és különösképpen ne aggódjunk – az eredmény szinte biztosan megéri a fáradságot!

Amikor egy oldal is elég...

A jól működő és a szemnek is kedves webhelyek felépítése és összeállítása nem szükségképpen bonyolult feladat. Ha egyetlen dolog (például egy helyi esemény) számára készítünk webes megjelenést, amelynek a leírásához nincs szükség sok adatra, egyetlen oldallal is célt érhetünk, és még csillogó-villogó grafikai elemeket sem kell használnunk. Igazság szerint az ilyen egyoldalas megjelenés számos előnnyel jár:

- A webhelyen közölt adatok letöltése gyorsabb, mint a kiterjedtebb webhelyek esetében.
- A teljes webhely tartalma kinyomtatható egyetlen paranccsal, még ha az eredmény több lapot vesz is igénybe.
- A látogatók könnyedén lemezre menthetik a webhelyet a későbbiekre, különösen akkor, ha a grafikai elemek számát korlátozzuk.
- Az oldalon belüli hivatkozások többnyire gyorsabban célba jutnak, mint az oldalak közti társaik.

A 23.1. ábra egy weboldal első részét mutatja, amely jobban szolgálja a célközönségét egyetlen hosszú oldalként, mint több oldalból álló webhelyként. Az oldal – más bevezető oldalakhoz hasonlóan – azzal indul, hogy röviden leírja, mit tartalmaz, és kinek szól. Ezt követi egy részletes tartalomjegyzék, amelynek a pontjai egyenesen az oldal megfelelő részeihez visznek. Az oldal összesen nagyjából nyolc nyomtatott oldalnyi anyagot tartalmaz, amely bevezet az ingatlanvásárlás világába – ezt valóban érdemes lehet kinyomtatni, és esetleg jegyzetekkel kiegészíteni.

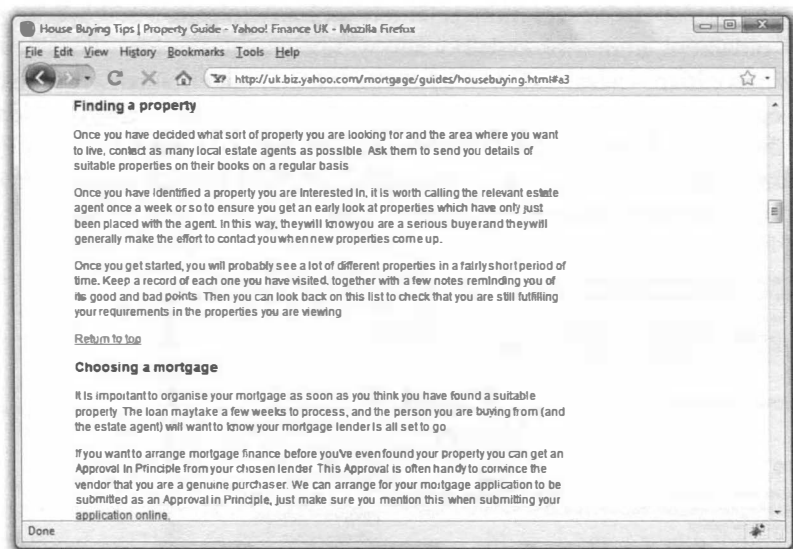


23.1. ábra

A jó tartalomjegyzék segít, hogy eligazodjunk egy hosszú weboldalon

Amint a 23.2. ábrán láthatjuk, az oldal egyes részeinek végén egy-egy hivatkozást találunk, amely visszaugrik a tartalomjegyzékhez, így az oldalon hasonlóan navigálhatunk, mintha egy több oldalból álló webhelyen lennénk. Mivel a tartalomjegyzék jó hivatkozásgyűjteményt ad, a látogatók sokkal szívesebben tárolnak könyvjelzőként, illetve mentenek lemezre egyetlen oldalt nyolc-tíz másik helyett.

A könyv eddigi részeiben látott csillogó-villogó grafikák és oldaltervek után persze könnyű megfeledezni arról, hogy milyen előnyökkel rendelkezik a jó öreg vázlat, amely igen hatékony eszközt ad a kezünkbe a hosszabb weboldalak kidolgozásához.



23.2. ábra

A hosszú weboldalak szakaszai végén mindig helyezzünk el egy-egy hivatkozást a tartalomjegyzékre

Egyszerű webhely felépítése

Jóllehet az egyoldalas webhelyeknek is megvan a maguk szerepe, a legtöbb cég és egyéni tartalomszolgáltató rövid, könnyen átlátható oldalakra bontja a webhelyét, grafikus navigációs elemekkel körülvéve, amelyek lehetővé teszik, hogy a látogatók pár kattintással minden szükséges tájékoztatáshoz hozzájussanak.

Ráadásul több, rövidebb oldallal kiváltva a hosszabbakat, elejét vehetjük a fárasztó görgetésnek, ami különösen azoknak nehezen elviselhető, akik mobilkészüléken vagy viszonylag kicsi (800 x 600 képpontnál kisebb) felbontású monitoron nézik a tartalmat.

A honlap vagy kezdőoldal szerepe egyrészt abban áll, hogy megjelenítse a céget az Interneten – fontosabb azonban az a szerepe, hogy kiindulópontként szolgáljon a webhely többi része felé. A webhely főoldalán elegendő tájékoztatást kell nyújtanunk a szervezetről ahhoz, hogy a felhasználó világos képet alkothasson róla, továbbá itt érdemes elhelyeznünk a cég hagyományos címét, telefonszámát és elektronikus levél-címét, ha a látogató kérdésekkel vagy visszajelzésekkel kíván élni. Fontos emellett, hogy a kezdőoldallról átláthatóak legyenek azok az irányok, amelyeket követve bejárhatjuk a webhelyet. A 23.3. ábrán egy jól felépített kezdőoldalt láthatunk, a fent említett elemekkel: megtaláljuk itt a cég alapvető adatait, a kapcsolati lehetőségeket, valamint a továbblépés irányait a különböző érdeklődési körű látogatók számára.



23.3. ábra

Az itt látható egyetemi kezdőoldal egyszerű megjelenésével, szikár, de lényegre törő grafikai elemeivel és átlátható szerkezetével szinte csábítja a látogatót a felfedezésre

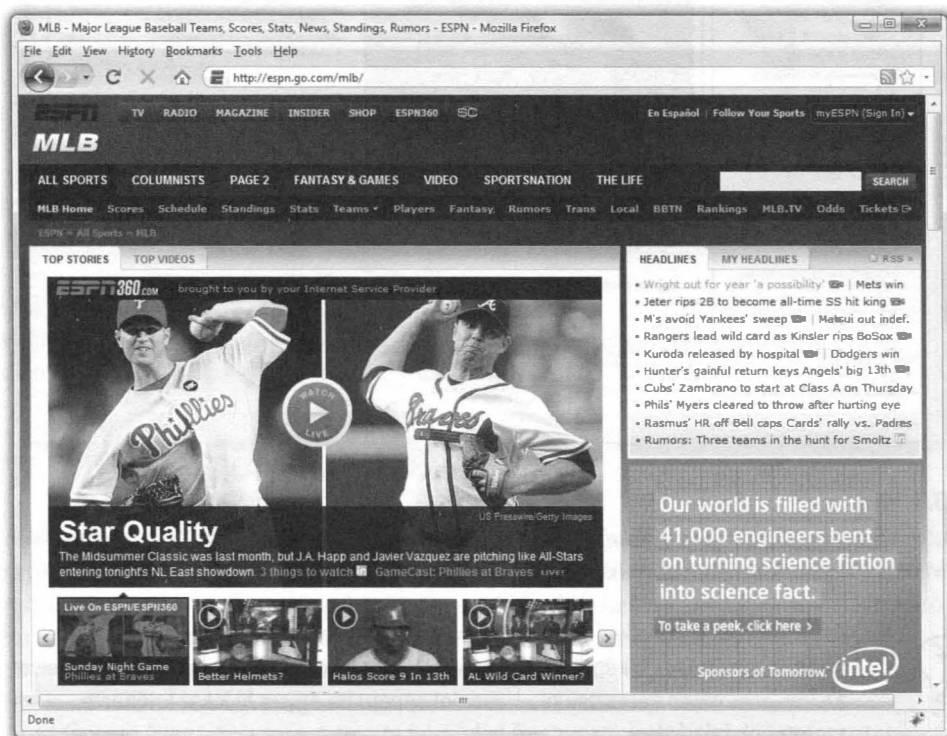


Akármeekkora webhellyel is dolgozzunk, érdemes gondosan elrendeznünk az erőforrásainkat. A képeket például elhelyezhetjük egy külön mappában, amelyet az `images` névre keresztelünk. Hasonlóképpen, ha egyes fájlokat letöltésre szeretnénk felajánlani, nagyszerű helyük lehet egy `downloads` nevű mappában. Így könnyebben nyilvántarthatjuk az egyes erőforrásokat a típusuk alapján (HTML-oldalak, GIF-képek, és így tovább). Ha pedig a webhely tartalmát különböző kategóriákba – „Company”, „Products”, „Press” stb. – rendezzük, az egyes oldalakat hasonló nevű könyvtárakban (`company`, `products`, `press` stb.) tárolhatjuk.

A webhelyfejlesztők kezdő korukban gyakran elkövetik azt a hibát, hogy olyan oldalakat készítenek, amelyek jelentősen elütnek a webhely többi oldalától. Hasonlóan gyakori hiba az olyan, nyilvánosan elérhető képtárak használata, amelyeket webfejlesztők ezrei koptattak unalmassá. Ne feledjük, az Interneten egyetlen kattintással a világ másik felére ugorhatunk! Egyetlen valódi módszer kínálkozik arra, hogy a webhelyünket megjegyezhetővé és egységes egészként felismerhetővé tegyük – ha

tartjuk magunkat egy egyedi, félreismerhetetlen vizuális témához. Más szóval, törekedjünk arra, hogy elkülönítsük magunkat más webhelyektől, de a webhelyen belül tartsuk szem előtt az egységességet.

Az egységesség előnyeiről könnyen meggyőződhetünk, ha ellátogatunk egy olyan nagy és népszerű webhelyre, mint például az ESPN.com. Ha összehasonlítjuk az MLB (23.4. ábra) és az NFL (23.5. ábra) szakaszt, meglehetősen hasonló oldalszerkezetet találunk.



23.4. ábra

Az ESPN.com MLB szakasza

Mindkét példán láthatjuk a navigációs elemeket az oldal tetején (néhány almenüvel egyetemben), egy nagyobb területet középen, a kiemelt hírhez tartozó képpel, egy téglalapot a jobb oldalon a friss hírek hivatkozásaival, valamint ez alatt egy további téglalapot egy hirdetéssel. Az MLB és az NFL oldala között egyetlen igazi különbséget fedezhetünk fel: ez pedig a színösszeállítás – az MLB esetében alapvetően kék árnyalatokat láthatunk, az NFL-nél pedig a zöld szín uralkodik. Hiába térnek el azonban a színek, mindkét esetben azonnal tudjuk, hogy ha a legfrissebb hírekre vagyunk

kíváncsiak, az oldal jobb szélén kell keresgelnünk. Ha pedig a webhely egy másik részére szeretnénk továbbhaladni, vagy visszatérni a főoldalra, felül találjuk a navigációs elemeket.



23.5. ábra

Az ESPN.com NFL szakasza

Az elemek összhangja biztonságérzetet ad a látogatóknak, akiknek így nem kell tartaniuk attól, hogy valahol elvesztik a fonalat. Ennek a módszernek a karbantartás szempontjából is vannak előnyei, hiszen így bizonyos elemeket újra és újra felhasználhatunk. Erre jellemzően a dinamikus programozás keretein belül kerül sor. Jóllehet ez a témakör kívül esik könyvünk keretein, annyit elmondhatunk, hogy a módszer előnye abban áll, hogy az egyes HTML-kódrészleteket így nem kell lemásolnunk, majd újra és újra beillesztenünk – elég egy helyen tárolni és dinamikusán alkalmazni a tartalomra. Így az esetleges módosításokat nem kell ezernyi fájlban elvégezni, elég, ha egy helyen tesszük ezt meg.

Nagyobb webhely összeállítása

Egy összetett webhelyen a kifinomult szerkezet és grafika egységes alkalmazása sokat emelhet a webhely szervezetségén és megjelenésén. A külső és a belső összhangjának bemutatására tekintsük meg néhány olyan webhely navigációs lehetőségeit (és az emögött meghúzódó szervezetséget), amely rengetegféle tartalmat szolgáltat a szerteágazó közönség számára.

A 23.6. ábrán az Amazon.com kezdőoldalt láthatjuk, amelyen az oldalsó navigációs sáv egyik pontját választottuk. Az Amazon célja nem több és nem kevesebb, mint különféle termékek eladása. Következésképpen egyszerű, hogy a navigáció alapján a különféle termékcsoportok szolgáljanak, amint azt az ábrán is láthatjuk.



23.6. ábra

Az Amazon.com elsődleges navigációs elemei a termékcsoportok

Jóllehet az Amazon elsődleges célja a termékek eladása, mégiscsak kell némi tájékoztatást adnia arról, hogy voltaképpen mi is ez a cég valójában, miként léphetünk vele kapcsolatba, sőt a kereskedő–vásárló kapcsolat javítására is érdemes módot adnia. Az ilyesfajta tájékoztatást a weboldal alsó részének, vagyis láblécének hivatkozásait

követve kaphatjuk meg, amely kívül esik az ábra képernyőfelvételének területén. A webhely sablonjának létrehozásakor meg kell határoznunk a tartalom legfontosabb területeit és ezek viszonyait, mindeközben azonban a felhasználók kielégítő tájékoztatásáról sem szabad megfeledkeznünk – különösen ha ez javítja a rólunk kialakított képet, és a felhasználó úgy érzi, hogy figyelnek rá.

Következő példánk a Starbucks.com egy másodlagos oldala. Érdekes megfigyelnünk, hogy a Starbucks.com webhely minden oldala az egyik ismert navigációs eljárást alkalmazza: egy vízszintes sávon találhatjuk az elsődleges navigációs elemeket, a másodlagos elemek pedig a bal oldalon látható függőleges oszlopban kaptak helyet. Amint a 23.7. ábrán is láthatjuk, az aktív navigációs elem (ez esetben az „about us”) kiemelve jelenik meg. Ez a képi jelzés segít, hogy eligazodjunk a webhelyen. Az ilyesfajta jelzések alkalmazása hasznos módszer, hiszen a látogatók nem feltétlenül a kezdőoldaltól érkeznek – egy keresőprogram vagy egy másik webhely hivatkozása is elhozhatja őket az adott weboldalra. Ha pedig látogatóink jöttek, érdemes mindent megtennünk annak érdekében, hogy otthon érezzék magukat, de legalábbis tudják, hová kerültek a webhelyen belül.



23.7. ábra

A Starbucks.com eme másodlagos oldala kiemeli az aktív elsődleges navigációs elemet, míg a másodlagos navigációs elemeket az oldal bal oldalán láthatjuk

Ha megfigyeljük a különböző navigációs elemeket – „our coffees”, „our stores”, „starbucks card”, „at home”, „for business”, „about us” és „shop online” (kávéink, üzleteink, Starbucks-kártya, otthon, vállalkozóknak, magunkról és internetes vásárlás) –, világossá válhat, hogy a webhelynek számos különféle típusú látogatót kell kiszolgálnia, akik más-más célból jutnak el ide. Saját webhelyünk tartalmának szerkesztésénél mérjük fel, hogy melyek a számunkra legfontosabb elemek, majd gondolkodjunk el a felhasználóink lehetséges igényeiről, és végül próbáljunk közéleti találati a kettő között.

A 23.8. ábra egy újabb navigációs módot mutat be, de ezúttal kicsit eltérünk a megszokott felső sáv/bal oldali sáv felosztástól. Itt ugyanis a bal oldali sáv navigációs elemei (vagyis ez esetben a másodlagos navigáció) egy lenyíló menüben is megjelennek a fő navigációs elemek alatt (a 17. órán megtanultuk, miként készíthetünk ilyen menüket). Bármelyik fő navigációs elem fölé helyezzük az egérmutatót, ilyen lenyíló menükhöz jutunk. Ez gyakorlatilag azt jelenti, hogy a látogatók előtt közvetlenül ott van az oldaltérkép, hiszen így egyetlen kattintással eljuthatnak a webhely egyik pontjáról egy tetszőleges másik pontra.



23.8. ábra

A BAWSI.org webhelyen minden fő navigációs elemhez almenü tartozik

Észrevehetjük azt is, hogy az oldalsó navigációs ablak „Overview” (Áttekintés) pontjának stílusa kissé eltér a többitől (lila és vastagabbak a betűi), ami jelzi a látogatóknak, hogy melyik weboldalon is vannak éppen. Ez az apró képi jelzés finoman eligazít a webhely szerkezetén belül.

Rengeteg különböző navigációs rendszer létezik, és számos módon jelezhetjük a látogatóknak, hogy hol tartózkodnak az adott pillanatban, és merre haladhatnak tovább. Mindazonáltal érdemes tudnunk, hogy több kutatás is kimutatta, hogy az emberek összezavarodnak, és elvesztik a biztonságérzetüket, ha hétnél több választási lehetőséggel kerülnek szembe, és igazán akkor érzik magukat biztonságban, ha öt vagy annál kevesebb lehetőség közül kell választaniuk. Így hát próbáljuk elkerülni, hogy egymás mellett ötnél több hivatkozást (akár szöveges, akár grafikus alakban) kínáljunk, hétnél többet pedig semmi esetre sem alkalmazzunk. Az Amazon.com ez esetben némiképp mentességet élvez, hiszen itt egy internetes nagyáruházról van szó, ahol a felhasználók eleve elvárják, hogy rengeteg „osztály” közül választhassanak. Ha azonban már hétnél több hivatkozást kell megjelenítenünk, feltétlenül osszuk fel a listát 5-7 csoportra, kiemelt fejlécekkel – ahogy azt az Amazon.com esetében is láthatjuk a 23.6. ábrán.

Az is sokat segíthet a látogatóinknak a biztos eligazodásban, ha ügyelünk arra, hogy a webhely minden oldala elérhető legyen legfeljebb két (esetleg három) lépésre a kezdőoldaltól. Ha pedig a látogató befejezte egy mellékvágányon található oldal olvasását, segítsünk neki visszajutni a kategória főoldalára, vagy magára a kezdőoldalra. Más szóval, olyan „lapos” hivatkozási szerkezetet építsünk fel, amelyben a legtöbb oldal egy vagy két hivatkozás mélységben fekszik. Nem szerencsés, ha a látogatóknak a böngésző Vissza gombjára kell hagyatkozniuk a webhelyünk bejárása során.

Fenntartható HTML-kód készítése

Ha valaha is foglalkoztunk programozással, tudhatjuk, hogy milyen fontos, hogy fenntartható kódot készítsünk, vagyis törekedjünk arra, hogy ha később másvalaki belenéz, ne érezze magát teljesen elveszve. A kihívás tehát abban áll, hogy a kódunkat első látásra minél érthetőbbé tegyük. Gondoljunk arra, hogy eljöhet majd az idő, amikor visszanézünk egy saját magunk készítette weboldalra, és fogalmunk sem lesz, mi járt a fejünkben, amikor a kódot éppen így írtuk meg. Szerencsére megvan a módja annak, hogy miként vehetjük fel a harcot az ilyen memóriazavarokkal.

A kód értelmezése megjegyzésekkel



A JavaScript-parancsfájlokban belül a megjegyzéssorok elején a `//` karaktereket kell feltüntetnünk. (Ez esetben záró címkékre nincs szükség.) A stíluslapokon a megjegyzések előtt a `/*`, utánuk pedig a `*/` karaktereket kell elhelyeznünk.

A HTML `<!--` és `-->` címkéi nem működnek megfelelően a parancsfájlokban, illetve a stíluslapokon. Ugyanakkor fontos, hogy elhelyezzünk egy `<!--` címkét a `<script>` és a `<style>` elemek után, valamint egy `-->` címkét a megfelelő `</script>` és a `</style>` elemek előtt. Ezzel elrejtjük a parancsfájl, illetve a stílusok parancsait a régebbi böngészők előtt, amelyek egyébként egyszerű szöveggént értelmeznék és megjelenítenék azokat az oldalon.

Amikor csak egy HTML-oldal fejlesztésébe fogunk, tartsuk észben, hogy egyszer a jövőben valakinek szinte bizonyosan módosítania kell. Az egyszerű, szöveges oldalak olvasása és értelmezése többnyire nem okoz nehézséget, de az összetettebb oldalak, amelyek grafikát, táblázatokat és más szerkesztési trükköket alkalmaznak, komoly kihívás elé állíthatják a szemlélőt.

Ahhoz, hogy jobban megértsük a leírtakat, nyissunk meg szinte bármilyen weblapot a böngészőnkben. Ha az Internet Explorert használjuk, válasszuk a View (Nézet) menü Source (Forrás) pontját, a Firefoxban pedig szintén a View menü Page Source (Oldal forrása) pontját. Jó eséllyel egy kusza kódhalmazt fogunk látni, amelyről még azt is nehéz elhinni, hogy egyáltalán HTML-ben készült. Ez eredhet abból, hogy a kódot valamilyen tartalomkezelő rendszer hozta létre dinamikusan, de az is megeshet, hogy a programozó nem ügyelt a megfelelő kódszerkezetre, az olvashatóságra, a megjegyzések használatára, és egyáltalán, nem gondolt arra, hogy mások számára is érthetővé tegye a munkáját. Saját weboldalaink készítésénél ezért érdemes kissé jobban ügyelni a kód olvashatóságára és rendezettségére.

Amint a korábbi órákon láthattuk, a magunk, illetve mások számára írt megjegyzéseket a `<!--` és `-->` címkék között helyezhetjük el. Az itt feltüntetett szöveg nem jelenik meg a böngészőben, de bárki számára láthatóvá válik, aki a HTML-kódot egy szövegszerkesztővel vagy a böngésző „Oldal forrása” parancsával nyitja meg. Az alábbi rövid kód segít az emlékezetünkbe idézni, hogy miként is formázhatjuk a megjegyzéseket:

```
<!-- Ezt a képet naponta frissíteni kell. -->

```

Az `` elem előtt elhelyezett megjegyzés jelzi, hogyan használjuk a képet. Akárki is olvassa a kódot, nyomban világossá válik a számára, hogy a képet naponta frissíteni kell. A megjegyzésben álló szöveget a böngésző teljes egészében figyelmen kívül hagyja.



A megjegyzések nagyszerű szolgálatot tehetnek egy oldal azon részeinek elrejtésében is, amelyek jelenleg még nem készültek el. Így ahelyett, hogy a félkész szöveget és grafikát megjelenítve magyarázkodnánk, hogy még nem készültünk el a munkával, időlegesen elrejthetjük őket a megfelelő HTML-kódrészletek köré helyezett nyitó- és zárócímekkel. Így fokozatosan készíthetjük el a weboldal részeit, és a készültségüknek megfelelően, lépésről lépésre tehetjük azokat közzé.



Önálló feladat

A kód kiegészítése megjegyzésekkel

Érdemes lehet most némi időt szentelnünk annak, hogy sorra vegyük az eddigiekben elkészített weboldalakat, parancsfájlokat és stíluslapokat, és kiegészítsük azokat megjegyzésekkel, amelyek a későbbiekben hasznosak lehetnek saját magunk és mások számára. Hajtsuk végre az alábbi lépéseket:

1. Helyezzünk megjegyzést minden különleges oldalszerkezeti vagy formázási megoldás elé.
2. Alkalmazzunk megjegyzést minden olyan `` elem előtt, amelynek a szerepe lényeges, de ezt az `alt` szöveg nem írja le megfelelően.
3. Érdemes egy (vagy akár több) megjegyzéssel összefoglalnunk, hogyan kapcsolódnak egymáshoz képileg a cellák egy `<table>` elemen belül.
4. Ha hexadecimális színkódokat használunk (például: `<div style="color: #8040B0">`), jelezzük egy megjegyzéssel, hogy milyen színről van szó (kékeslila).
5. Lássuk el a megjegyzéseket behúzásokkal, így kiemelhetjük őket a kódból, és könnyebbé válik mind a megjegyzések, mind a HTML-kód olvasása. Ne feledkezzünk meg a HTML-kód behúzásairól sem, hiszen ezek olvashatóbbá teszik a kódot – erről a következőkben még szólnunk.

Behúzások az érthetőségért

Tartozunk egy vallomással. A könyv eddigi részében anélkül vezettük egyre mélyebbre az Olvasót egy HTML-kódolási stílusba, hogy erről egy szót is szoltunk volna. Most itt az ideje, hogy kiterítsük a lapjainkat. Minden bizonnyal feltűnt, hogy a könyvben szereplő HTML-kódok behúzásaiban észlelhető némi hasonlóság. Egészen pontosan arról van szó, hogy a gyermekelemek a szülőjükhöz képest két szóközzel jobbra kezdődnek. Mindemellet, amennyiben az elem tartalma egynél több sorra terjed ki, ez is behúzást kap az elemen belül.

A behúzások valódi értékét úgy mérhetjük fel a legegyszerűbben, ha megnézzük, hogyan is fest a HTML-kód nélkülük. Valahogy így:

```
<table>
<tr><td>Cell One</td><td>Cell Two</td></tr>
<tr><td>Cell Three</td><td>Cell Four</td></tr>
</table>
```

Itt a behúzások hiánya mellett a táblázat sorai és oszlopai sem válnak el egymástól. Hasonlítsuk össze a fentieket a következő kóddal, amely ugyanezt a táblázatot írja le:

```
<table>
  <tr>
    <td>Cell One</td>
    <td>Cell Two</td>
  </tr>
  <tr>
    <td>Cell Three</td>
    <td>Cell Four</td>
  </tr>
</table>
```

Ez utóbbi, behúzásokkal bőségesen ellátott kódban világosan látszik, hogyan választják el a sorokat és az oszlopokat a `<tr>` és `<td>` elemek.



Ha másokkal együtt dolgozunk, vagy közös munkát tervezünk, érdemes együtt leülni, és közösen kialakítani egy kódolási szabályzatot. Így biztosak lehetünk benne, hogy mindenki ugyanazon a (web)oldalon áll.

A behúzások összehangolt rendje talán még a megjegyzéseknél is fontosabb szerepet kaphat abban, hogy érthetővé és fenntarthatóvá tegyük a HTML-kódunkat. Ráadásul nem kell feltétlenül elfogadnunk a könyvben használt módszert. Ha kettő helyett három vagy négy szóközt szeretnénk használni, nincs semmi gond. Akkor sem követünk el hibát, ha kicsit tömörebbre szeretnénk venni a kódot, és az elemeken belüli tartalmat nem húzzuk be. A legfontosabb, hogy kialakítsunk egy kódolási stílust, és a továbbiakban fegyelmezetten tartuk hozzá magunkat.

Összefoglalás

Ezen az órán elméletben és néhány példa kapcsán bemutattuk, miként szervezhetjük a weboldalainkat egységes webhellyé, amely egyszerre tájékoztat, kedves a szemnek, és könnyen bejárható. Manapság a webes felhasználók meglehetősen érzékenyek arra, hogy jól felépített webhelyet kapjanak, így hamar elhagyják a webhelyünket, ha a szerkezete nem teszi lehetővé az egyszerű navigálást.

Emellett tanultunk a HTML-kódban elhelyezett behúzások és megjegyzések fontosságáról is, amelyek megkönnyítik weboldalaink karbantartását. A megjegyzések használatában világossá válik, amint belegondolunk abba, hogy később is visszatérhetünk egy régebben készített kódhoz, ráadásul így mások számára is érthetővé tehetjük az elgondolásainkat.

A behúzások használata első ránézésre esztétikai szempontnak tűnhet, de egyáltalán nem az, hiszen ezzel elérhetjük, hogy a weboldal szerkezete könnyen áttekinthetővé váljon.

Kérdezz-felelek

- K:** *Léteznek olyan weboldalak, amelyek arra kéri a látogatókat, hogy változtassák meg a böngészőablak szélességét, vagy módosítsanak más beállításokat, mielőtt továbblépnek. Mi ennek az oka?*
- V:** Nos, ha durván akarnánk fogalmazni, azt mondhatnánk, hogy ezeknek a webhelyeknek a készítői nem igazán törődnek a felhasználókkal. Soha ne kényszerítsük a felhasználókat arra, hogy valamit máshogy tegyenek a böngészőjünkben, mint addig, és különösképpen soha ne méretezzük át a böngészőablakot automatikusan. Ez a használhatóság szabályainak legsúlyosabb megsértése. Amikor egyes webhelyek a beállítások módosítására kérnek fel, emögött a készítőik jó szándéka áll, miszerint a meghatározott ablakméret vagy betűtípus választása az oldal előnyösebb megjelenítését teszi lehetővé. Persze kevesen vannak, akik eleget tesznek a kérésnek, és módosítanak a beállításaikon (miért is tennék?), így ezek a webhelyek többnyire furcsa és olvashatatlan tartalmat adnak. Sokkal jobban járunk, ha megszívleljük a könyvben leírtakat, és mindenféle ablakméret- és böngészőbeállítás mellett megkíséréljük olvashatóvá és vonzóvá tenni a weboldalainkat. Fontos megértenünk, hogy minél jobban szerkesztjük meg a webhelyünket, annál használhatóbbnak tartják majd azokat a látogatóink.
- K:** *Előfordulhat, hogy a rengeteg megjegyzés és szóköz lassítja az oldalak letöltését?*
- V:** Az a kevés fölös szöveg elhanyagolható mennyiségű adat más, nagyobb méretű erőforrásokhoz (például képekhez és egyéb multimédiás tartalmakhoz) viszonyítva. Mindemellett a lassabb, betárcsázós kapcsolatok tömörítve viszik át a szöveges adatokat, így a formázáshoz szükséges szóközők jobbára egyáltalán nem csökkentik a sebességet. Több száz szavas megjegyzésre van szükség ahhoz, hogy akár egyetlen másodpercnyi késlekedést okozzunk az oldal betöltésében. Ha ez nem lenne elég, ne feledjük, hogy manapság a legtöbb felhasználó szélessávú (kábel-, DSL- stb.) kapcsolattal rendelkezik, ahol a szöveg átvitele igen gyors. Az oldalak betöltését a grafikai elemek lassíthatják le, így ezeknek a tömörítésén érdemes ügyködnünk (lásd a 10. fejezetben leírtakat) – a szöveges megjegyzésekkel nem kell törődnünk.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Milyen három módszer ismeretes arra, hogy a weboldalainkból egységes webhelyet állítsunk össze?
2. Melyik az a két adattípus, amelyet feltétlenül el kell helyoznünk a kezdőoldalon?
3. Szeretnénk a következő üzenetet eljuttatni a weboldalunk későbbi szerkesztőinek anélkül, hogy ez a felhasználók előtt is megjelenjen: „Don't change this image of me. It's my only chance at immortality” (Ne cseréld ki ezt a rólam készült képet. Ez az egyetlen esélyem a halhatatlanságra.) Hogyan tennénk meg ezt?

Válaszok

1. Használjunk egységes hátteret, színeket, betűtípusokat és stílusokat. A hivatkozások szövegeit, illetve grafikáit ismételjük meg azoknak az oldalaknak a tetején, amelyekre utalnak. Ismételjük ugyanazt a fejléctet, gombokat, vagy más jellemző elemeket a webhely minden oldalán.
2. Adjunk elegendő tájékoztatást magunkról, hogy a látogató azonnal felmérhesse, mi a webhely neve, és miről is szól pontosan. Emellett, akármilyen is az üzenetünk, amit a közönségünknek tolmácsolni szeretnénk, közöljük tömören és egyszerűen. Legyen ez mottó vagy jogvédett reklámszöveg, feltétlenül tüntessük itt fel.
3. Helyezzük el az alábbi megjegyzést közvetlenül az elem előtt:

```
<!-- Don't change this image of me.  
It's my only chance at immortality. -->
```

Gyakorlatok

- Fogjunk egy jó öreg grafitceruzát, és vázoljuk fel a webhelyünket kis téglalapokkal, majd kössük össze azokat nyilakkal. Firkantsuk le az egyes weboldalak szerkezetét úgy, hogy krumplikat rajzolunk a képek, és hullámvonalakat a szövegek helyére. Az egyes nyilak a navigációs elemek krumplijaiból induljanak, és a megfelelő weboldalhoz vezessenek. Ez a vázlat még akkor is hasznos lehet, ha egyébként rendelkezünk a legeslegújabb webhelykezelő eszközökkel, hiszen így azonnal láthatjuk, hogy mely oldalakat lehet könnyen elérni, és hogyan működik együtt a szomszédos oldalak szerkezete – ráadásul mindezt még azelőtt, hogy az oldalakat összekötő kódból egyetlen sort is megírtunk volna. Akár hisszük, akár nem, jómagam is ezt a módszert követem. Van úgy, hogy a papírnál és ceruzánál nincs jobb társ a munkában!
- Nyissuk meg a jelenlegi webhelyünket felépítő HTML-fájlokat, és tekintsük át a bennük található megjegyzéseket és behúzásokat. Találunk olyan részeket, amelyek esetleg magyarázatra szorulnak egy későbbi olvasó számára? Ha igen, helyezzünk el alkalmas megjegyzéseket. Nehezen áttekinthető a kód szerkezete? Nem látszanak jól a címsorok és a bekezdések? Ha igen, alkalmazzunk behúzásokat úgy, hogy a kód szerkezete megfeleljen a hierarchiának, így a későbbiekben könnyen megtalálhatjuk a szerkeszteni kívánt részt.



24. ÓRA

Segítség a kereséshez

A lecke tartalma:

- Hogyan népszerűsíthetjük a webhelyünket?
- Hogyan érhetjük el, hogy az ismert keresők feltüntessék a webhelyünket a találdataik között?
- Hogyan optimalizálhatjuk a webhelyünket a keresők számára?

Egy webhely éppen annyit ér, amennyire megközelíthető – ha senki találja meg az oldalainkat, hiába készítettünk nagyszerű szerkezetet, gazdag tartalmat és tökéletes kódot. A következőkben bemutatott új HTML-elemek nem eredményeznek látható változást a weboldalaink felületén, de igen lényeges szerepet töltenek be abban, hogy a közönségünk valóban ráakadjon arra, amit kínálunk. A legtöbb webfejlesztő számára ez lehet könyvünk legegyszerűbb, mégis talán legfontosabb órája. Megtanuljuk, milyen újabb elemeket helyezünk el az oldalainkon, és miként módosítsuk a webhelyünk szerkezetét ahhoz, hogy a keresők nagyobb valószínűséggel jelezzenek nálunk találatot, ha valaki a webhelyünk témájához vagy a cégünkhöz kötődő kulcsszavakat alkalmaz – ez a *keresésoptimalizálás* (search engine optimization, SEO). Az ilyen szolgáltatásokat kínáló cégek reklámszövegével ellentétben semmiféle trükk nem ismeretes, amely biztosítaná, hogy a találati listák élére kerüljünk. Mindazonáltal létezik pár ügyes fogás, ami pénzbe sem kerül, ugyanakkor érdemes élnünk vele, ha azt szeretnénk, hogy a webhelyünk előkelő helyezést érjen el a keresők találdatai között.

A webhelyünk népszerűsítése

Feltételezhetően azért készítettük el a webhelyünket, hogy valakinek felkeltsük az érdeklődését, hiszen egyébként miért is fáradtunk volna? Ha a weboldalainkat helyi hálózaton vagy egy cég belső hálózatán tesszük közzé, esetleg valamilyen adathordozón terjesztjük, könnyen elérhetjük, hogy a felhasználók rátaláljanak az általunk kínált tartalomra. Ha azonban a webhelyet a keresők által indexelt milliárdnyi weboldal közé eresztjük, a közönség megnyerése máris nehezebb feladatnak tűnik.

Ahhoz, hogy belefogjunk a kérdés kezelésébe, először is alapjaiban meg kell értenünk, hogy miképpen akadhatnak rá a webhelyünkre a látogatók. Lényegében három módon juthatnak el hozzánk:

- Valaki mesél nekik a webhelyről, és megadja a címét, amelyet ők közvetlenül írnak be a böngészőbe.
- Egy másik webhely hivatkozására kattintva jutnak el a mi webhelyünkre.
- A webhelyünk egy keresőmotor (a Google, a Bing, a Yahoo! vagy más kereső) adatbázisából kerül a látogatónk találati listájára,

Kis idő- és energiabefektetéssel magunk is lendíthetünk a webhely forgalmán. Ha azt szeretnénk, hogy többen értesüljenek rólunk szájhagyomány útján, nem kell mást tennünk, mint használni a szánkat – és minden más elérhető közlési csatornát. Ha rendelkezünk kapcsolati adatbázissal vagy levelezőlistával, híreszteljük ott a webhelyünk megjelenését. Tüntessük fel a webhely címét a névjegykártyánkon, illetve céges dokumentumainkban. Ha van némi pénzünk, vásároljunk reklámidőt a TV-ben és a rádióban. Röviden, törjünk előre a reklámfronton. A jó öreg szájhagyomány még mindig a leghatékonyabb reklámozási mód – még az Interneten is.

A ránk mutató hivatkozások számának növelése nem igényel nagy agymunkát – jöllehet ez nem jelenti azt, hogy egyszerű feladat lenne. Ha a témakörünkben léteznek akár elektronikus, akár nyomtatott szaknévsorok, nekünk is ott a helyünk! Vegyünk részt a közösségi hálózatok életében, hozzunk létre a Facebook-on „rajongói” oldalt, amennyiben ez illik a webhelyünk mondanivalójához. Hozzunk létre Twitter-felhasználót a webhelyünk számára, és így is lépünk kapcsolatba az ügyfeleinkkel – persze ismét csak akkor, ha ez a módszer nem ütközik a webhelyünk stílusával. Keressük fel azokat a helyeket, ahol ügyfelekre találhatunk, olyan webnaplókat, amelyek a témánkba vágnak, és vegyünk részt ezeknek a közösségeknek az életében. Ez nem azt jelenti, hogy keressünk egy fórumot a kérdéses témakörben, és árásszuk el a webhelyünk hivatkozásaival. Inkább lépünk fel a téma szakértőjeként, tanácsokat és ajánlásokat adva, miközben pedig a saját webhelyünk címére hivatkozva. E könyv keretei között erről a témáról nem igazán mondhatunk többet a bátorító szavakon kívül.



A Mashable nevű (<http://www.mashable.com/>) igen népszerű, nagy forgalmú és széles körben elfogadott webhely (mindez a pontosságának és a komoly hozzáadott értéknek köszönhető) komoly segítséget nyújthat abban, hogy sikerrel vegyünk részt a közösségi hálózatok életében, különösen ha ezt üzleti céllal tesszük.

Könyvünk leginkább a harmadik pontban, vagyis annak biztosításában segíthet, hogy a tartalmunkra rábukkannak a keresőprogramok, és megfelelő módon indexelik. Jogosan feltételezhetjük, hogy ha a webhelyünk kimarad a Google adatbázisaiból, bizony komoly bajban vagyunk.

A keresők alapjában véve hatalmas adatbázisok, amelyek igyekeznek minden, az Interneten fellelhető tartalmat indexelni – köztük videókat és egyéb gazdag médiaanyagokat. A webhelyek bejárására és az adatbázisok felépítésére automatikus eljárásokat alkalmaznak – ezeket hívjuk keresőrobotoknak (bot). Ha a tartalom indexelése megtörtént, a keresőalkalmazások különféle kifinomult módszerekkel rangsorolják az oldalakat annak érdekében, hogy eldöntsék, egy felhasználói keresés nyomán melyik találat álljon az első, a második, a harmadik helyen, és így tovább.

Amikor a kereső feldolgozza egy felhasználó kérését, olyan weboldalakokat keres, amelyek tartalmazzák a megadott kulcsszavakat és kifejezéseket. A keresés azonban nem annyiból áll, hogy „ha egy oldal tartalmazza a keresett kifejezést, adjuk vissza az eredményben”, ugyanis a tartalom rangsorolt, aszerint, hogy milyen gyakorisággal és milyen környezetben fordulnak elő benne a keresett kulcsszavak és kifejezések, továbbá milyen más webhelyek hivatkoznak rá, hitelesítve a benne foglaltakat. Ezen az órán megtanulunk pár módszert annak biztosítására, hogy a tartalmunk megjelenjen a keresők találatai között – mégpedig a megadott tartalom és környezet függvényében.

Weboldalaink felvétele a nagyobb keresők adatbázisaiba

Ha azt szeretnénk, hogy a felhasználók megtalálják az oldalainkat, alapvető fontosságú, hogy kérelmezzük a nagyobb keresőoldalaknál az oldalaink indexelését. Jóllehet a keresők automatikusan is keresztül-kasul bejárják a Webet, ez a legjobb módszer arra, hogy biztosan felkerüljünk a listájukra. E webhelyek mindegyikénél meg kell adnunk az URL-címünket, a webhely rövid leírását, és egyes esetekben egy kategóriát vagy kulcsszavak listáját, amelyek a webhelyünkhöz köthetők. A kapott űrlapok kitöltése egyszerű, egy órán belül végzünk az összessel, és még arra is marad időnk, hogy bejegyezzük magunkat pár, a témakörünkhöz kapcsolódó szaknévsorba. (Hogy ezekre miként akadhatunk rá? Természetesen a nagyobb keresők segítségével!)

Mielőtt bejegyeztetnénk az oldalainkat

Várjunk csak egy pillanatra! Mielőtt beadnánk a kérelmeinket a webhelyünk feltüntetésére, olvassuk végig az óra teljes anyagát, egyébként komoly csávába kerülhetünk, és mindössze azt állapíthatjuk meg szomorúan, hogy lecsúsztunk az egyszerű megoldásokról.

Hogy értsük, miről is van szó, képzeljük el a következő helyzetet: közzéteszünk egy webhelyet, amelyen automatikus csótányroppantókat kínálunk. Íme egy lehetséges ügyfél: egy internet-felhasználó, aki küzd a csótányokkal, ráadásul allergiás a rovarirtó vegyszerekre. Emberünk kinyitja a laptopját, lesöpri a csótányokat a billentyűzetről, belép a kedvenc keresőoldalára, és beírja a „csótány” szót a keresőmezőbe. A kereső pillanatokon belül kiírja az első 10 találatot – a 10 254 weboldalból, amely tartalmazza a „csótány” szót. Benyújtottuk a kérelmünket a webhelyünk bejegyzésére, így tudhatjuk, hogy ott lesz valahol a listán.

Lehetséges ügyfelünk ráadásul gazdag ember, és már nem bír a csótányokkal. Mi pedig még házhozzállítást is vállalunk a lakhelye környezetében. Szeretnénk, ha a webhelyünk harmadikként jelenne meg a találati listában, vagy beérjük a 8542. hellyel? Nos, ennyi minden bizonyítva elegendő a helyzet megvilágítására. Bekerülni egy kereső adatbázisába még félsikernek sem nevezhető – fel is kell küzdenünk magunkat a találati rangsor elejére.



Léteznek olyan webhelyek, amelyek egyetlen űrlapot biztosítanak, és az ezen megadott adatokat automatikusan elküldik az összes nagyobb és számos kisebb keresőnek.

Ezek a webhelyek – mint a <http://www.scrubtheweb.com/>, a <http://www.submitexpress.com/> és a <http://www.hypersubmit.com/> – nagyszerűen példázák, miként lehet üzletet csinálni webhelyek bejegyzéséből a különböző adatbázisokba. A célközönségunktől függően ez a szolgáltatás akár kifizetődő is lehet, de mindenképpen azt ajánlanánk, hogy látogassunk el közvetlenül a nagyobb keresők oldalaira (a felsorolásukat lásd az óra korábbi részében), és töltsük ki a bejegyzési űrlapjaikat. Így biztosak lehetünk benne, hogy megfelelő válaszokat adtunk a feltett kérdésekre (amelyek mindenütt kicsit eltérőek), és pontosan tudhatjuk, hogy milyen alakban jelenik meg a webhelyünk az egyes keresők találati listáján.

A webhelyünk felvétele a nagyobb keresők adatbázisaiba egyszerű, mégis tartogathat néhány buktatót. Mindegyik keresőmotor másképpen jelzi, hová kell kattintanunk az oldalaink bejegyzéséhez. Az alábbi lista megkímélhet némi bosszúságtól – itt megtalálhatjuk az ismertebb keresők ingyenes bejegyzési címét, valamint annak a hivatkozásnak a megnevezését, amelyre a bejegyzéshez kattintanunk kell:

- Google – Keressük fel a <http://www.google.com/addurl/> címet, adjuk meg a webhelyünk címét és rövid leírását, majd írjuk be a hullámos ellenőrző szöveget, vagyis a CAPTCHA-t (Completely Automated Public Turing test to tell Computers and Humans Apart, vagyis „teljesen automatizált nyilvános Turing-teszt a számítógép és az ember megkülönböztetésére”). Végezetül kattintsunk az Add URL (URL felvétele) gombra a webhely bejegyzéséhez.
- Yahoo! Search – Látogassunk el a <http://siteexplorer.search.yahoo.com/submit> címre, írjuk be a webhelyünk címét, majd kattintsunk a Submit URL (URL benyújtása) gombra.
- Bing – Keressük fel a <http://www.bing.com/docs/submit.aspx> címet, írjuk be az ellenőrző szöveget, adjuk meg a webhelyünk címét, majd kattintsunk a Submit URL gombra.
- AllTheWeb – Az AllTheWeb keresési eredményeit a Yahoo! Search biztosítja, így csak annyi a feladatunk, hogy a fentieknek megfelelően bejegyeztessük a webhelyünket a Yahoo! Search adatbázisába.
- AltaVista – Az AltaVista keresési eredményeit is a Yahoo! Search adja, így ha ott bejegyeztetjük a webhelyünket, itt is megjelenik a találati listában.

Tippek a keresőknek

El kell árulnunk egy szomorú tény: az égvilágon semmit sem tehetünk annak érdekében, hogy előkelő helyet biztosítsunk magunknak egy keresőkifejezésre adott találati listában, bármelyik keresőről legyen is szó (már ha nem vásárolunk közvetlenül hirdetési felületet). Végül is érthető, hiszen ha lennének ilyen biztosítékok, miért ne élne velük mindenki, aki csak felkerült a listára? Arra azonban törekedhetünk, hogy ne mi legyünk az utolsók, és éppen annyi esélyünk legyen az első helyre, mint a lista többi szereplőjének. Ezzel elérkeztünk a keresésoptimalizálás (SEO) témaköréhez – vagyis megpróbáljuk úgy átalakítani a weboldalaink tartalmát és szerkezetét, hogy a keresők előnyben részesítsék azokat a versenytársaikkal szemben.

Minden kereső más módszerrel határozza meg, hogy mely oldalak lehetnek a legmegfelelőbbek, és eszerint helyezik azokat feljebb vagy lejjebb a találati rangsorban. Nem érdemes azonban leragadni ezeknél a különbségeknél, ugyanis mindannyian ugyanazokra az alapvető szempontokra építenek. A következő felsorolás szinte minden olyan szempontot tartalmaz, amelyet a keresők figyelembe vesznek annak megállapításánál, hogy mely oldalak felelnek meg leginkább a keresőkifejezésnek:

- Megjelennek a kulcsszavak az oldal `<title>` elemén belül?
- Megjelennek a kulcsszavak az oldal első néhány sorában?
- Megjelennek a kulcsszavak az oldal egy `<meta />` elemén belül?
- Megjelennek a kulcsszavak az oldal `<h1>` címsoraiban?
- Megjelennek a kulcsszavak a weboldalon szereplő képek nevében vagy azok alt jellemzőjében?

- A webhely hány oldala hivatkozik a szóban forgó oldalra?
- Hány hivatkozás mutat az oldalunkra más webhelyekről? Hány további oldal hivatkozik ezekre a hivatkozó oldalakra?
- Hányszor választották a felhasználók az oldalt a korábbi keresések találati listájából?



Egyes, túlságosan is buzgó webprogramozók egy kulcsszó több tucat vagy akár több száz példányát is elhelyezik egy oldalon, igen apró betűvel vagy csaknem láthatatlan színnel, csak hogy magasabbra helyezték magukat a keresőknek az adott kulcsszóra vonatkozó találati rangsorában. Ezt a módszert nevezik a keresőmotorok elárasztásának (search engine spamming).

Akármilyen vonzónak is tűnik ez a lehetőség, nem érdemes kísértésbe esnünk, ugyanis a komolyabb keresők ismerik ezt a trükköt, és azonnal törölnek minden olyan oldalt az adatbázisaikból, amelynél a „szeméjtjelző” bekapcsol valamilyen gyanús mintázat szerint ismétlődő szó vagy szövegrészlet kapcsán. Az természetesen megengedett (és hasznos), ha a fontosabb kulcsszavakat többször is feltüntetjük az oldalunkon – a tartalom rendes részeként. Ügyeljünk azonban arra, hogy ezeket a szavakat hétköznapi mondatokban vagy kifejezésekben használjuk, így megkímélhetjük magunkat a „szeméttrendőrség” zaklatásától.



A `<meta />` elemeket mindig a `<head>`, valamint a `<title>` és a `</title>` elemek után, de a záró `</head>` címke elé helyezzük.

Az XHTML szabványai szerint bármely dokumentumról is legyen szó, a `<title>` elemnek az első helyen kell állnia a `<head>` részben.

A helyezésünkön azzal javíthatunk leginkább, ha átgondoljuk, hogy milyen kulcsszavakat használhat a reménybeli közönségünk a kereséshez. A gyakran használt, egyszavas keresőkifejezések csatamezejéről érdemes idejekorán kihátrálnunk, hiszen egy olyan keresőszó, mint a „étel”, rendszerint annyi találatot ad, hogy nagyobb esélyünk van nyerni a lottón, mint itt bejutni az első tíz közé. Összpontosítsunk inkább a ritkább szavakra, valamint a két-három szavas kifejezésekre, amelyek jobban illenek a témakörünkhöz – így például az „étel” helyett jobb választás a „házas déli ételek”. Ha összegyűjtöttük ezeket a kifejezéseket, gondoskodjunk róla, hogy többször is felbukkanjanak az oldalunkon, a legfontosabbakat pedig helyezzük el a `<title>` elemen belül, valamint az első címsorban, illetve a bevezető bekezdésben.

A fenti listában felsorolt szempontok közül talán a `<meta />` elemek használatát övezi a legnagyobb homály. Vannak, akik olyan tisztelettel viseltetnek irántuk, mintha a használatuk azonnal a rangsor elejére repítené az oldalukat, mások viszont lesajnálóan nyilatkoznak róluk, feleslegesnek és haszontalannak tartva őket. Az igazság, mint oly sokszor, most is a két szélsőség között van.

A `<meta />` általános rendeltetésű elem, amelyet bármely dokumentum `<head>` részében elhelyezhetünk, ha olyan adatokat szeretnénk feltüntetni, amelyek a `<body>` szövegében nem kapnak helyet. A legtöbb kereső a `<meta />` elemeket azért vizsgálja meg, hogy rövid összefoglalót kapjon az adott oldalról, néhány fontosabb kulcsszóval egyetemben. Az automatikus csótányroppantó megrendelő űrlapja esetében a következő két elemet tüntethetnénk fel:

```
<meta name="description"
content="Order form for the SuperSquish cockroach flattener." />
<meta name="keywords"
content="cockroach, roaches, kill, squish, supersquish" />
```



Ha eltekintünk az előző, csótányroppantós példától, és a keresők szakértőit kérdezzük, megtudhatjuk, hogy a `<meta />` elemben elhelyezett leírás ideális méretét valahová 100 és 200 karakter közé helyezik, a kulcsszavak listájának ideális hossza pedig 200 és 400 karakter közé tehető. A szakértők arra is felhívják a figyelmet, hogy nem érdemes szöközőkre vesztegetnünk a drága biteket – ennek megfelelően jártunk el a csótányos példában is. Végezetül pedig, nem érdemes ugyanazokat a kulcsszavakat különböző kifejezésekben nyaklóló nélkül ismételgetnünk, egyes keresőknél ugyanis ez büntetést von maga után.



Ha valamilyen rejtélyes okból nem szeretnénk, hogy egy adott weboldal bekerüljön a kereső adatbázisába, az alábbi `<meta />` elemet helyezhetjük el az oldal `<head>` részében:

```
<meta name="robots" content="noindex, noindex" />
```

Ennek hatására egyes keresőrobotok elkerülik az oldalt. Ha komolyabb védelmet szeretnénk a kutakodó robotok ellen, kérjük meg a webkiszolgálónk rendszergazdáját, hogy helyezze el az oldal címét a kiszolgáló `robots.txt` nevű fájljában. (Tudni fogja, mit jelent ez, és hogyan kell csinálni – ha mégsem, hasznos segítséget kaphatunk a <http://www.robotstxt.org/> címen.) Ez esetben biztosak lehetünk benne, hogy a nagyobb keresők robotjai elkerülik a kérdéses oldalt. Ez a módszer jól alkalmazható belső, céges oldalakra, amelyeket nem feltétlenül szeretnénk viszontlátni a keresők találatai között.

A példában szereplő első elem biztosítja, hogy a találatok listájában megfelelő leírás jelenjen meg a weboldalról. A második `<meta />` elem némiképp előkelőbb helyre löki az oldalt minden olyan találati rangsorban, ahol a keresőkifejezések között előfordult a felsorolás valamelyik eleme.

Fontos, hogy minden olyan oldal esetében, amelyet a keresők figyelmébe szeretnénk ajánlani, tüntessünk fel `<meta />` elemeket a `name="description"` és a `name="keywords"` jellemzőkkel. Ez nem feltétlenül eredményez ugrásszerű javulást az oldalaink megítélésében, ráadásul egyes keresők figyelmen kívül hagyják a `<meta />` elemeket – ártani azonban semmiképpen sem árt.

A keresők eredményeinek javítási lehetőségeit a 24.1. példában mutatjuk be.

24.1. példa *Weboldal, amely kevés nyilvánosságot kap az internetes keresések során*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Fractal Central</title>
  </head>

  <body>
    <div style="text-align:center">
      
    </div>
    <div style="width:133px; float:left; padding:6px; text-align:center;
border-width:4px; border-style:ridge">
      Discover the latest software, books and more at our online
      ➡ store.<br />
      <a href="orderform.html"></a>
    </div>
    <div style="float:left; padding:6px">
      <h2>A Comprehensive Guide to the<br />
      Art and Science of Chaos and Complexity</h2>
      <p>What's that? You say you're hearing about "fractals" and
      "chaos" all over the place, but still aren't too sure what they
      are? How about a quick summary of some key concepts:</p>
      <ol>
        <li><p>Even the simplest systems become deeply complex and richly
        beautiful when a process is "iterated" over and over, using the
        results of each step as the starting point of the next. This is
        how Nature creates a magnificently detailed 300-foot redwood tree
        from a seed the size of your fingernail.</p></li>
        <li><p>Most "iterated systems" are easily simulated on computers,
        but only a few are predictable and controllable. Why? Because a
        tiny influence, like a "butterfly flapping its wings," can be
        strangely amplified to have major consequences such as completely
        changing tomorrow's weather in a distant part of the
        world.</p></li>
        <li><p>Fractals can be magnified forever without loss of detail,
        so mathematics that relies on straight lines is useless with
        them. However, they give us a new concept called "fractal
        dimension" which can measure the texture and complexity of
        anything from coastlines to storm clouds.</p></li>
        <li><p>While fractals win prizes at graphics shows, their chaotic
        patterns pop up in every branch of science. Physicists find
        beautiful artwork coming out of their plotters. "Strange
        attractors" with fractal turbulence appear in celestial mechanics.
        Biologists diagnose "dynamical diseases" when fractal rhythms fall
```

```

    out of sync. Even pure mathematicians go on tour with dazzling
    videos of their research.</p></li>
</ol>
<p>Think all these folks may be on to something?</p>
</div>
<div style="text-align:center">
  <a href="http://netletter.com/nonsense/"></a>
</div>
</body>
</html>

```

Ezt az oldalt elvileg igen könnyen meg kellene találnunk, hiszen egyetlen, meglehetősen behatárolt témával foglalkozik, és bizonyos szokatlan tudományos fogalmak, amelyek ugyanakkor a hozzáértők számára magától értetődően keresőkifejezések, többször is megjelennek benne. Mégis számos olyan területet találhatunk a kódban, ahol a keresés eredményessége növelhető.

Most pedig hasonlítsuk össze a fenti oldal kódját a 24.2. példában látható módosított változattal. A két oldal az emberi szemnek igen hasonló, a keresőrobotok számára azonban gépzongorázni lehet a különbséget. Az alábbiakban felsoroljuk a változtatásokat, bemutatva a hatásukat az indexelésre:

- Fontos keresőkifejezéseket helyeztünk el a `<title>` elemben és az oldal első címsorában. Az eredeti oldalon még a *fractal* szó sem tűnt fel ebben a két kulcspozícióban.
- `<meta />` elemekkel egészítettük ki a kódot, így a keresők közvetlenül hozzájuthatnak az oldal leírásához és kulcsszavaihoz.
- Igen beszédes `alt` jellemzővel egészítettük ki az első `` elemet. Jóllehet ezt a jellemzőt nem minden kereső figyeli, egyesek számára fontos jelentéssel bír.
- Eltávolítottuk a tudományos fogalmakat körülvevő idézőjeleket (mint a "fractal" és az "iterated" esetében), mivel bizonyos keresők a „*fractal*” és a *fractal* szavakat különbözőnek tekintik. Az idézőjeleket a `"` kóddal helyettesítettük – a keresőrobotok ezt egyszerűen figyelmen kívül hagyják. Ez már csak azért is üdvös lépés, mert az XHTML szabvány amúgy is felhív a `"` kód használatára az idézőjelek helyett.
- A *fractal* kulcsszót két példányban is elhelyeztük a megrendelési űrlap területén.

Lehetetlen számokba önteni, hogy mennyivel gyakrabban találnák meg a fraktálok és a káosz iránt érdeklődők a 24.2. példa oldalát, mint a 24.1. példában bemutatottat – azt azonban biztosan elmondhatjuk, hogy valamicskét javítottunk a helyzeten. A robotok igényeit szem előtt tartó átalakítások „mellékhatásaként” – mint az oly gyakran lenni szokott – az oldalunk az emberi szem számára is értelmezhetőbbé vált. A keresésoptimalizáláson tehát mindenki csak nyert.

24.2. példa A 24.1. példa javított változata

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Fractal Central: A Guide to Fractals, Chaos, and
      ➤ Complexity</title>
    <meta name="description" content="A comprehensive guide to fractal
      geometry, chaos science and complexity theory." />
    <meta name="keywords" content="fractal,fractals,chaos science,chaos
      theory,fractal geometry,complexity,complexity theory" />
  </head>

  <body>
    <div style="text-align:center">
      
    </div>
    <div style="width:133px; float:left; padding:6px; text-align:center;
      border-width:4px; border-style:ridge">
      Discover the latest fractal software, books and more at the
      <span style="font-weight:bold">Fractal Central</span> online
      ➤ store.<br />
      <a href="orderform.html"></a>
    </div>
    <div style="float:left; padding:6px">
      <h2>A Comprehensive Guide to Fractal Geometry,<br />
        Chaos Science, and Complexity Theory</h2>
      <p>What's that? You say you're hearing about &quot;fractals&quot;
        and &quot;chaos&quot;; all over the place, but still aren't too
        sure what they are? How about a quick summary of some key
        concepts:</p>
      <ol>
        <li><p>Even the simplest systems become deeply complex and richly
          beautiful when a process is &quot;iterated&quot;; over and over,
          using the results of each step as the starting point of the next.
          This is how Nature creates a magnificently detailed 300-foot
          redwood tree from a seed the size of your fingernail.</p></li>
        <li><p>Most &quot;iterated systems&quot;; are easily simulated on
          computers, but only a few are predictable and controllable. Why?
          Because a tiny influence, like a &quot;butterfly flapping its
          wings,&quot; can be strangely amplified to have major consequences
          such as completely changing tomorrow's weather in a distant part
          of the world.</p></li>
        <li><p>Fractals can be magnified forever without loss of detail,
          so mathematics that relies on straight lines is useless with them.
          However, they give us a new concept called &quot;fractal
          dimension&quot;; which can measure the texture and complexity of
          anything from coastlines to storm clouds.</p></li>
        <li><p>While fractals win prizes at graphics shows, their chaotic
          patterns pop up in every branch of science. Physicists find

```

```

    beautiful artwork coming out of their plotters. "Strange
    attractors" with fractal turbulence appear in celestial
    mechanics. Biologists diagnose "dynamical diseases" when
    fractal rhythms fall out of sync. Even pure mathematicians go on
    tour with dazzling videos of their research.</p></li>
</ol>
<p>Think all these folks may be on to something?</p>
</div>
<div style="text-align:center">
  <a href="http://netletter.com/nonsense/"></a>
</div>
</body>
</html>

```

A fenti módosítások nyomán sokkal valószínűbbé vált, hogy a keresők megfelelően indexelik majd az oldalunkat. A tartalom helyes indexelése mellett persze fontos a minősége is, valamint az, hogy hány egyéb webhely hivatkozik az oldalunkra.

A keresésoptimalizálás további fogásai

A legfontosabb tanács ezen a téren talán az lehet, hogy egy fillért se adjunk olyan cégeknek, amelyek az optimalizálás területén konkrét eredményekkel kecsegtetnek. Ha egy cég azt ígéri, hogy a Google találati rangsorában az első helyre tornássa fel a weboldalunkat, azonnal fordítsunk hátat, és fussunk minél messzebbre – illetéges senki sem ígérhet, hiszen a keresőalgoritmusok rengeteg változótól függenek, és a lista első helyezettje egyetlen hét alatt is sokszor megváltozhat. Ezzel persze nem azt akarjuk mondani, hogy a keresésoptimalizálással foglalkozó cégek minál csaliók. Léteznek valóban értékes munkát végző tanácsadók, akik segíthetnek a webhelyünk szerkezetének és tartalmának helyrehozásában, de ez a pár aprócska sziget elvész a kérértlen leveleket küldözgető csaliók tengereiben. Íme egy példa:

```

"Dear google.com,
I visited your website and noticed that you are not listed in most
of the major search engines and directories..."

```

Ez a levél (amely a google.com webhely tulajdonosát értesíti arról, hogy a webhelye nincs bejegyezve a nagyobb keresőknél) a Google saját útmutatásai között példaként szerepel a következő megjegyzéssel: „a keresőkkel kapcsolatos kérértlen elektronikus leveleket ugyanazzal a tartózkodással kezeljük, mint az éjszakai zsírógőz tabletta reklámjait vagy az elűzött diktátorok átutalási kérelmeit”. Igen, nem tévedés, valaki magának a Google-nek is küldött némi levélszemetet, miszerint szívesen javítaná a webhely rangját... a Google találati között! További jótanácsokat a <http://www.google.com/webmasters/> címen kaphatunk.

A következőkben felsorolunk néhány ingyenesen elvégezhető lépést, amelyek segítenek az oldalaink tartalmát a keresőmotorok lelkivilágához igazítani:

- Adjunk az oldalainknak kellően pontos címet, ami rövid, ugyanakkor beszédes és egyedi. Ne kíséreljük meg kulcsszavakkal teletömni a címeket.
- Alkalmazzunk az emberi szemnek barátságos URL-eket, azaz például használjunk bennük könnyen megjegyezhető szavakat. A `http://www.mycompany.com/products/super_widget.html` és hozzá hasonló címeket egyszerűbb megjegyezni (és a keresők is könnyebben indexelik őket értelmes módon), mint valami ilyesmit: `http://www.mycompany.com?c=p&id=4&id=49f8sd7345fea`.
- Hozzunk létre olyan URL-eket, amelyek tükrözik a webhelyünk könyvtárszerkezetét. Persze ez feltételezi, hogy rendelkezünk könyvtárszerkezettel (elég nagy baj, ha nem ez a helyzet).
- Ha csak lehetséges, a navigáció céljaira grafika helyett szöveges elemeket alkalmazzunk.
- Ha a webhelyünk tartalmának egyes részei több lépésre kerültek a kezdőoldaltól, segítsük a látogatóinkat morzsaútvonalakkal, hogy könnyen visszatálálhassanak a kiindulóponthoz. A morzsaútvonal egyúttal újabb szavakat szolgáltat az indexelésre a keresőrobotoknak. Így ha egy főzéssel kapcsolatos webhely Southern Cooking kategóriájában teasütemények receptjeit nézegetjük, az aktuális oldal morzsaútvonala ilyesmi lehet: Home>Southern Cooking>Recipes>Biscuits
- Az oldal tartalmában használjuk a rendeltetésüknek megfelelően a címsorokat (`<h1>`, `<h2>`, `<h3>`).

Amellett tehát, hogy a felhasználók számára gazdag és hasznos tartalmat biztosítunk, tartsuk magunkat a fenti útmutatásokhoz, így az oldalaink egyre feljebb kúsznak majd a találati rangsorban.

Összefoglalás

Ezen az órán egy igen fontos területre merészkedtünk – megtanultuk, miként „súgjunk” a keresőknek (amilyen a Google, a Bing vagy a Yahoo), hogy a reménybeli látogatóink könnyebben ráakadjanak az oldalainkra. Láthattuk azt is, hogy ami tökéletesen működőképes HTML-kódnak tűnik, hogyan optimalizálható úgy, hogy a keresőrobotok is felfedezzék a rejtett értékeit. Végezetül, gazdagabbak lettünk néhány útmutatással arra nézve, hogy miként tegyük keresőbaráttá a webhelyünk egészét. A 24.1. táblázatban összefoglalást adunk az óra során bemutatott elemekről és jellemzőkről.

24.1. táblázat A 24. órán bemutatott HTML-elemek és -jellemzők

Elem/Jellemző	Leírás
<code><meta /></code>	A dokumentumhoz kapcsolódó metaadatokat tartalmaz (vagyis olyan adatokat, amelyek magára a dokumentumra vonatkoznak). Leggyakrabban a dokumentum leírását, valamint a jellemző kulcsszavakat tároljuk benne. A dokumentum <code><head></code> részében kell feltüntetnünk.

Jellemzők

name="név"	Ezzel adhatjuk meg, hogy milyen típusú adatot tárolunk a content jellemzőben. Így például a name="keywords" arra utal, hogy a weboldal kulcsszavait találhatjuk meg ott.
content="érték"	A http-equiv, illetve a name jellemzőkhöz tartozó üzenet vagy érték. Így ha a http-equiv értéke refresh, a content jellemzőben a várakozás idejét kell feltüntetnünk másodpercben, majd egy pontosvesszőt követően a betölteni kívánt weboldal címét.

Kérdezz-felelek

- K: *Valóban külön űrlapot kell kitöltenünk a webhelyünk minden oldalához minden egyes keresőnél?*
- V: Nem. Elég a kezdőoldalt bejegyeztetnünk (ez feltehetően kapcsolatban áll a webhelyünk többi oldalával), hiszen a keresőrobotok úgyis bejárják az oldal hivatkozásait (majd a hivatkozott oldal hivatkozásait, és így tovább), így végül a webhelyünk összes lapját indexelik.
- K: *Elküldtem egy keresőmotorhoz a bejegyzési kérelmet, de ha az oldalamat keresem, nem jelenik meg a találati listában – még a cég egyedi nevét beírva sem. Mit tehetek?*
- V: A legtöbb nagy keresőmotor a rendelkezésünkre bocsát egy űrlapot, amelynek a segítségével ellenőrizhetjük, hogy a weboldalunk megtalálható-e a kereső adatbázisában. Ha nem találjuk, beküldhetünk egy újabb bejegyzési kérelmet. Vegyük figyelembe, hogy a bejegyzés kérelmezése után gyakran napokba, sőt hetekbe telik, míg a keresőrobotok bejárják a webhelyünket.
- K: *A <meta /> elemben érdemes feltüntetni a kulcsszavak minden írásmódját, beleértve a kis- és nagybetűs változatokat is?*
- V: A kis- és nagybetűkkel egyáltalán ne törődjünk – a keresőkifejezéseket a leggyakrabban csupa kisbetűvel adják meg. Fontos azonban, hogy feltüntessük a kulcsszavak ismert változatait, sőt azok gyakoribb elírásait is. A <meta /> elem működése egyszerűnek tűnhet, de az órán bemutatottnál lényegesen kifinomultabb módszerek is léteznek a használatára. Az elem jellemzőiről és alkalmazásáról nagyszerű tájékoztatást kaphatunk a http://en.wikipedia.org/wiki/Meta_element címen.
- K: *Valóban használhatjuk a <meta /> elemet arra, hogy automatikusan újra betöltsük a weboldalunkat néhány másodpercenként vagy pár percenként?*
- V: Igen, de ennek nincs sok értelme, hacsak nem rendelkezünk olyan programmal vagy parancsfájllal a webkiszolgálónkon, amely friss adatokat küld az oldalunknak. Ha pedig így áll a helyzet, jó eséllyel inkább az AJAX lehetőségeit használjuk a frissítésre (az ezzel kapcsolatos alapvető tudnivalókról a 21. órán ejtettünk szót). A <meta /> elem frissítési lehetőségét használhatósági okokból mind a W3C, mint a felhasználók helytelenítik.

Ismétlés

Ez a rész ismétlő kérdésekből és válaszokból áll, amelyek segítségével megszilárdíthatjuk az ebben a leckében szerzett tudásunkat. Próbáljuk megválaszolni a kérdéseket a válaszok ellenőrzése előtt.

Ismétlő kérdések

1. Ha kölyökkutyák örökbefogadásáról szóló oldalt teszünk közzé, mit érdemes tennünk, hogy minden nagyobb kereső találatai között megjelenjen a webhelyünk, ha valaki a *kölyök*, a *kutya*, illetve az *örökbefogadás* szavakkal hajt végre keresést?
2. Tegyük fel, hogy úgy döntünk, a kulcsszavainkat több száz példányban megjelenítjük a HTML-kódban, még hozzá fehér alapon fehér háttérrel, hogy a látogatóinknak ne tűnjön fel. Mit lépnek erre a keresőrobotok?
3. Melyik a jobb megoldás: ha a webhelyünk egész tartalmát egyetlen könyvtárba ömlesztjük, vagy ha több könyvtárba válogatjuk?

Válaszok

1. Mindenekelőtt győződjünk meg arról, hogy a *kölyök*, a *kutya* és az *örökbefogadás* szavak kellő számban előfordulnak a kezdőoldalon (ez valószínűleg így is van), és adjuk a weboldalunknak a *Kutyakölykök örökbefogadása nálunk!* címe, vagy valami hasonlót. Emellett helyezzük el az alábbi `<meta />` elemeket az oldal `<head>` részében:

```
<meta name="description"
content="dog adoption information and services" />
<meta name="keywords" content="puppy, dog, adoption" />
```

Tegyük közzé a weboldalunkat az Interneten, és jegyeztessük be a nagyobb keresőknél (a listát lásd a korábbiakban).

2. A keresőrobotok figyelmen kívül hagyják az ilyen ismétlődéseket, sőt még az is könnyen megeshet, hogy feketelistára tesznek, és kénytelen reklámterjesztőnek minősítenek.
3. Feltétlenül szervezzük a tartalmat könyvtárszerkezetbe. Ez megkönnyíti a webhely karbantartását, ráadásul így könnyebben értelmezhető URL-eket használhatunk, és morzsaútvonalakat is létrehozhatunk.

Gyakorlatok

- Elérkeztünk az utolsó óra végéhez. Ha összeállt a webhelyünk, amelyet már megmutatnánk a világnak, tekintsük át a tartalmát még egyszer a keresésoptimalizálás szempontjából, majd jegyeztessük be a nagyobb keresőknél.



A FÜGGELÉK

HTML- és CSS-források az Interneten

Az ebben a függelékben található hivatkozások csupán egy kis szeletét jelentik annak a számtalan forrásnak, amelyre egy egyszerű kulcsszavas kereséssel rábukkanhatunk, de ha zűg a fejünk a kínálattól, jó kiindulópontot jelenthetnek.

Általános információk a HTML-lel, az XHTML-lel és a CSS-sel kapcsolatban

The World Wide Web Consortium (W3C):

<http://www.w3.org/>

The W3C Markup Validation Service (leírókód-ellenőrző szolgáltatás):

<http://validator.w3.org/>

W3Schools.com Web Building Tutorials (webfejlesztési leckék):

<http://www.w3schools.com/>

The Web Standards Project:

<http://www.webstandards.org/>

The HTML Writer's Guild:

<http://www.hwg.org/>

The Web Developer's Virtual Library (virtuális webfejlesztői könyvtár):

<http://www.wdvl.com/>

Webböngészők

Apple Safari:

<http://www.apple.com/safari/>

Google Chrome:

<http://www.google.com/chrome/>

Microsoft Internet Explorer:

<http://www.microsoft.com/windows/ie/>

Mozilla Firefox:

<http://www.getfirefox.com/>

Opera:

<http://www.opera.com/>

Weboldalak tervezése

Web Monkey:

<http://webmonkey.wired.com/webmonkey/>

A List Apart („azoknak, akik webhelyeket készítenek”):

<http://www.alistapart.com/>

Web Pages That Suck (borzasztó webhelyek):

<http://www.webpagesthatsuck.com/>

HTML Help (Web Design Group):

<http://www.htmlhelp.com/>

Szoftver

Adobe Creative Suite

<http://www.adobe.com/products/creativesuite/>

Corel PaintPro:

<http://www.corel.com/>

GIMP (GNU Image Manipulation Program):

<http://gimp.org/>

Picasa:

<http://picasa.google.com/>

Mapedit:

<http://www.boutell.com/mapedit/>

Shareware.com:

<http://shareware.cnet.com/>

Classic FTP:

<http://www.nchsoftware.com/classic/>

Cyberduck (FTP-ügyfélprogram):

<http://cyberduck.ch/>

FileZilla (FTP-ügyfélprogram):

<http://filezilla-project.org/>

Színek és grafika

Microsoft Clip Art Gallery:

<http://dgl.microsoft.com/>

Barry's Art Server:

<http://www.barrysclipart.com/>

HTML Color Picker:

<http://www.pagetutor.com/pagetutor/makapage/picker/>

HTML-színkódok:

<http://htmlcolorcodes.org/>

Color Scheme Designer:

<http://colorschemedesigner.com/>

Kuler by Adobe:

<http://kuler.adobe.com/>

Color Blender:

<http://www.meyerweb.com/eric/tools/color-blend/>

Multimédia

Apple QuickTime:

<http://www.apple.com/quicktime/>

Windows Movie Maker:

<http://www.microsoft.com/windowsxp/using/moviemaker/default.mspx>

RealAudio:

<http://www.real.com/>

Adobe Flash:

<http://www.adobe.com/products/flash/>

Sound Central:

<http://www.soundcentral.com/>

MIDIworld:

<http://www.midiworld.com/>

Fejlesztői források haladóknak

WebReference:

<http://www.webreference.com/>

JavaScript.com:

<http://www.javascript.com/>

Az IRT.org fejlesztői forrásai:

<http://www.irt.org/>

Bérelt webtárhely-szolgáltatás

Web Hosting Geeks (tárhelyszolgáltatók értékelése):

<http://webhostinggeeks.com/>

A Small Orange (webtárhely-szolgáltató):

<http://asmallorange.com/hosting/shared/>

Bluehost (webtárhely-szolgáltató):

http://www.bluehost.com/tell_me_more.html

Daily Razor (webtárhely-szolgáltató):

<http://www.dailyrazor.com/php/promo.php>

DreamHost (webtárhely-szolgáltató):

<http://www.dreamhost.com/hosting.html>

Just Host (webtárhely-szolgáltató):

<http://www.justhost.com/web-hosting>

Lunar Pages (webtárhely-szolgáltató):

<http://www.lunarpages.com/starter-hosting/>

Webhely-üzemeltetési szolgáltatások

A Google eszközei webmesterek számára:

<http://www.google.com/webmasters/>

Open Directory Project:

<http://dmoz.org/about.html>

Freedback.com (ingyenes űrlapfeldolgozó szolgáltatás):

<http://www.freedback.com/>



B FÜGGELÉK

XHTML 1.1 és CSS 2 gyorsalpaló

Az XHTML 1.1 a HTML modern, az XML-re épülő változata, amely lehetővé teszi, hogy egyszerűbben határozzunk és valósítsunk meg bővítéseket a nyelvhez. Ebben a függelékben röviden áttekintjük az XHTML 1.1-nek azokat az elemeit és jellemzőit, amelyekkel a leggyakrabban találkozhatunk, valamint a CSS 2 részét képező tulajdonságokat. A teljes leírásokat megtalálhatjuk a <http://www.w3.org/> címen.

Annak érdekében, hogy könnyebben megtaláljuk a keresett információt, a függelék a HTML-elemeket a szerepük szerint csoportosítja, a következő sorrendben:

- Szerkezeti elemek
- Szövegtömbök és bekezdések
- Szövegformázó elemek
- Listák
- Hivatkozások
- Táblázatok
- Beágyazott tartalom
- Stílus
- Űrlapok
- Parancskódok

Az elemeket az egyes szakaszokon belül ábécésorrendben soroltuk fel, és a következő információkat mellékeljük hozzájuk:

- **Használat** – itt az elem általános leírása szerepel.
- **Kezdő- és zárócímke** – itt azt adjuk meg, hogy az említett címkék kötelezőek, elhagyhatók vagy tiltottak-e.
- **Jellemzők** – Itt az elem jellemzőit soroljuk fel, a hatásuk rövid leírásával egyetemben. Az egérműveletekhez kapcsolódó, illetve az ügyféloldali parancsfájlok meghívására használatos jellemzőket nem tüntetjük fel; ha ezekre a műveleti jellemzőkre is kíváncsiak vagyunk, a teljes leírást megtaláljuk a W3C webhelyén.
- **Üres?** – itt azt jelezzük, hogy az elem lehet-e üres is.
- **Megjegyzések** – Itt az elem használatára vonatkozó, megjegyzésre érdemes kiegészítő információkat találunk.

A CSS-stílustulajdonságokat hasonló formában mutatjuk be, azzal a különbséggel, hogy jellemzők helyett az elfogadható értékeket tüntetjük fel.



Az XHTML 1.1 több olyan alapvető jellemzőt is tartalmaz, amelyek számos elemhez kapcsolódhatnak. Ezekre a jellemzőkre az egyes elemek felsorolásakor a *mag-, nyelvi, esemény-* jelöléssel hivatkozunk. Az említett jellemzőcsoportokkal részletesen foglalkozunk, miután minden XHTML-elemet bemutatunk; vagyis az ezekbe a csoportokba tartozó konkrét jellemzőknek a leírását ott találjuk meg.

Szerkezeti XHTML-elemek

Az XHTML számos elemre támaszkodik a dokumentumok (vagyis nem a bennük található szövegek) szerkezetének meghatározásában, bizonyos elemek pedig olyan információkat nyújtanak, amelyekre a böngészőknek, illetve a keresőprogramoknak van szükségük.

Megjegyzések <!-- ... -->

Használat	Olyan megjegyzések vagy parancskódok beszúrására szolgál, amelyeket a böngésző nem jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	Nincsenek.
Üres?	Igen.
Megjegyzések	A megjegyzéseknek nem muszáj egysorosnak lenniük; tetszőleges hosszúságúak lehetnek. A zárócímkének nem kötelező ugyanabban a sorban lennie, mint a kezdőcímkének.

<!doctype...>

Használat	A változatinformáció a HTML-dokumentum első sorában szerepel, és nem elemnek, hanem SGML-deklarációnak minősül.
-----------	-----------------------------------------------------------------------------------------------------------------

<body>...</body>

Használat	A dokumentum tartalmát foglalja magába.
Kezdő- és zárócímke	Elhagyható/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Csak egyetlen <body> elem lehet, és a <head> elem után kell következnie. Ha kereteket használunk (lehetőleg ne tegyük), a <body> elem helyén egy <frameset> elem állhat.

<div>...</div>

Használat	A div (division, szakasz) elem a szöveg blokkokba rendezésére szolgál.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Nem használható p elemen belül.



Olyan HTML-weboldalakkal is találkozhatunk, amelyek a <div> elemet az align (igazítás) nevű jellemzővel együtt használják. Ezt a jellemzőt az XHTML-ből és a HTML 5-ből eltávolították; az igazítást ehelyett a text-align CSS-stílusulajdonság segítségével ajánlják. Ezzel a stílusulajdonsággal a függelék későbbi részében foglalkozunk.

<h1>...</h1> – <h6>...</h6>

Használat	A hat címsorelemet (a h1 a legfelső szintű vagy legfontosabb címsort jelzi) a törzsben (body) használják az információk hierarchikus elrendezésére.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	A grafikus böngészők a címsorokat a fontosságuknak megfelelő méretben jelenítik meg: a <h1> a legnagyobb, a <h6> pedig a legkisebb méretű címsort jelenti.

<head>...</head>

Használat	Ez az elem a dokumentum fejlécét adja meg, és olyan elemeket tartalmaz, amelyek információkat nyújtanak a felhasználóknak és a keresőprogramoknak.
Kezdő- és zárócímke	Elhagyható/elhagyható.
Jellemzők	<i>nyelvi</i> profile="url" – A metaadatok (meta) helyét meghatározó URL.

Üres?	Nem.
Megjegyzések	Dokumentumonként csak egyetlen <head> elem lehet, amelynek a nyitó <html> elem után és a <body> elem előtt kell állnia.



A `profile` jellemző a HTML 5-ben nem megengedett.

<hr />

Használat	A vízszintes elválasztóvonalak (<i>horizontal rule</i>) a weboldal szakaszainak elválasztására szolgálnak.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Igen.

<html>...</html>

Használat	A html elem a teljes dokumentumot magába foglalja.
Kezdő- és zárócímke	Elhagyható/elhagyható.
Jellemzők	<i>nyelvi</i>
Üres?	Nem.
Megjegyzések	A változatinformáció a <!doctype...> bevezetésben is szerepel, ezért a feltüntetése itt nem létfontosságú.

<meta />

Használat	A dokumentumról nyújt információkat.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<i>nyelvi</i> http-equiv=" <i>kiszolgálóparancs</i> " – egy HTTP-válaszfejléc neve. name=" <i>név</i> " – A metainformáció neve. content=" <i>érték</i> " – A metainformáció tartalma. scheme=" <i>séma</i> " – Egy sémát rendel a metaadatok értelmezéséhez.
Üres?	Igen.



A `scheme` jellemző a HTML 5-ben nem megengedett.

...

Használat	A dokumentumot rendezi egy szövegszakasz meghatározásával.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

<title>...</title>

Használat	A weboldalnak adott nevet (címet) tartalmazza. A <title> elemet a <head> elembe kell elhelyezni, és a tartalma a böngészőablak címsorában jelenik meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>nyelvi</i>
Üres?	Nem.
Megjegyzés	Egy dokumentumhoz csak egyetlen cím rendelhető.

XHTML-szövegtömbök és -bekezdések

A szövegtömbök (vagy *blokkok*) szerkezetét úgy alakíthatjuk ki, hogy egy adott célt szolgáljanak – létrehozhatunk például bekezdéseket. Ez nem keverendő össze a szöveg formázásának módosításával.

<blockquote>...</blockquote>

Használat	Hosszú idézetek megjelenítésére szolgál.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>cite="url"</i> – Az idézett szöveg URL-je.
Üres?	Nem.

**
**

Használat	Sortörést kényszerít ki.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Igen.

<cite>...</cite>

Használat	Hivatkozásra szolgál.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<code>...</code>`

Használat	Egy megjelenítendő kódrészletet határoz meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<h1>...</h1> – <h6>...</h6>`

Használat	Szövegcímsorok meghatározására szolgál.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<p>...</p>`

Használat	Egy bekezdést határoz meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<pre>...</pre>`

Használat	Előre formázott szöveget jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`...`

Használat	Erősebb kiemelést biztosít.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`_{...}`

Használat	Alsó indexbe tett szöveget jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`^{...}`

Használat	Felső indexbe tett szöveget jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

Szövegformázó XHTML-elemek

Ezekkel az elemekkel az általános szövegjellemzők (például a szöveg mérete, a betűvastagság vagy a stílus) módosíthatók, de az ajánlott módszer a CSS-stílustulajdonságok használata. A függelék későbbi részében teljes leírást találunk a stílustulajdonságokról, amelyekkel a szövegek formázása hihetetlenül aprólékosan szabályozható.

`...`

Használat	Félkövér (bold) szöveget jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<big>...</big>`

Használat	Nagy méretű szöveget jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.



Ezt az elemet eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre van hatással, amit célszerűbb CSS-kóddal szabályozni.

`<i>...</i>`

Használat	Dőlt betűs szöveget (italic) jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<small>...</small>`

Használat	Kis méretű szöveget jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<tt>...</tt>`

Használat	Azonos szélességű betűket (teletype vagy monospaced) jelenít meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.



Ezt az elemet eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre van hatással, amit célszerűbb CSS-kóddal szabályozni.

XHTML-listák

Listák segítségével strukturáltabb formában jeleníthetjük meg a szövegeket. A listák egymásba ágyazhatók.

`<dd>...</dd>`

Használat	Egy <code><dl></code> (<i>definition list</i>) elemben használt meghatározásleírást ad meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Blokkszintű tartalom is lehet benne (például egy <code><p></code> elem).

`<dl>...</dl>`

Használat	Egy meghatározáslistát (<i>definition list</i>) hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Legalább egy <code><dt></code> vagy <code><dd></code> elemet tartalmaznia kell, tesztölges sorrendben.

`<dt>...</dt>`

Használat	Egy <code><dl></code> (<i>definition list</i>) elemen belül használt meghatározás-kifejezést (vagy címkét) ad meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Szöveget kell tartalmaznia (amely szövegformázó elemekkel módosítható).

`...`

Használat	Egy listaelemet határoz meg egy listán belül.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

...

Használat	Rendezett listát hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Legalább egy listaelemet tartalmaznia kell.

...

Használat	Rendezetlen listát hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Legalább egy listaelemet tartalmaznia kell.

XHTML-hivatkozások

A hiperhivatkozások alapvető szerepet töltenek be az XHTML-ben. Ezek az elemek teszik lehetővé, hogy más dokumentumokra, az adott dokumentum más részeire, illetve külső fájlokra hivatkozzunk.

<a>...

Használat	Hivatkozások és horgonyok meghatározására szolgál.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>charset="kódolás"</i> – A forrás karakterkódolása. <i>name="név"</i> – Egy horgonyt határoz meg. <i>href="hivatkozott_url"</i> – A hivatkozott forrás URL-je. <i>rel="hivatkozástípus"</i> – Előre mutató hivatkozástípusok. <i>rev="hivatkozástípus"</i> – Visszafelé mutató hivatkozás-típusok. <i>shape="érték"</i> – Ügyféloldali képtérképek meghatározását teszi lehetővé meghatározott alakzatok (default, rect, circle, poly, vagyis alapértelmezett, téglalap, kör és sokszög) használatával. <i>coords="értékek"</i> – Az alakzat méretét állítja be képpontban vagy százalékos hosszúságban.
Üres?	Nem.



A *charset*, *name*, *rev*, *shape* és *coords* jellemzők a HTML 5-ben nem megengedettek.

<base />

Használat	A dokumentum minden URL-jét az itt megadott helyhez kell viszonyítani.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<i>href="hivatkozott_url"</i> – A hivatkozott forrás URL-je.
Üres?	Igen.
Megjegyzések	A dokumentum <head> részében kell elhelyezni.

<link />

Használat	Egy hivatkozás és egy forrás kapcsolatát határozza meg.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>charset="kódolás"</i> – A forrás karakterkódolása. <i>href="hivatkozott_url"</i> – A forrás URL-je. <i>rel="hivatkozástípus"</i> – Előre mutató hivatkozástípusok. <i>rev="hivatkozástípus"</i> – Visszafelé mutató hivatkozástípusok. <i>media="megjelenítő"</i> – A célközeget határozza meg (ami screen, print, projection, braille, speech vagy all – képernyő, nyomtatás, vetítés, Braille-írás, beszéd vagy mind – lehet). <i>target="hely"</i> – Azt adja meg, hogy hol kell megjeleníteni a forrást (az értéke blank, parent, self, top – üres, szülő, önmagában, felül – vagy egy felhasználó által megadott név lehet).
Üres?	Igen.
Megjegyzések	A dokumentum <head> részében kell elhelyezni.



A **charset**, **rev** és **target** jellemzők a HTML 5-ben nem megengedettek.

XHTML-táblázatok

A táblázatok az adatok táblázatos formában történő megjelenítését szolgálják. Az XHTML előtt a táblázatokot széles körben használták oldalformázási célokra, de a stíluslapok megjelenésével a W3C hivatalosan is nem ajánlotta tette ezt a megoldást, és kötetünk szerzői sem javasolják az alkalmazását.

<caption>...</caption>

Használat	A táblázat címét adja meg.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.
Megjegyzések	Elhagyható.

`<col />`

Használat	Csoportokba rendezi az egyes oszlopokat, hogy közös jellemzőértékeket adhassunk nekik.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<p><i>mag-, nyelvi, esemény-</i></p> <p><i>span="oszlopszám"</i> – A csoportban található oszlopok száma.</p> <p><i>width="szélesség"</i> – Oszlopszélesség százalékban, képpontban vagy minimumértékkel kifejezve.</p> <p><i>align="igazítás"</i> – A cellák tartalmát vízszintesen igazítja (az igazítás <i>left</i>, <i>center</i>, <i>right</i>, <i>justify</i> vagy <i>char</i> – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet).</p> <p><i>char="karakterigazítás"</i> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell.</p> <p><i>charoff="karaktereltolás"</i> – A sor első igazítási karakteréhez való eltolást határoz meg.</p> <p><i>valign="függőleges_igazítás"</i> – A cellák tartalmát függőlegesen igazítja (az igazítás <i>top</i>, <i>middle</i>, <i>bottom</i> vagy <i>baseline</i> – felülre, középre, alulra vagy alapvonalra – lehet).</p>
Üres?	Igen.



A *width*, *align*, *char*, *charoff* és *valign* jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

`<colgroup>...</colgroup>`

Használat	Egy oszlop csoportot határoz meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<p><i>mag-, nyelvi, esemény-</i></p> <p><i>span="oszlopszám"</i> – A csoportban található oszlopok száma.</p> <p><i>width="szélesség"</i> – Az oszlopok szélessége.</p> <p><i>align="igazítás"</i> – A cellák tartalmát vízszintesen igazítja (az igazítás <i>left</i>, <i>center</i>, <i>right</i>, <i>justify</i> vagy <i>char</i> – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet).</p> <p><i>char="karakterigazítás"</i> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell.</p> <p><i>charoff="karaktereltolás"</i> – A sor első igazítási karakteréhez való eltolást határoz meg.</p> <p><i>valign="függőleges_igazítás"</i> – A cellák tartalmát függőlegesen igazítja (az igazítás <i>top</i>, <i>middle</i>, <i>bottom</i> vagy <i>baseline</i> – felülre, középre, alulra vagy alapvonalra – lehet).</p>
Üres?	Nem.



A `width`, `align`, `char`, `charoff` és `valign` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

```
<table>...</table>
```

Használat	Egy táblázatot hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<p><i>mag-</i>, <i>nyelvi</i>, <i>esemény-</i></p> <p><code>width="szélesség"</code> – A táblázat szélessége.</p> <p><code>cols="oszlopszám"</code> – Az oszlopok száma.</p> <p><code>borderwidth="szegélyszélesség"</code> – A táblázat körüli szegély szélessége képpontban.</p> <p><code>frame="keret"</code> – A táblázat látható oldalait adja meg (a lehetséges értékek: <code>void</code>, <code>above</code>, <code>below</code>, <code>hsides</code>, <code>lhs</code>, <code>rhs</code>, <code>vsides</code>, <code>box</code> és <code>border</code>, vagyis üres, felül, alul, vízszintes oldalak, bal, jobb, függőleges oldalak, doboz és szegély).</p> <p><code>rules="elválasztóvonalak"</code> – A táblázat látható elválasztóvonalait adja meg (a lehetséges értékek: <code>none</code>, <code>groups</code>, <code>rows</code>, <code>cols</code> és <code>all</code>, vagyis nincs, csoportok, sorok, oszlopok és minden).</p> <p><code>cellspacing="cellaköz"</code> – A cellák közötti távolságot adja meg.</p> <p><code>cellpadding="cellakitöltés"</code> – A cellákon belüli térközt adja meg.</p> <p><code>summary="leírás"</code> – Szöveges leírást ad a táblázatról (például a fogyatékkal élők számára).</p>
Üres?	Nem.



A `width`, `align`, `border`, `frame`, `rules`, `cellspacing`, `cellpadding` és `summary` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

```
<tbody>...</tbody>
```

Használat	A táblázat törzsét határozza meg.
Kezdő- és zárócímke	Elhagyható/elhagyható.
Jellemzők	<p><i>mag-</i>, <i>nyelvi</i>, <i>esemény-</i></p> <p><code>align="igazítás"</code> – A cellák tartalmát vízszintesen igazítja (az igazítás <code>left</code>, <code>center</code>, <code>right</code>, <code>justify</code> vagy <code>char</code> – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet).</p> <p><code>char="karakterigazítás"</code> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell.</p> <p><code>charoff="karaktereltolás"</code> – A sor első igazítási karakteréhez való eltolást határoz meg.</p>

`valign="függőleges_igazítás"` – A cellák tartalmát függőlegesen igazítja (az igazítás `top`, `middle`, `bottom` vagy `baseline` – felülre, középre, alulra vagy alapvonalra – lehet).
 Üres? Nem.



Az `align`, `char`, `charoff` és `valign` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

`<td>...</td>`

Használat	Egy cella tartalmát határozza meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<p><code>mag-</code>, <code>nyelvi</code>, <code>esemény-</code> <code>abbr="név"</code> – Rövid (abbreviated) név. <code>axis="tengelynevek"</code> – A cellához tartozó sor- és oszlop- fejléceket adják meg. <code>rowspan="sorok_száma"</code> – A cella által átfogott sorok szá- mát adja meg. <code>colspan="oszlopszám"</code> – A cella által átfogott oszlopok számát adja meg. <code>align="igazítás"</code> – A cellák tartalmát vízszintesen igazítja (az igazítás <code>left</code>, <code>center</code>, <code>right</code>, <code>justify</code> vagy <code>char</code> – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet). <code>char="karakterigazítás"</code> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell. <code>charoff="karaktereltolás"</code> – A sor első igazítási karak- teréhez való eltolást határozza meg. <code>valign="függőleges_igazítás"</code> – A cellák tartalmát függőlegesen igazítja (az igazítás <code>top</code>, <code>middle</code>, <code>bottom</code> vagy <code>baseline</code> – felülre, középre, alulra vagy alapvonalra – lehet). <code>headers="fejlécek"</code> – Fejlécinformációkat ad meg a cellához. <code>scope="hatókör"</code> – Azt jelzi, hogy egy cella nyújt-e fejlécinformációkat más cellák számára.</p>
Üres?	Nem.



Az `axis`, `align`, `char`, `charoff`, `valign` és `scope` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

```
<tfoot>...</tfoot>
```

Használat	A táblázat láblécét határozza meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>align="igazítás"</i> – A cellák tartalmát vízszintesen igazítja (az igazítás left, center, right, justify vagy char – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet). <i>char="karakterigazítás"</i> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell. <i>charoff="karaktereltolás"</i> – A sor első igazítási karakteréhez való eltolást határoz meg. <i>valign="függőleges_igazítás"</i> – A cellák tartalmát függőlegesen igazítja (az igazítás top, middle, bottom vagy baseline – felülre, középre, alulra vagy alpvonalra – lehet).
Üres?	Nem.



Az align, char, charoff és valign jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

```
<th>...</th>
```

Használat	A táblázatfejléc cellatartalmát határozza meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>axis="név"</i> – Rövid név. <i>axes="tengelynevek"</i> – A cellához tartozó sor- és oszlopfejléceket adják meg. <i>rowspan="sorok_száma"</i> – Az egy cella által átfogott sorok számát adja meg. <i>colspan="oszlopszám"</i> – Az egy cella által átfogott oszlopok számát adja meg. <i>align="igazítás"</i> – A cellák tartalmát vízszintesen igazítja (az igazítás left, center, right, justify vagy char – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet). <i>char="karakterigazítás"</i> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell. <i>charoff="karaktereltolás"</i> – A sor első igazítási karakteréhez való eltolást határoz meg. <i>valign="függőleges_igazítás"</i> – A cellák tartalmát függőlegesen igazítja (az igazítás top, middle, bottom vagy baseline – felülre, középre, alulra vagy alpvonalra – lehet). <i>headers="fejlécek"</i> – Fejlécinfókat ad meg egy cellához. <i>scope="hatókör"</i> – Azt jelzi, hogy egy cella nyújt-e fejlécinformációkat más cellák számára.
Üres?	Nem.



Az `axis`, `axes`, `align`, `char`, `charoff` és `valign` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

`<thead>...</thead>`

Használat	A táblázat fejlécét határozza meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<p><i>mag-, nyelvi, esemény-</i></p> <p><code>align="igazítás"</code> – A cellák tartalmát vízszintesen igazítja (az igazítás <code>left</code>, <code>center</code>, <code>right</code>, <code>justify</code> vagy <code>char</code> – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet).</p> <p><code>char="karakterigazítás"</code> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell.</p> <p><code>charoff="karaktereltolás"</code> – A sor első igazítási karakteréhez való eltolást határoz meg.</p> <p><code>valign="függőleges_igazítás"</code> – A cellák tartalmát függőlegesen igazítja (az igazítás <code>top</code>, <code>middle</code>, <code>bottom</code> vagy <code>baseline</code> – felülre, középre, alulra vagy alapvonalra – lehet).</p>
Üres?	Nem.



Az `align`, `char`, `charoff` és `valign` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

`<tr>...</tr>`

Használat	Táblázatcellák egy sorát határozza meg.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<p><i>mag-, nyelvi, esemény-</i></p> <p><code>align="igazítás"</code> – A cellák tartalmát vízszintesen igazítja (az igazítás <code>left</code>, <code>center</code>, <code>right</code>, <code>justify</code> vagy <code>char</code> – balra, középre, jobbra, sorkizárt vagy karakter szerinti – lehet).</p> <p><code>char="karakterigazítás"</code> – Egy karaktert ad meg, amelyhez az oszlopnak igazodnia kell.</p> <p><code>charoff="karaktereltolás"</code> – A sor első igazítási karakteréhez való eltolást határoz meg.</p> <p><code>valign="függőleges_igazítás"</code> – A cellák tartalmát függőlegesen igazítja (az igazítás <code>top</code>, <code>middle</code>, <code>bottom</code> vagy <code>baseline</code> – felülre, középre, alulra vagy alapvonalra – lehet).</p>
Üres?	Nem.



Az `align`, `char`, `charoff` és `valign` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

Beágyazott tartalmat szabályozó XHTML-elemek

Beágyazott tartalom vagy *beágyazás* lehet kép, képtérkép, Java-kisalkalmazás, Flash-animáció és más multimédiás vagy programozott tartalom, amelyet egy weboldalon elhelyezünk, hogy további szolgáltatásokat nyújtsunk.

`<area />`

Használat	Az <code><area></code> elemet hivatkozások és horgonyok meghatározására használják.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<p><i>mag-</i>, <i>nyelvi</i>, <i>esemény-</i> <code>shape="érték"</code> – Ügyféloldali képtérképek meghatározását teszi lehetővé meghatározott alakzatok (default, rect, circle, poly, vagyis alapértelmezett, téglalap, kör és sokszög) használatával. <code>coords="értékek"</code> – Az alakzat méretét állítja be képpontban vagy százalékos hosszúságban. <code>href="hivatkozott_URL"</code> – A hivatkozott forrás URL-je. <code>nohref="nohref"</code> – Azt jelzi, hogy a területhez nem tartozik művelet. <code>alt="helyettesítő_szóveg"</code> – Helyettesítő szöveget jelenít meg.</p>
Üres?	Igen.

``

Használat	Egy képet szúr be a dokumentumba.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<p><i>mag-</i>, <i>nyelvi</i>, <i>esemény-</i> <code>src="forrás_URL"</code> – A kép URL-je. <code>alt="helyettesítő szöveg"</code> – Megjelenítendő helyettesítő szöveg. <code>height="magasság"</code> – A kép magassága. <code>width="szélesség"</code> – A kép szélessége. <code>border="szegély"</code> – A szegély szélessége. <code>hspace="vízszintes_térköz"</code> – A képet az egyéb tartalomtól elválasztó vízszintes térköz. <code>vspace="függőleges_térköz"</code> – A képet az egyéb tartalomtól elválasztó függőleges térköz. <code>usemap="térkép_URL-je"</code> – Egy ügyféloldali képtérkép URL-je. <code>ismap="ismap"</code> – Egy kiszolgálóoldali képtérképet azonosít.</p>
Üres?	Igen.



A `hspace` és `vspace` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

`<map>...</map>`

Használat	Ha az <code><area></code> elemmel együtt használják, egy ügyféloldali képtérképet hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <code>name="név"</code> – A létrehozandó képtérkép neve.
Üres?	Nem.

`<object>...</object>`

Használat	Egy objektumot szűr be.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <code>declare="declare"</code> – Olyan jelző, amely csak bevezet egy objektumot, de nem hozza létre azt. <code>classid="objektum_URL-je"</code> – Az objektum URL-je. <code>codebase="kódalap_URL-je"</code> – Az egyéb jellemzők által meghatározott URL-ek feloldásához használandó URL. <code>data="adatok_URL-je"</code> – Az objektum adatainak URL-je. <code>type="adattípus"</code> – Az adatok internetes adattípusa. <code>codetype="kódtípus"</code> – A kód internetes adattípusa. <code>standby="várakozási_üzenet"</code> – A betöltés közben megjelenítendő szöveg. <code>height="magasság"</code> – Az objektum magassága. <code>width="szélesség"</code> – Az objektum szélessége. <code>border="szegély"</code> – Szegélyt jelenít meg az objektum körül. <code>hspace="vízszintes_térköz"</code> – Az objektum oldalai és a weboldal egyéb tartalma közötti térköz. <code>vspace="függőleges_térköz"</code> – Az objektum alja és teteje, valamint a weboldal egyéb tartalma közötti térköz. <code>usemap="térkép_URL-je"</code> – Egy képtérkép URL-je. <code>shapes="shapes"</code> – Azoknak a területeknek a meghatározását teszi lehetővé, ahol hiperhivatkozásokat kell keresni, amennyiben az objektum egy kép. <code>name="név-URL"</code> – Az URL, amelyet egy űrlap részeként át kell adni.
Üres?	Nem.



A `hspace` és `vspace` jellemzőket eltávolították a HTML 5-ből, mert kizárólag a megjelenítésre vannak hatással, amit célszerűbb CSS-kóddal szabályozni.

<param />	
Használat	Egy objektum kezdőértékeinek beállítására szolgál.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	name="név" – A paraméter nevét határozza meg. value="érték" – Az objektumparaméter értéke. valuetype="értéktípus" – Az érték típusát (ami data, ref vagy object – adat, hivatkozás, objektum – lehet) határozza meg. type="tartalomtípus" – Az internetes tartalom típusa.
Ures?	Igen.

XHTML-stílusok

A stíluslapokat (mind az oldalon belüli, mind a külső stíluslapokat) a <style> elemen keresztül építik be a HTML-dokumentumokba.

<style>...</style>	
Használat	Egy belső stíluslapot hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	nyelvi type="tartalomtípus" – Az internetes tartalomtípus. media="megjelenítő" – A célközöget határozza meg (ami screen, print, projection, braille, speech vagy all – képernyő, nyomtatás, vetítés, Braille-írás, beszéd vagy mind – lehet). title="cím" – A stílus neve.
Ures?	Nem.
Megjegyzések	A <head> elembe kell elhelyezni.

XHTML-űrlapok

Az űrlapok olyan felületet hoznak létre, amelyen a felhasználók lehetőségek közül választhatnak, információkat adhatnak meg, illetve adatokat adhatnak vissza a webkiszolgálónak feldolgozásra.

<button>...</button>	
Használat	Egy gombot hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	mag-, nyelvi, esemény- name="név" – A gomb neve. value="érték" – A gomb értéke. type="típus" – A gomb típusa (ami button, submit vagy reset – általános gomb, küldés, alaphelyzet – lehet). disabled="disabled" – A gomb állapotát kikapcsoltra állítja.
Ures?	Nem.

<fieldset>...</fieldset>

Használat	Összetartozó vezérlőket foglal csoportba.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

<form>...</form>

Használat	Egy űrlapot hoz létre. Olyan vezérlőket tartalmaz, amelyeken keresztül adatokat fogadhatunk a felhasználóktól.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>action="művelet_URL-je"</i> – A kiszolgálói művelet URL-je. <i>method="post/get"</i> – A használandó HTTP-függvény. A get elavultnak minősül. <i>enctype="hordozótípus"</i> – Az internetes hordozótípust (MIME) határozza meg. <i>accept="tartalomtípusok"</i> – A kiszolgáló által elfogadható tartalomtípusok listája. <i>accept-charset="kódolások"</i> – A karakterkódolások felsorolása.
Üres?	Nem.

<input />

Használat	Űrlapon használt vezérlők meghatározására szolgál.
Kezdő- és zárócímke	Kötelező/tiltott.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>type="vezérlőtípus"</i> – A beviteli vezérlő típusa (ami text, password, checkbox, radio, submit, reset, file, hidden, image, button, vagyis szöveg, jelszó, jelölőnégyzet, választógomb, „küldés” gomb, „alaphelyzet” gomb, fájl, rejtett, kép vagy gomb lehet). <i>name="név"</i> – A vezérlő neve (a submit és reset típusok kivételével kötelező a megadása). <i>value="érték"</i> – A vezérlő kezdőértéke (a választógombok és jelölőnégyzetek esetében kell megadni). <i>checked="checked"</i> – A választógombokat bejelölt állapotba állítja. <i>disabled="disabled"</i> – Letiltja a vezérlőt. <i>readonly="readonly"</i> – A szöveg (text) és jelszó (password) típusú vezérlőket teszi írásvédetté. <i>size="méret"</i> – A vezérlő szélessége képpontban, illetve a text és password típusok esetében a karakterek számában megadva. <i>maxlength="legnagyobb_hosszúság"</i> – A legfeljebb beírható karakterek száma.

`src="kép_URL-je"` – Egy kép vezérlő URL-je.
`alt="helyettesítő_szóveg"` – Helyettesítő szöveges leírás.
`usemap="térkép_URL-je"` – Egy ügyféloldali képtérkép URL-je.
`accept="fájl_típusok"` – A feltölthető fájl típusokat adja meg.

Üres?	Igen.
-------	-------

`<label>...</label>`

Használat	Címkével lát el egy vezérlőt.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>for="vezérlő"</i> – A címkét egy azonosított vezérlőhöz rendeli.
Üres?	Nem.

`<option>...</option>`

Használat	Választási lehetőségeket határoz meg egy <code><select></code> elemben.
Kezdő- és zárócímke	Kötelező/elhagyható.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>selected="selected"</i> – Azt adja meg, hogy az adott lehetőség van-e kiválasztva. <i>disabled="disabled"</i> – Letiltja a vezérlőt. <i>label="címke"</i> – Egy címkét határoz meg a lehetőségcsoport számára. <i>value="érték"</i> – Az adott vezérlővel átadandó érték.
Üres?	Nem.

`<select>...</select>`

Használat	Lehetőségeket határoz meg, amelyek közül a felhasználó választhat.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>name="név"</i> – Az elem neve. <i>size="méret"</i> – A sorok számában mért szélesség. <i>multiple="multiple"</i> – Több elem egyidejű kiválasztását teszi lehetővé. <i>disabled="disabled"</i> – Letiltja a vezérlőt.
Üres?	Nem.

`<textarea>...</textarea>`

Használat	Egy többsoros beviteli szövegmezőt hoz létre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i> <i>name="név"</i> – A vezérlő neve. <i>rows="sorok_száma"</i> – A sorok számában mért szélesség.

`cols="oszlopok_száma"` – Az oszlopok számában mért magasság.

`disabled="disabled"` – Letiltja a vezérlőt.

`readonly="readonly"` – A megjelenített szöveget írásvédetté teszi.

Üres?	Nem.
Megjegyzések	A megjelenítendő szöveget a kezdő- és a zárócímke között kell elhelyezni.

XHTML-parancskódok

Parancskódok segítségével adatokat dolgozhatunk fel és más dinamikus műveleteket hajthatunk végre. A parancskódokat a `<script>` elem segítségével ágyazzák be a weboldalakba, amely egyben a használt parancsnyelvet (JavaScript, VBScript stb.) is meghatározza.

`<noscript>...</noscript>`

Használat	Alternatív tartalmat nyújt azoknak a böngészőknek, amelyek nem képesek végrehajtani a parancskódokat.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<i>mag-, nyelvi, esemény-</i>
Üres?	Nem.

`<script>...</script>`

Használat	A <code><script></code> elem ügyféloldali parancskódokat tartalmaz, amelyeket a böngésző hajt végre.
Kezdő- és zárócímke	Kötelező/kötelező.
Jellemzők	<code>type="parancskódtípus"</code> – A kódhoz használt internetes parancsnyelv. <code>src="parancsfájl_URL-je"</code> – Egy külső parancsfájl URL-je. <code>defer="defer"</code> – Azt jelzi, hogy a parancskód nem változtatja meg a dokumentum tartalmát.
Üres?	Nem.
Megjegyzések	Az alapértelmezett parancsnyelvet a <code><meta /></code> elemben állíthatjuk be.

Általános XHTML-jellemzők

A fentiekben a következő hat magjellemzőt rövidítettük *mag-* néven:

- `id="azonosító"` – Globális azonosító.
- `class="stílusosztályok"` – Osztályok listája szóközzel elválasztva.
- `style="stílusok"` – Stílusinformációk.
- `title="cím"` – Egy adott elemről nyújt további információt (szemben a `<title>` elemmel, amely a teljes weboldalnak ad címet).

- `accesskey="gyorsbillentyű"` – Egy adott elem eléréséhez használható billentyűparancsot állít be.
- `tabindex="tabulátorsorrend"` – Egy elem helyét adja meg a tabulátorsorrendben.

A fentiekben az alábbi két nyelvi jellemzőt a *nyelvi* rövidítéssel jeleztük:

- `lang="nyelv"` – A nyelv azonosítója.
- `dir="szövegirány"` – A szöveg olvasási iránya (ami `ltr` és `rtl`, vagyis balról jobbra vagy jobbról balra lehet).

A következő belső műveleteket (eseményjellemzőket) az *esemény*-összefoglaló névvel adtuk meg. Ha többet szeretnénk tudni az alkalmazásukról a konkrét elemekben, nézzünk utána a W3C leírásában:

- `onclick="eseménykód"` – A mutatóeszközzel (például egérrel) egyet kattintottak.
- `ondblclick="eseménykód"` – A mutatóeszközzel (például egérrel) duplán kattintottak.
- `onmousedown="eseménykód"` – Az egérgombbal kattintottak, és lenyomva tartják.
- `onmouseup="eseménykód"` – Az egérgombot, amellyel kattintottak, és amelyet lenyomva tartottak, felengedték.
- `onmouseover="eseménykód"` – Az egérrel egy objektum fölé vitték a mutatót.
- `onmousemove="eseménykód"` – Az egeret megmozdították.
- `onmouseout="eseménykód"` – Az egérmutatót elmozdították az objektum fölül.
- `onkeypress="eseménykód"` – Egy billentyűt lenyomtak és felengedtek.
- `onkeydown="eseménykód"` – Egy billentyűt lenyomtak és lenyomva tartanak.
- `onkeyup="eseménykód"` – A billentyűt, amelyet lenyomtak, felengedtek.

Méretbeállító CSS-stílustulajdonságok

A méretbeállító tulajdonságokra jónéhány CSS-stílusszabály támaszkodik valamilyen formában. Nélkülük nehéz lenne méretezni az elemeket.

height	
Használat	Egy elem magasságát állítja be.
Értékek	auto, hossz, %

line-height	
Használat	Elemek sora között állítja be a távolságot.
Értékek	normal, hossz, %

max-height	
Használat	Egy elem lehetséges legnagyobb magasságát állítja be.
Értékek	none, hossz, %

max-width

Használat	Egy elem lehetséges legnagyobb szélességét állítja be.
Értékek	<i>none, hossz, %</i>

min-height

Használat	Egy elem lehetséges legkisebb magasságát állítja be.
Értékek	<i>hossz, %</i>

min-width

Használat	Egy elem lehetséges legkisebb szélességét állítja be.
Értékek	<i>hossz, %</i>

width

Használat	Egy elem szélességét állítja be.
Használat	Egy elem lehetséges legnagyobb magasságát állítja be.
Értékek	<i>auto, hossz, %</i>

Szöveg- és betűstílus-beállító CSS-tulajdonságok

A CSS-stílusok lelkét a szöveg- és betűstílus-beállító tulajdonságok jelentik, amelyekkel hihetetlen mértékben szabályozhatjuk a weboldalak szövegének megjelenését.

color

Használat	A szöveg színét állítja be.
Értékek	<i>szín</i>

direction

Használat	A szöveg irányát állítja be (balról jobbra vagy jobbról balra haladó).
Értékek	<i>ltr, rtl</i>

font

Használat	Gyorsírási tulajdonság, amely lehetővé teszi, hogy minden betűtulajdonságot egyetlen meghatározással állítsunk be.
Értékek	<i>font-style, font-variant, font-weight, font-size/line-height, font-family</i>

font-family

Használat	Az előnyben részesített betűcsaládok neve, illetve általános betűcsaládnevek egy elem számára.
Értékek	<i>betűcsalád_neve, generic-family</i>

font-size	
Használat	A betűméretet állítja be.
Értékek	xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, <i>hossz</i> , %
font-style	
Használat	A betűstílust állítja be.
Értékek	normal, italic, oblique
font-variant	
Használat	A szöveget kiskapitális vagy normál betűkkel jeleníti meg.
Értékek	normal, small-caps
font-weight	
Használat	A betűk súlyát (vastagságát) állítja be.
Értékek	normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900
letter-spacing	
Használat	Csökkenti vagy növeli a szöveg karakterei közötti térközt.
Értékek	normal, <i>hossz</i>
text-align	
Használat	A szöveg igazítását állítja be egy elemen belül.
Értékek	left, right, center, justify
text-decoration	
Használat	Valamilyen „díszítést” állít be a szövegre.
Értékek	none, underline, overline, line-through, blink
text-indent	
Használat	Behúzza az első szövegsort egy elemen belül.
Értékek	<i>hossz</i> , %
text-transform	
Használat	A szöveg betűinek betűállását szabályozza.
Értékek	none, capitalize, uppercase, lowercase
white-space	
Használat	Az üreshelyek kezelését állítja be egy elemen belül.
Értékek	normal, pre, nowrap

word-spacing

Használat	Csökkenti vagy növeli a szavak közötti térközöket.
Értékek	normal, <i>hossz</i>

Háttérbeállító CSS-stílustulajdonságok

Az oldalak, illetve az azokon szereplő egyes elemek háttérének módosítására több CSS-stílustulajdonság is használható.

background

Használat	Gyorsíráshoz tulajdonság, amely lehetővé teszi, hogy minden háttértulajdonságot egyetlen meghatározással állítsunk be.
Értékek	background-color, background-image, background-repeat, background-attachment, background-position

background-attachment

Használat	Azt adja meg, hogy a háttérkép rögzített, vagy együtt gördül az oldal többi részével.
Értékek	scroll, fixed

background-color

Használat	Egy elem háttérszínét állítja be.
Értékek	color-rgb, color-hex, color-name, transparent

background-image

Használat	Háttérként egy képet állít be.
Értékek	url, none

background-position

Használat	A háttérkép kezdőpozícióját állítja be.
Értékek	top left, top center, top right, center left, center center, center right, bottom left, bottom center, bottom right, <i>x-%</i> , <i>y-%</i> , <i>x-pozíció</i> , <i>y-pozíció</i>

background-repeat

Használat	Azt adja meg, hogy a háttérkép ismétlődik-e, és ha igen, hogyan.
Értékek	repeat, repeat-x, repeat-y, no-repeat

Szegélybeállító CSS-stílustulajdonságok

Minden blokkelemnek van szegélye, amely formázható. A szegélyeket természetesen láthatatlanra is állíthatjuk, de az elemek szegélyeire különféle stílusokat is alkalmazhatunk.

border

Használat	Gyorsírásos tulajdonság, amely lehetővé teszi, hogy a négy szegély tulajdonságát egyetlen meghatározással állítsuk be.
Értékek	border-width, border-style, border-color

border-bottom

Használat	Gyorsírásos tulajdonság, amely lehetővé teszi, hogy az alsó szegély minden tulajdonságát egyetlen meghatározással állítsuk be.
Értékek	border-bottom-width, border-style, border-color

border-bottom-color

Használat	Az alsó szegély színét állítja be.
Értékek	border-color

border-bottom-style

Használat	Az alsó szegély stílusát állítja be.
Értékek	border-style

border-bottom-width

Használat	Az alsó szegély szélességét állítja be.
Értékek	thin, medium, thick, <i>hossz</i>

border-color

Használat	A négy szegély színét állítja be.
Értékek	<i>szín</i>
Megjegyzések	Egy-négy szín megadásával állítható be.

border-left

Használat	Gyorsírásos tulajdonság, amely lehetővé teszi, hogy a bal oldali szegély minden tulajdonságát egyetlen meghatározással állítsuk be.
Értékek	border-left-width, border-style, border-color

border-left-color

Használat	A bal oldali szegély színét állítja be.
Értékek	border-color

border-left-style

Használat A bal oldali szegély stílusát állítja be.

Értékek border-style

border-left-width

Használat A bal oldali szegély szélességét állítja be.

Értékek thin, medium, thick, *hossz*

border-right

Használat Gyorsíráshoz tulajdonság, amely lehetővé teszi, hogy a jobb oldali szegély minden tulajdonságát egyetlen meghatározással állítsuk be.

Értékek border-right-width, border-style, border-color

border-right-color

Használat A jobb oldali szegély színét állítja be.

Értékek border-color

border-right-style

Használat A jobb oldali szegély stílusát állítja be.

Értékek border-style

border-right-width

Használat A jobb oldali szegély szélességét állítja be.

Értékek thin, medium, thick, *hossz*

border-style

Használat A négy szegély stílusát állítja be.

Értékek none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

Megjegyzések Egy-négy stílus megadásával állítható be.

border-top

Használat Gyorsíráshoz tulajdonság, amely lehetővé teszi, hogy a felső szegély minden tulajdonságát egyetlen meghatározással állítsuk be.

Értékek border-top-width, border-style, border-color

border-top-color

Használat A felső szegély színét állítja be.

Értékek border-color

border-top-style

Használat	A felső szegély stílusát állítja be.
Értékek	border-style

border-top-width

Használat	A felső szegély szélességét állítja be.
Értékek	thin, medium, thick, <i>hossz</i>

border-width

Használat	Gyorsíráshoz tulajdonság, amely lehetővé teszi, hogy a négy szegély szélességét egyetlen meghatározással állítsuk be.
Értékek	thin, medium, thick, <i>hossz</i>
Megjegyzések	Egy-négy szélesség megadásával állítható be.

Margóbeállító CSS-stílustulajdonságok

A margók lehetővé teszik, hogy az elemek külső éle körül (az elem szegélyén kívül) némi helyet hagyjunk.

margin

Használat	Gyorsíráshoz tulajdonság, amely lehetővé teszi, hogy minden margótulajdonságot egyetlen meghatározással állítsunk be.
Értékek	margin-top, margin-right, margin-bottom, margin-left

margin-bottom

Használat	Egy elem alsó margóját állítja be.
Értékek	auto, <i>hossz</i> , %

margin-left

Használat	Egy elem bal margóját állítja be.
Értékek	auto, <i>hossz</i> , %

margin-right

Használat	Egy elem jobb margóját állítja be.
Értékek	auto, <i>hossz</i> , %

margin-top

Használat	Egy elem felső margóját állítja be.
Értékek	auto, <i>hossz</i> , %

Kitöltésbeállító CSS-stílustulajdonságok

A kitöltés lehetővé teszi, hogy az elemek körül (az elem szegélyén belül) némi helyet hagyjunk.

padding

Használat	Gyorsíráshoz tulajdonság, amely lehetővé teszi, hogy minden kitöltési tulajdonságot egyetlen meghatározással állítsunk be.
Értékek	padding-top, padding-right, padding-bottom, padding-left

padding-bottom

Használat	Egy elem alsó kitöltését állítja be.
Értékek	hossz, %

padding-left

Használat	Egy elem bal oldali kitöltését állítja be.
Értékek	hossz, %

padding-right

Használat	Egy elem jobb oldali kitöltését állítja be.
Értékek	hossz, %

padding-top

Használat	Egy elem felső kitöltését állítja be.
Értékek	hossz, %

Elrendezési és megjelenítési CSS-stílustulajdonságok

A CSS-ben az elrendezési és megjelenítési stílustulajdonságok rendkívül fontos szerepet játszanak abban, hogy meghatározzuk az elemek helyét és elrendezését az oldalon.

bottom

Használat	Az eltolást állítja be az elem alsó szegélye és a szülőelem alsó szegélye között.
Értékek	auto, hossz, %

clear

Használat	Meghatározza egy elemnek azokat az oldalait, ahol más úsztatott elemek nem megengedettek.
Értékek	left, right, both, none

clip

Használat	Egy elem alakját állítja be.
Értékek	auto, <i>alak</i>
Megjegyzések	Az elemet a böngésző megjelenítéskor a megadott alakra vágja.

cursor

Használat	A megjelenítendő egérmutató-típust határozza meg.
Értékek	url, auto, crosshair, default, pointer, move, e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help

display

Használat	Azt határozza meg, hogy az adott elemet meg kell-e jeleníteni, és ha igen, hogyan.
Értékek	none, inline, block, list-item, run-in, compact, marker, table, inline-table, table-row-group, table-header-group, table-footer-group, table-row, table-column-group, table-column, table-cell, table-caption

float

Használat	Azt állítja be, hogy egy kép vagy szöveg hol jelenjen meg egy másik elemhez viszonyítva.
Értékek	left, right, none

left

Használat	Az eltolást állítja be az elem bal oldali szegélye és a szülőelem bal oldali szegélye között.
Értékek	auto, <i>hossz</i> , %

overflow

Használat	Azt határozza meg, hogy mi történjen, ha egy elem tartalma túlsordul a rendelkezésére álló területen.
Értékek	auto, visible, hidden, scroll

position

Használat	Egy elem elhelyezését határozza meg statikus, relatív, abszolút vagy rögzített elhelyezéssel.
Értékek	static, relative, absolute, fixed

right

Használat Az eltolást állítja be az elem jobb oldali szegélye és a szülőelem jobb oldali szegélye között.

Értékek auto, *hossz*, %

top

Használat Az eltolást állítja be az elem felső szegélye és a szülőelem felső szegélye között.

Értékek auto, *hossz*, %

vertical-align

Használat Egy elem függőleges igazítását állítja be.

Értékek baseline, sub, super, top, text-top, middle, bottom, text-bottom, *hossz*, %

visibility

Használat Azt határozza meg, hogy egy elemnek megjelenítendőnek (láthatónak) vagy rejtettnek (láthatatlannak) kell lennie.

Értékek visible, hidden, collapse

z-index

Használat Egy elem z-sorrendbeli (a függőleges veremben elfoglalt) helyét határozza meg.

Értékek auto, *szám*

Lista- és felsorolásjel-beállító CSS-stílustulajdonságok

Lehet, hogy még nem ébredtünk rá, hogy mennyire rugalmasan formázhatjuk a felsorolásokat a CSS segítségével. A listákra és felsorolásjelekre több CSS-stílus is alkalmazható.

list-style

Használat Gyorsírási tulajdonság, amely lehetővé teszi, hogy minden listatulajdonságot egyetlen meghatározással állítsunk be.

Értékek list-style-type, list-style-position, list-style-image

list-style-image

Használat A lista felsorolásjeleként egy képet állít be.

Értékek none, *url*

list-style-position

Használat	Azt adja meg, hogy hol kell elhelyezni a lista felsorolásijelét.
Értékek	inside, outside

list-style-type

Használat	A felsorolás típusát állítja be.
Értékek	none, disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-alpha, upper-alpha, lower-greek, lower-latin, upper-latin, hebrew, armenian, georgian, cjk-ideographic, hiragana, katakana, hiragana-iroha, katakana-iroha

Táblázatformázó CSS-stílustulajdonságok

A haladók néhány olyan táblázattulajdonságot is használhatnak, amelyeknek a segítségével finomhangolhatják a táblázatok leképezését és megjelenítését.

border-collapse

Használat	Egy táblázat szegélymodelljét állítja be.
Értékek	collapse, separate

border-spacing

Használat	A szomszédos cellák szegélye közötti távolságot állítja be.
Értékek	<i>hossz hossz</i>

caption-side

Használat	A táblázatcím helyét állítja be a táblázathoz viszonyítva.
Értékek	top, bottom, left, right

empty-cells

Használat	Azt határozza meg, hogy a látható tartalommal nem rendelkező celláknak legyen-e szegélye.
Értékek	show, hide

table-layout

Használat	A táblázat elrendezését határozza meg.
Értékek	auto, fixed
Megjegyzések	A rögzített méretű táblázatok leképezését felgyorsítja a böngészőben, ha fixed-re állítjuk őket.

Tárgymutató



jel 129
& jel 97
» 286
.asp 30
.htm 30
.jsp 30
.php 30
@import 68, 337
„lapos” hivatkozási szerkezet 383
<a href> 123, 183, 201
<a id> 123
<a> 120, 122, 130
<area /> 191
 87-88
<big> 89
<blockquote> 78
<body> 34

 36

 36
<dd> 76, 266
<div> 62, 74, 283, 304
<dl> 76, 266
<dt> 76, 266
 88
<embed /> 204, 206
 87, 92
<form> 359
<frame /> 219
<frameset> 217
<h1> 38

<head> 35
<hr /> 37
<html> 34
<i> 87-88
<iframe> 221
 174, 178
<input /> 363
 76, 266
<link /> 55, 336
<map/> 277
<meta /> 394-395, 400
<noframes> 218
<object> 204, 206
 76, 266
<option> 367
<p> 35
<param> 205
<pre> 90-91
<script> 347
<select> 367
<small> 89
 65
<src> 219
<strike> 90
 88
<style> 64
<sub> 89
<sup> 89
<table> 104
<td> 104

A, Á`<textarea>` 368`<th>` 105`<title>` 34, 40`<tr>` 104`<tt>` 90-91`<u>` 90`` 76, 266`a:active` 133`a:hover` 133`a:link` 133`a:visited` 133

AAC 210

ablakok megnyitása 132

abszolút cím 121, 129

abszolút elhelyezés 57, 253, 256, 338

action 359

active 133

adatfolyam 210

Adjust Hue/Lightness/Saturation 160

Adobe 154

Adobe Acrobat 341

aktuális sor 260

aláhúzott szöveg 90

alapértelmezett érték 74

alaphelyzet 369

align 230

AllTheWeb 393

álosztály 132, 134

alsó index 89

alt 174, 177

AltaVista 393

Amazon 380

analóg 143

anchor 122

animáció 168

Animált GIF fájl 168

Animation Shop 169

áramló tartalom 210

áramló videó 210

árnyalás 164

ASCII fájl 29

átfedés 254, 258

átfedési sorrend 258

áthúzott szöveg 90

átívelés 110, 112

átívelő cella 112

átlátszó képek 165

átlátszóság 164-165

átméretezés 159, 178

attribute 120

aural 336

autoStart 205

azonosítók 63, 353

ázsiai nyelv 100

B

background 187

background-color 59, 113, 143, 186, 267

background-image 113, 186

background-position 187

background-repeat 186

bal belső margó 269

bal oldali sáv 382

banner 162

Barry's Clipart Server 155

baseline 111, 182

beágyazott alkalmazás 205

beágyazott listák 286

behúzás 38, 60, 78, 385

bejegyzés 392

bejegyzetlen védjegy 97

bejegyzett üzleti védjegy 97

bekezdés 35

belső keret 221

belső margó 230, 238, 251, 269, 272

belső stíluslapok 50, 64

bélyegkép 159, 183

betűméret 55

betűszín 93

betűtípus 92

betűvonal 182

betűvonalhoz igazítás 111

beviteli elemek 360

beviteli vezérlő 365

Bing 393

block 57, 253

blog 23

Blogger 23

blokkelem 253

blokk szintű elemek 72

body 130

bold 89

bolder 89

böngésző 28

böngészőbarát szín 144

border 58, 104

border-color 58, 104

border-spacing 113

border-style 58, 104

border-width 58, 104

bot 391

bővíthető jelölőnyelv 43

C, Cs

bővítmény 202
 braille 336
 burkoló 314
 button 162
 CAPTCHA 393
 cella 104
 cellpadding 113
 cellspacing 113
 CGI 359
 character entity 97, 131
 Character Map 100
 checked 366
 cím 34, 40
 címkék 51
 címsor 38, 40
 class 62, 73
 Classic FTP 16
 classid 204
 clear 246, 260
 color 59, 92-93, 113, 267
 Color Scheme Generator 143, 151
 Colorblind Web Page Filter 149
 cols 217, 368
 colspan 112
 content 401
 coords 192
 copyright 97
 Courier New 91
 Create New Image 163
 Crop 157
 CSS 49-50, 132
 CSS 2 51
 CSS Zen Garden 231, 314
 CSS-stílusok 57, 292
 cursor: pointer 307

D

dashed 148
 dinamikus webhely 345
 Directory Listing Denied 22
 display 57
 display: block 307
 display: none 307
 display: block; 284
 díszítés 133
 dithering 164
 dobozmodell 250, 267
 DOCTYPE 253
 document 353
 document.getElementById() 308
 document.write() 351

dokumentumgyökér 16, 19, 121
 dokumentum-objektummodell 352
 dokumentumtípus 253
 dokumentumtípus-meghatározás 217
 dőlt betű 87
 DOM 352
 DoPDF 341
 dotted 148
 double 59
 DTD 217

E, É

egérmutató 307
 egyenlő szélességű karakterek 91
 egyenlőségjel 130
 egymásba ágyazott listák 78, 266
 egymásba ágyazott táblázatok 117
 egyoldalas megjelenés 374
 egységesség 378
 egysoros szövegmező 363
 egyszerű szövegszerkesztő 29
 egyszerű táblázat 104
 egyszerű webhely 376
 egyszerű weboldal 29
 egyszintű függőleges navigációs sáv 284
 ékezetes betűk 96
 elektronikus levélcím 130
 elem 51, 122
 elemsor 242
 elérhetetlen oldalak 32
 elhelyezés 57, 253
 előnézet 160
 előugró ablak 132
 elrendezés 314
 elrendezési tulajdonságok 57
 elsődleges betűtípus 93
 elsődleges navigációs elemek 280
 eltolás 254
 em 329
 email address encoder 131
 e-mail cím 131
 embossed 336
 érvényesség 33, 40
 érvényességvizsgálat 41
 és jel 130
 eseményjellemzők 303
 eseménykezelés 303
 eszközléírás 177

F

Facebook 390
 fájlátviteli protokoll 14
 fájlkezelés 19

fájlok átvitele 14
 fejléc 35
 felbontás 155, 314
 felhasználói műveletek 303, 353
 félkövér 87
 félkövér szöveg 88
 felső index 89
 felsorolásijel 76, 80, 187, 267
 felsorolásijelek elhelyezése 270
 fényképek előkészítése 156
 Firebug 42
 FireFTP 15
 firstLeftAlign 73
 Flickr 174, 176
 float 179, 230, 243, 260
 folyásirány 260
 folyékony elrendezés 316, 318
 folytonos szegély 59
 font-family 60, 92
 font-size 60, 92-93
 font-style 60, 89
 font-weight 60, 89
 főoldal 22
 fordított perjel 120, 351
 formázási tulajdonságok 58
 forrásszerkesztő 23
 forrópont 274
 frame 168, 213
 frameborder 220
 frameset 216
 FTP 14
 FTP-ügyfél 14, 16
 függőleges görgetés 162
 függőleges igazítás 111, 242
 függőleges ismétlés 186
 függőleges navigációs sáv 281

G, Gy

get 359
 Gickr 168
 GIF 164
 GIMP 154, 157
 gomb 369
 Google 393
 Google Images 174
 Google Maps 332
 gördítősávok 315
 görgethető listák 367
 grafikai alkalmazás kiválasztása 154
 grafikus szerkesztők 23, 29
 gyermekelem 78
 gyökérkönyvtár 16

H

handheld 336
 hasábok magassága 324
 háttérkép 113, 155, 186
 háttérkép mentése 155
 háttérszín 113, 143, 186
 heading 40
 height 58, 107, 178, 251
 helper application 202
 helyettesítő szöveg 177
 helyi abszolút cím 138
 helyi webhely 23
 hexadecimális színátalakító 146
 hexadecimális színkód 144-145
 hidden 261, 364
 hiperszöveg 2
 hiperszöveges jelölőnyelv 2
 hivatkozás elektronikus levélcímre 130
 hivatkozásellenőrző 137
 hivatkozásként használt képek 132
 hivatkozások formázása 132
 hivatkozásstílus 135
 holtter 196
 honlap 10, 22, 376
 hordozófüggő stíluslapok 335
 horgony 122-123, 129
 horgonyhivatkozás 122
 hover 133
 hozzáférhetőség 149
 href 120, 123, 130
 htdocs 18
 HTML 2, 42
 HTML 5 44
 html kiterjesztés 30
 HTML Validator 42
 HTML-címkék 31
 HTML-elem 33
 HTML-jellemzők 72
 HTML-megjegyzés 124
 HTML-táblázat 103
 HTML-ürlapok 357
 http:// 121, 126, 129
 http-equiv 401

I, Í

id 64, 73, 123, 219
 idézetek 349
 igazítás 60, 110, 179, 230, 242
 imagemap 188
 index.html 21
 indexfájl 21
 indexoldal 21

információhordozók 336
 inline 57, 253, 290
 inline frame 221
 inline style 65
 input 363
 inside 270-271
 internetszolgáltató 28
 írógépbetű 90
 italic 89
 iTunes 210

J

JavaScript 346
 JavaScript-kód 347
 Jegyzetomb 29
 jellemző 34, 120
 jellemzők 72
 jelölőnégyzet 365
 JPEG 161

K

kapcsos zárójel 54
 karakteregyed 97, 131
 karakterkód 97, 131
 karaktertábla 100
 kép átméretezése 159
 kép felbontása 156
 kép mentése 155
 kép területekre osztása 190
 képek átalakítása hivatkozássá 183
 képek függőlegesen igazítása 181
 képek igazítása 179
 képek szöveges leírása 177
 képek tömörítése 156
 képernyőfelbontás 314
 képkocka 168
 képpont 156
 képszerkesztő programok 154
 képtérkép 188, 191, 272-273
 képtérkép HTML-kódja 191
 képváltás 353
 kérdőjel 130
 keresés 391
 keresésoptimalizálás 389, 399
 keresőmotorok elárasztása 394
 keresőrobot 391
 keret 213, 250
 keretváz 216
 két pont 121
 kettőskereszt 64, 123, 129, 193
 kézbesítés 4
 kezdőoldal 22, 376
 kijelölő 61

kiszolgálóoldali parancsfájlok 346
 kiterjesztések 30
 kitöltés 61, 230, 238, 251
 kódolási stílus 386
 kódon belüli stílusok 65
 komplementer 143
 körülvágás 157
 kötelező címkék 34
 középre igazítás 74
 középre igazított képek 180
 közösségi hálózatok 391
 kulcsszavak 391, 395
 küldés 369
 különleges karakterek 96
 külső stíluslapok 50

L

lapolvasó 157
 látogatott hivatkozás 135
 lebegő leírás 298, 301
 leftAlignStyle 73
 leírókód 3
 lenyíló listák 367
 lenyíló menü 382
 levélcímgyűjtő programok 131
 levélcímkezőlő 131
 levélszemét 131
 Lightbox 355
 lighter 89
 line-height 60
 link 133
 Link Checker 137
 lista 76
 listaelemek 267
 listaelemek megkülönböztetése 283
 listák 265
 list-item 57
 list-style-image 267
 list-style-position 267, 270-271
 list-style-type 80, 267

M

magasság 58, 107, 178, 251
 mailto 130-131, 358
 main 314
 Mapedit 190
 margin 230-231, 267
 margin-left 96
 margó 230-231, 250, 272
 más webhelyekről származó anyagok 155
 Mashable 391
 másoldalos navigáció 382

másodlagos navigációs elemek 280
 Math.random() 351
 maxlength 363
 media 336
 meghatározás-lista 266
 meghatározáslisták 76
 megjegyzések 124, 350, 384
 méretarány 159
 mértékegység 55
 mértékegységek 58
 method 359
 MIME-típus 205
 minimális oldalszélesség 323
 minőség 161
 min-width 323
 morzaútvonal 400
 mozaikos háttér 166
 MP3 210
 MPEG 210
 multimédia 199
 multimédiafájlok beágyazása 204
 multiple 368

N, Ny

nagyobb webhely 380
 name jellemző 123
 navigációs listák 280
 negatív behúzás 275
 népszerűsítés 390
 név-érték párok 364
 nevesített szín 143
 névvel ellátott horgony 123
 none 57
 no-repeat 186
 normal 89
 Notepad 29
 nyelv 34
 nyitócímke 33
 nyomatékösztítés 88
 nyomógomb 162
 nyomtatási előnézet 341
 nyomtatásra szánt oldalak 334
 nyomtatóbarát stíluslap 337
 nyomtatóbarát weboldal 331

O, Ö

oblique 89
 Office Online Clip Art and Media 155
 oldalhelyek megjelölése 123
 oldalon belüli hivatkozás 122
 oldalszerkezet 35
 oldalváz 34
 olvashatóság 148

onclick 303-304, 353
 ondblclick 303
 onkeydown 303
 onkeyup 303
 onload 303
 onmousedown 303
 onmousemove 303
 onmouseout 303, 353
 onmouseover 303, 353
 onmouseup 303
 összefűzés 351
 oszlopátívelés 112
 oszlopok 103
 oszlopszélesség 110
 osztályazonosító 204
 outside 270
 overflow 260-261

P

padding 61, 113, 230, 238, 267
 padding-left 268
 Page not found 137
 parancsfájlok 346
 PDF 341
 perjel 120
 PHP 358
 pixel 156
 plug-in 202
 pluszjel 351
 PNG 165
 pont 55
 pontosvessző 55, 89
 pop-up window 132
 position 254, 300
 position: relative 321
 post 359
 print 336
 projection 336
 protokoll 121
 pt 55
 public_html 18

Q

Quality 161
 QuickTime 203

R

radio 367
 Red Eye Removal 160
 refresh 401
 region 190
 rejtett adatok 364
 rejtett beviteli elem 364
 reklámcsíki 162

reklámozás 390
 relatív cím 120-121
 relatív elhelyezés 57, 253
 rendezetlen lista 76, 266
 rendezett lista 76, 266
 repeat-x 186
 repeat-y 186
 Reset 369
 Reset Color 160
 rétegek 165
 robots.txt 395
 rögzített elrendezés 314
 rögzített szélességű oldalak 315
 rögzített-folyékony kevert elrendezés
 319, 324
 rows 217, 368
 rowspan 112
 rugalmas elrendezés 329

S, Sz

Salesforce.com 358
 sans-serif 93
 Save as JPEG 161
 Save Background As 155
 Save Image As 155
 Save Picture As 155
 Scale Image 159
 screen 336
 scroll 261
 scrolling 220
 search engine spamming 394
 segédalkalmazás 202
 select 363
 selected 368
 selector 61
 SEO 389, 393
 shape 192
 size 363
 solid 59, 148
 sorátívelés 112
 sorkizárt 84
 sormagasság 60
 sorok 103
 soron belüli elem 253
 sortörés 35
 spanning 112
 speech 336
 src 174, 353
 Starbucks 381
 stílusazonosítók 64
 stíluslap létrehozása 52
 stíluslapfájl 50

stíluslapok 49-50
 stílusosztályok 61
 stílus szabályok 50, 55
 stílustulajdonság 61
 stílustulajdonságok 50, 57
 style 73
 styles.css 55
 subject 130
 submit 359, 369
 szájhagyomány 390
 számozás 351
 szegély 58, 104, 250
 szélesség 58, 107, 178, 251
 személtjelző 394
 szín visszaállítása 160
 színek száma 164
 színelmélet 142
 színerék 142
 színeskorrekció 160
 színmélység 155
 színösszeállítás 147
 színösszetevők 146
 színpalettás képek 165
 színséma-generátor 143
 színsémák 143
 szintévesztők 149
 színválasztás 142
 szögletes zárójel 365
 szöközőket tartalmazó fájlnevek 137
 szöveg behúzása 78
 szöveg formázása 37
 Szöveg igazítása 72
 szövegbekezdés 35
 szöveges adatok 363
 szövegformázás 87, 89
 szövegközi keret 221
 szövegmező 363
 szövegszerkesztő 14, 29
 szülőelem 78, 267
 szülőkönyvtár 121

T, Ty

táblázat 103
 táblázat háttérképe 113
 táblázatcím 105
 táblázatok szegélye 104
 táblázatos elrendezés 113
 tag 31, 122
 tagolás 38
 találati rangsor 393
 talpas betűtípusok 334
 target 132, 219

tárgy 131
 tárhelyszolgáltatás 29
 tárolóelem 35
 tartalom 251
 tartalom elrejtése 304
 tartalomjegyzék 375
 tartománynév 129
 teljes cím 129
 teljes magasság 252
 teljes szélesség 252
 terület 190
 text-align 60, 74, 110, 179, 242
 textarea 363
 text-decoration 61
 TextEdit 29
 text-indent 60
 The World's Worst Website 144
 this 308
 title 40, 175, 178, 298
 több szóköz 106
 többsoros szövegmező 368
 többszintű függőleges navigációs sáv 286
 többszintű lista 80
 többszintű stíluslapok 49
 tömb 349
 tömörítés 156, 161
 tooltip 177, 298
 törzs 35
 Transparency 164
 triadikus 143
 tty 336
 túlfolyás 260
 tv 336
 Twitter 390
 type 363

U, Ü

ügyféloldali parancsfájlok 346
 uiMode 205
 új ablak megnyitása 132
 új kép létrehozása 163
 üres címke 34
 üres elemek 34, 36
 űrlap létrehozása 359
 űrlapadatok elküldése 369
 űrlapelemek elnevezése 364
 űrlapgomb 359
 űrlapkezelő rendszer 358
 űrlapok 357
 usemap 192
 úsztatás 179, 230, 243, 246

V

választó 61
 választógomb 367
 választólisták 367
 váltakozó színű sorok 118
 változó szélesség 219
 value 367, 369
 VBScript 347
 véletlenszerű tartalom 349
 vertical-align 110, 181, 242
 visible 261
 visited 133
 viszonyított cím 120
 vízszintes görgetés 162
 vízszintes igazítás 179
 vízszintes ismétlés 186
 vízszintes navigációs sáv 290
 vízszintes vonal 37
 vörösszem-hatás 160

W

W3C Markup Validation Service 41
 webcímek 120
 webes tartalom 3
 webgazda 6
 webhely 10
 webnapló 23
 weboldal 10
 webszínek 143
 width 58, 107, 178, 251
 Windows Media Player 204
 WMV 205
 WordPress 23, 314
 World Wide Web 2
 wrapper 314

X

XHTML 3, 31, 44
 XHTML 1.1 33
 XML 43
 xml:lang 34
 xmlns 34

Y

Yahoo! Search 393
 YouTube 208

Z, Zs

zárócímke 33
 z-index 254, 258, 300