Opera Software

Webes szabványok

0.1. verzió

Szerkesztette: Nagy Gusztáv
2010. február

Jogi nyilatkozat

Ez a jegyzet a http://dev.opera.com/articles/view/1-bevezeto-a-webes-szabvanyokba/ címen elérhető tananyagsorozat szerkesztett változata.

Szerzői: Chris Mills, Ben Buchanan, Tom Hughes-Croucher, Mark Norman “Norm” Francis, Linda Goin, Paul Haine, Jen Hanen, Benjamin Hawkes-Lewis, Ben Henick, Christian Heilmann, Roger Johansson, Peter-Paul Koch, Jonathan Lane, Tommy Olsson, Nicole Sullivan és Mike West.

A szerkesztett jegyzet mindenkori legfrissebb verziója a http://nagygusztav.hu címről tölthető le.

Tartalomjegyzék

1. A szabványok világa

1.1. Miért használd a webes szabványokat?

A fő okok arra, hogy miért érdemes bevonni a webes szabványokat a fejlesztésbe, majd csak a negyedik fejezetben lesz részletesen elemezve, de egy átfogó képet már itt is felállíthatunk. A webes szabványok használatának a következő előnyei vannak:

1. **Hatékony kód**: ahogy haladsz majd a leírásokban, észreveheted, hogy legjobb módszerek a webfejlesztésben sokszor a kód újra használhatóságára alapoznak — a HTML tartalmat **szétválaszthatod** a stílustól (CSS) és a működéstől (JavaScript), így a fájlok kisebbek maradnak, a kódot csak egyszer kell megírni, és később bárhol felhasználhatod, ahol újra szükség van rá.
2. **Egyszerű karbantartás**: ez nagyon közel áll az előző ponthoz — ha a HTML részt csak egyszer írod meg, aztán a stílusokat és a működést csak akkor definiálod (osztályokkal és függvényekkel), amikor szükséged van rájuk, akkor egy későbbi időpontban elég lesz a változást **egyetlen helyen** elvégezned, és ez az egész webhelyen azonnal megváltozik, ahelyett, hogy mindenütt egyenként elvégezd a módosítást!
3. **Hozzáférés**: a következő két pont ismét összetartozik — az egyik legnagyobb probléma a weboldalakkal az, hogy hozzáférhetővé tegyük mindenki számára, függetlenül attól, hogy kiről és milyen körülményekről van szó. Ebbe beletartozik az olyan weboldalak készítése is, amelyeket fogyatékos személyek is használnak, mint például vakok vagy gyengén látók, mozgássérültek (akik nehezen, vagy egyáltalán nem tudják használni a kezeiket). Ha webes szabványokat és a legjobb módszereket használod, akkor képes leszel olyan weboldalakat készíteni, amelyeket a fogyatékkal élők is használhatnak, ráadásul **semmilyen extra munkára** nem lesz szükséged ehhez.
4. **Kompatibilitás**: ez alatt azt értem, hogy a weboldalaid nem csak a különböző platformokon (mint például Windows, Mac vagy Linux) fognak megbízhatóan működni, hanem a különböző eszközökön is, amelyekbe manapság beletartoznak a mobiltelefonok, a tévék és a játékkonzolok is. Ezeknek az eszközöknek különböző korlátai vannak, mint például a képernyőméret, a számítási kapacitás, az irányíthatóság vagy sok más egyéb, de a jó hír az, hogy a webes szabványok és a legjobb módszerek használatával szinte egészen biztosan **garantálható**, hogy a weboldalad a legtöbb ilyen eszközön is működni fog. Jelenleg több mobiltelefon van a világon, mint PC, és ezek nagy része internetképes. Biztosan megengedheted magadnak, hogy egy ekkora piacot teljesen figyelmen kívül hagyj?
5. **Webes keresők és keresőrobotok**: ez alatt azt értem, amire sok helyen keresőoptimalizálás névvel hivatkoznak. Ismét csak azt tudom mondani, hogy ha a webes szabványokat és a legjobb módszereket használod, akkor a weboldalad a lehető legátláthatóbb lesz a keresőrobotok számára, amelyek pásztázzák a weboldalakat, így az oldalad jobb eredményt fog elérni az olyan keresőkben, mint például a Google. Ennek már saját tudománya van, a SEO, de ismétlem, már csak annyival, hogy webes szabványokat használsz, máris **sokat tettél a pozíciódért** a keresők találati listájában.

A fenti előnyök ellenére a legtöbb weboldal a weben mégsem követi a webes szabványokat, és rengeteg webfejlesztő még ma is régi, elavult módszerekkel dolgozik szerte a világon. Vajon miért? Ennek sok oka van — sokszor az oktatás hiányosságára vagy a cégük

házirendjére hivatkoznak, esetleg hogy nincs szükségük megtanulni a webes szabványokat, mert anélkül is megkapják a fizetést, túl nehéz megtanulni a szabványok használatát… Nézzük meg kicsit részletesebben ezeket az okokat, hogy eloszlathassuk a kifogásokban felmerült félreértéseket.

1. **Az oktatás hiánya**: ez egy elég nyomós érv, és éppen ez volt a fő oka annak, hogy ez a sorozat elindult. Az egyetemek többsége egyáltalán nem foglalkozik a webes szabványokkal a webfejlesztéssel kapcsolatos kurzusokon, de ha mégis, akkor leginkább csak régi, elavult módszereket tanítanak, és ezen bürokratikus okok miatt általában nehéz változtatni. A könyvek és a speciális oktatások általában drágák. De várj csak! Éppen **itt van egy kurzus**, ami ingyenes, elérhető az egyetemek számára is, így ez már nem lehet valódi kifogás többé.
2. **Céges policy**: semmi kétség nincs afelől, hogy rengeteg olyan cég létezik, amelynek régi és elavult a weboldala, ráadásul a házirendben arra kényszerítik az alkalmazottakat, hogy kiöregedett böngészőket használjanak. De a helyzet javul. Most, hogy ingyen elérhető ez a kurzus, amely megmutatja, hogyan léphetnek át ezen a helyzeten, még könnyebb lesz a változás. A webes szabványok használata ráveheti arra is a cégeket, hogy modernebb böngészőket is használjanak, mert a régi böngészőkben a szabványos oldalak már nem mutatnak olyan jól — bár továbbra is működnek azokban is. A felhasználóikat is rávehetik ezáltal a frissítésre. Üzleti előnye is van a váltásnak: azok a weblapok, amelyek webes szabványokat használnak — amint azt fentebb már kifejtettük — jobb eredményeket érnek el a keresőkben, és elérhetőek lesznek a fogyatékkal élő emberek, valamint a más eszközökön internetezők számára is. **Biztosan megengedhetik maguknak** a cégek, hogy ezeket az előnyöket figyelmen kívül hagyják?
3. **Nincs rájuk szükségem!**: tudom, hogy néhány fejlesztő azt mondja minderre, hogy „de hát én még régi módszereket használok, és mégis megfizetnek — miért vacakoljak ezekkel az új dolgokkal?” Ahogy már fentebb is kifejtettük, a webes szabványokkal a kód sokkal hatékonyabb lesz, könnyebb megírni és könnyebb karbantartani is. Lehetőséged lesz olyan modern kódot írni, amely mindenhol hozzáférhető és használható az alternatív eszközökön is — mindez nem elég izgalmas számodra? De a **képességeidet is használhatóbbá teszi** a jövőre nézve, így lehetőséged lesz többet keresni. Sok cég már most is megköveteli a webes szabványokban való jártasságot.
4. **Túl nehéz megtanulni!**: badarság. Ha átolvasol néhány leírást ebből a kurzusból, rá fogsz jönni, mennyire **egyszerű megérteni** a webes szabványok alapjait, akár új vagy a webfejlesztésben, akár csak a tudásodat frissíted. Semmivel sem nehezebb, mint a régi, elavult módszerek használata, viszont rengeteg előnye van ezekkel szemben.
5. **Nem minden böngésző támogatja a szabványt**: a szabványtámogatás a böngészőkben sokszor jelentősen eltért egymástól, és ez valódi rémálommá változtatta a munkát. De ezeknek az időknek már vége, a modern böngészők mindegyike kiváló szabványtámogatással rendelkezik. Természetesen szükség van arra, hogy régebbi böngészőket is támogassunk, amelyeknek nincs olyan jó szabványtámogatásuk. De ha **a legjobb módszereket használod**, biztos lehetsz benne, hogy az ilyen régi böngészők felhasználói is még megfelelő támogatást kaphassanak.

Amint láthatod, nem sok kifogásod maradt arra, hogy miért ne használd a webes szabványokat a munkád során. Ha kezdőként érkeztél ide, akkor máris sokkal jobb helyzetből

indulsz, mert azonnal a legjobb módszereket tanulhatod meg, és nem kell előtte „elfelejtened” a régieket.

1.2. A webes szabványok eljövetele

A böngészőháború alatt a Microsoft és Netscape főleg az új funkciók fejlesztésére koncentrált ahelyett, hogy a már támogatott funkciókat javították volna ki. Saját, védett funkciókat építettek be a böngészőkbe, emellett olyan új funkciókat valósítottak meg, amelyek már léteztek a többi böngészőkben, de ezek nem voltak kompatibilisek egymással.

A webfejlesztők ezekben az időkben egyre inkább arra voltak kényszerítve, hogy a valódi fejlesztés helyett az ilyen zavaros helyzetek megoldásával foglalkozzanak. Néha arra volt szükség, hogy két változatot készítsenek egy oldalból a két nagy böngésző számára, amelyek gyakorlatilag teljesen megegyeztek. Egyesek egyszerűen csak az egyik böngészőt támogatták, és a másik böngésző felhasználóit letiltották az oldalról. A munka így igazi rémálommá vált, és már nem voltak messze az elkerülhetetlen következmények a fejlesztők részéről.

1.2.1 A W3C megalakulása

1994-ben Tim Berners-Lee megalapította a **W3C**[1]-t (World Wide Web Consortium) a Massachusetts-i Technológiai Intézetnél, a CERN, a DARPA (ez lett az ARPA új neve), valamint az Európai Bizottság támogatásával. A W3C a különböző protokollok és technológiák szabványosítását tűzte ki maga elé egy olyan web felépítésének érdekében, amelyen az információ a világ népességének lehető legnagyobb része számára elérhetővé válhat.

A következő években a W3C több specifikációt is kiadott (amelyeket „ajánlásoknak” neveztek), mint például a HTML 4.0, a PNG képek szerkezete, vagy a CSS első és második verziója.

Viszont a W3C nem követeli meg az ajánlások betartását. A gyártók csak akkor kell betartsák a W3C dokumentumokban foglaltakat, ha a terméküket meg akarják jelölni, hogy megfelel a W3C feltételeinek. A gyakorlatban ez nem értékes eladási szempont, mivel a felhasználók közül szinte senki sem tudja, sőt valószínűleg nem is érdekli őket, hogy mi fán terem a W3C. Ennek következtében a „böngészők háborúja” zavartalanul folytatódhatott.

1.2.2 A Web Standards projekt

1998-ban a böngészőpiacot az Internet Explorer 4 és a Netscape Navigator 4 uralta. Megjelent az Internet Explorer 5 beta verziója, amely már egy új és saját fejlesztésű dinamikus HTML-t támogatott. Ez azt jelentette, hogy a professzionális webfejlesztők már öt különböző módon kellett tudjanak JavaScript kódokat írni.

Ennek eredményeképpen a professzionális webfejlesztők és webdesignerek egy csoportja összeállít, és együtt megalakították a **WaSP**[2]-ot (Web Standards Project). Az ötlet alapjában az volt, hogy a W3C dokumentumait szabványnak nevezzék ajánlások helyett,

[1] http://www.w3.org/ és http://www.w3c.hu/

[2] http://www.webstandards.org/

így talán sikerülhet meggyőzni a Microsoftot és a Netscape-et arról, hogy támogassák őket.

1.2.3 A webes szabványok felemelkedése

2000-ben a Microsoft kiadta az Internet Explorer 5 Macintosh Editiont. Ez egy nagyon fontos mérföldkő volt, mivel ez lett az alapértelmezett böngésző a Mac OS alatt, és már támogatta a W3C ajánlások egy jelentős részét. Ez az Opera megfelelő CSS és HTML támogatásával együtt már megadta a lehetőséget a webfejlesztők és webdesignerek számára, hogy először tapasztalhassák meg a szabványos weboldalak fejlesztésének a kényelmét.

A WaSP ráadásul sikerült meggyőzze a Netscape-et arról, hogy halassza el a Netscape Navigator 5.0 megjelenését addig, amíg nem támogatja jobban a szabványokat (ez a változat vált később a manapság nagyon népszerű böngésző, a Firefox alapjává). A WaSP ezen kívül indított egy **Dreamweaver Task Force** nevű csoportot is, hogy rávegye a Macromediát a népszerű fejlesztői eszközük szabványosabbá tételére.

2001-ben az **A List Apart**[3] nevű népszerű webfejlesztői oldalt is teljesen újratervezték, és az egyik cikkben, amely a miértekről és hogyanokról szólt, ez állt:

Hat hónapon, egy éven vagy legkésőbb két éven belül minden oldalt ezekkel a szabványokkal fognak készíteni. [...] Figyelhetjük, ahogy a tudásunk fölöslegessé válik, vagy elkezdhetjük már most tanulni a szabványos módszereket.

Ez egy kissé optimista kijelentés volt, az összes oldal még most, 2008-ban sem támogatja a szabványokat. De sokan hallgattak rájuk. A régi böngészők részesedése lecsökkent, és később két másik rendkívül népszerű oldalt is átalakítottak teljesen szabványosra: 2002-ben a Wired magazint, és 2003-ban az ESPN-t, akik így a webes szabványok és az új technológiák élharcosaivá váltak.

Ugyancsak 2003-ban Dave Shea elindította a **CSS Zen Garden**[4] oldalt. Ez valószínűleg nagyobb hatást ért el a webfejlesztőknél, mint addig bármi más, ugyanis képes volt azt bemutatni, hogy egy oldal teljes designját meg lehet változtatni egyszerűen a stílus lecserélésével, miközben a tartalom pontosan ugyanaz marad.

Azóta a professzionális webfejlesztők körében a webes szabványok alapvetővé váltak. Ez a sorozat most nagyszerű alapot adhat neked is ahhoz, hogy ugyanolyan tiszta, szemantikus, hozzáférhető és szabványos weboldalakat készíthess, mint a nagy vállalatok.

1.2.4 Tesztkérdések

Keress tovább a témában, hogy megtaláld a válaszokat az alábbi kérdésekre:

- Milyen böngészők érhetőek el manapság az interneten Windows, Mac OS X és Linux alatt?
- Hány százalékos részesedésük van ezeknek a böngészőknek?
- Milyen böngészőket használhatnak mobiltelefonokon a weblapok megnyitására?
- Hány „webes szabványt” publikált a W3C, és melyek azok, amelyeket a mai böngészők széles körben támogatnak?

---

[3] http://www.alistapart.com/

[4] http://www.csszengarden.com/

1.3. Hogyan működik az internet?

Néha előfordul, hogy szeretnél betekintést nyerni a dolgok mögé, hogy lásd a fogaskerekeket és az ékszíjakat az események mögött. A mai a te szerencsés napod. Bevezetlek a legújabb technológiák működésébe, melyek segítségével már elboldogulsz a világhálóval.

Ez a leírás az alapvető technológiákról szól, melyek a világhálót éltetik:

- Hiperszöveg jelölőnyelv (HTML)
- Hiperszöveg-átviteli protokoll (HTTP)
- Domainnév rendszer (DNS)
- Webszerverek és internetböngészők
- Statikus és dinamikus tartalom

Ezek mind nagyon alapvető dolgok – még ha az itt leírtak nagy része nem is segít egy jobb weboldal elkészítésében, megadja a megfelelő nyelvezetet ahhoz, hogy az ügyfeleiddel vagy másokkal beszélgethess a webről. Ez olyan, mint ahogy egy bölcs apáca mondta A muzsika hangja c. filmben: „Mert az olvasást hogy kezdik? — Á, B, C; Az éneklést így kezdik: Dó, Ré, Mi”. Ebben a leírásban röviden áttekintjük, hogyan kommunikálnak egymással a számítógépek a HTTP és TCP/IP használatával, aztán megnézzük a különböző nyelveket, melyek együtt alkotják az internetet képező weboldalakat.

1.3.1 Hogyan kommunikálnak egymással a számítógépek az interneten keresztül?

Szerencsére a számítógépek dolgait nem bonyolítottuk el túlságosan. Amikor a világhálót nézzük, a legtöbb oldal ugyanazt a kódot használja, a HTML-t, amely egy közös szabályrendszer, a HTTP (hiperszöveg-átviteli protokoll) alapján működik. A HTTP az internet általános „nyelvjárása” (előírása), figyelembe véve például, hogy egy windowsos rendszer tökéletes összhangban legyen a legújabb és legjobb Linux verzióval (Dó, Ré, Mi!). Az internetböngészőkön túl – ezek speciális programok, melyek értelmezik a HTTP utasításokat és lefordítják a HTML kódot emberek számára olvasható formára – a weboldalakat azért írják HTML-ben, hogy bármilyen számítástechnikai eszközön képes legyél azt elolvasni, telefonon, PDA-n, vagy az egyre népszerűbb videojáték rendszereken.

Annak ellenére, hogy ugyanazt a nyelvet beszélik, a webet használó eszközöknek szükségük van szabályokra, hogy egy másik eszközzel kommunikálni tudjanak – ez olyan, mint megtanulni, hogy felemeled a kezed, mielőtt kérdezel az órán. A HTTP lefekteti ezeket az alapvető szabályokat az internet számára. A HTTP-nek köszönhetően egy kliens (mint pl. a számítógéped) tudja, hogy ő az, akinek kezdeményeznie kell a kérést egy weboldal vagy egy szerver felé. A szerver egy számítógép, amely olyan programokat futtat, melyek érzékelik a kérésed, megkeresik a kívánt weboldalt, és elküldik a számítógépedre, hogy a böngésződ megjeleníthesse azt.

A kérés/válasz kör elemzése

Most átnézzük azokat az eljárásokat, melyek lehetővé teszik, hogy a számítógépek az interneten keresztül kommunikáljanak. A HTTP kérés/válasz kört alaposabban is megvizsgáljuk. Lentebb van néhány számozott lépés, ezek mentén haladunk előre, így érthetőbben el tudom magyarázni a fogalmak jelentését.

Minden kérés/válasz úgy kezdődik, hogy **beírjuk az URL-t** (egységes erőforrás-azonosító) a böngészőnk címsorába, valahogy úgy, mint a http://dev.opera.com. Nyiss meg egy böngészőt, írd be ezt a címet!

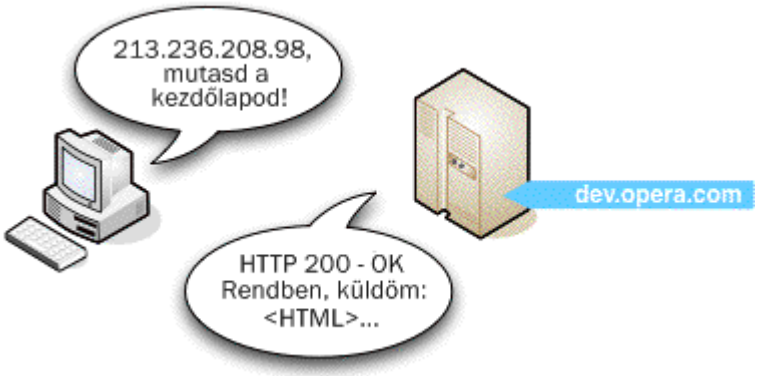
Most van egy dolog, amit lehet, hogy még nem tudsz: a böngészők nem arra használják az URL-eket, hogy lekérjék az oldalakat a szerverekről; **Internet Protokollt**, vagyis **IP** címeket használnak (mint a telefonszámok vagy az irányítószámok, csak ezek a címek szervereket azonosítanak). Például a http://dev.opera.com IP címe a 213.236.208.98.

Próbálj megnyitni egy **új fület** vagy új ablakot a böngésződön, majd írd be: http://www.apple.com, majd üss entert; aztán írd be: http://17.149.160.10/, és üss entert – pontosan ugyanoda jutsz. Most írd be a http://213.236.208.98/ címet – ugyanarra a szerverre jutunk, mint az 1. lépésben, habár most 403 „Belépés visszautasítva” hibát kapunk – ez azért van, mert nem rendelkezünk belépési engedéllyel a kiszolgáló gyökerébe.

A http://www.apple.com alapvetően ugyanazt az eredményt adja, mint a http://17.149.160.10/, de miért, és hogyan? Ez azért van, mert az emberek jobban emlékeznek szavakra, mint hosszú számsorokra. A rendszer, ami a munkát elvégzi, a DNS (domainnév rendszer), amely alapvetően egy átfogó, automatikus címtár, ami minden internetre csatlakozó eszköz címét eltárolja. Amikor beütöd a http://dev.opera.com címet a címsorba és leütöd az entert, a böngésző elküldi ezt a címet egy névszervernek, ami megpróbálja meghatározni a hozzá tartozó IP-címet. Készülékek garmadái csatlakoznak egyszerre a világhálóra, és nincs minden DNS szerveren listázva az összes hálóra kötött készülék, ezért van egy rendszer, ahol a kérésed végrehajtódik, hogy a megfelelő szerver teljesíteni tudja azt.

Tehát a DNS rendszer utána néz a www.opera.com weboldalnak és megtalálja, hogy az a 17.149.160.10 címen található, majd ezt a címet visszaküldi a böngésződnek.

A géped küld egy kérést az IP cím által megjelölt géphez, és várja, hogy visszajöjjön a válasz. Ha minden megfelelően működik, a szervergép visszaküld egy rövid üzenetet a kliensnek, hogy minden rendben, majd küldi a weblapot. Az üzenet ezen formáját egy HTTP fejléc tartalmazza.



1.1. ábra: Ebben az esetben minden rendben megy, a szerver a megfelelő weblapot küldi el

Ha valami hiba történik, például rosszul írod be az URL-t, egy HTTP hibát kap a böngésződ — a népszerűtlen 404 „Az oldal nem található” hiba a legáltalánosabb példa erre, amellyel találkozhatsz.

Most próbáld beírni a http://dev.opera.com/joniscool.html címet — az oldal nem létezik, így 404-es hibaüzenetet kapsz. Próbáld ki még pár nem létező lappal, különböző

weboldalakon, így különböző formában találkozhatsz a hibával. Ez azért van, mert néhány weboldal hiba esetén a szerver saját hibalapjára irányítja az eltévedt böngészőket, míg más oldalaknak saját, egyedi hibaüzenetei vannak, amelyekkel jelzik, ha hibás a hivatkozás. Ez egy különleges eljárás, amit ezen a kurzuson nem részletezünk, de szerencsére később, egy különálló leírásban lesz róla szó a dev.opera.com-on.

A végén még egy apróság az URL-ekről: általában egy weblap látogatásakor megnyitott első lap URL-ének végén nem látsz fájlnevet (pl. http://www.enoldalam.hu/), aztán később is vagy vannak fájlok a végén, vagy továbbra sincsenek. Valójában minden esetben fájlokat nyitsz meg, csak néha a webfejlesztők úgy állítják be a kiszolgálókat, hogy azok ne mutassák a fájlnevet az URL-ben — ez gyakran egyszerűbbé, könnyebben megjegyezhetővé teszi az URL-eket, ami elsősorban a felhasználói élményt növeli az oldaladon. Ezzel a témával sem fogjuk foglalkozni a tanfolyamon, mivel meglehetősen összetett dolog. A fájlok feltöltésével és a fájl/könyvtár struktúrával részletesen egy későbbi leírásban fogunk foglalkozni.

1.3.2 A tartalmak típusai

Megnéztük a HTTP kérés/válasz kört, most nézzük át a különböző típusú tartalmakat, amelyekkel az interneten találkozhatsz. 4 részre választottam szét ezeket — egyszerű szöveg, webes szabványok, dinamikus weblapok, és oylan elemek, melyekhez külső alkalmazásokra vagy beépülőkre van szükség.

Egyszerű szöveg

Az internet megjelenésekor, még a webes szabványok és a beépülők előtt az internet csak képekből és szövegekből állt — különböző fájlok .txt vagy valami hasonló kiterjesztéssel. Ha találkozott egy egyszerű szöveggel, a böngésző csak megjelenítette azt, úgy ahogy van, bonyolult eljárások nélkül. Manapság gyakran egyetemek weboldalain találkozhatsz ilyen egyszerű szöveggel.

Webes szabványok

A világháló alapvető építőkockája a három fő webes szabvány — a HTML (vagy XHTML, de ebben a leírásban ez most lényegtelen), a CSS és a JavaScript.

A **HTML**, mint „hiperszöveg jelölő nyelv” már a nevében érzékelteti a saját rendeltetését. A HTML-t régebben arra használták, hogy dokumentumokat tegyenek közzé, megadják ezek tartalmát és a felépítését, és definiálják a dokumentum különböző részeit (ezekben tárolják az összes weblap szövegét és más elemeit). A weblapok különböző részeinek azonosításához elemeket használ.

A **CSS** (egymásba ágyazott stíluslapok) teljes hozzáférést nyújt egy elem megjelenésének beállításához. Például nagyon egyszerűen, egyetlen stílus deklarációval beállíthatod, hogy minden paragrafus legyen dupla sorközű (line-height: 2em;), vagy hogy minden második szintű címsor legyen zöld (color: green;). Rengeteg előnye van annak, ha szétválasztod a tartalmat a formázástól. A HTML és a CSS közös használatát az 1.2. ábrán mutatjuk be, amelyen bal oldalon a sima HTML kódot láthatod formázás nélkül, míg jobb oldalon pontosan ugyanazt a HTML-t, de már CSS stílusokkal kiegészítve.

1.2. ábra: Sima HTML a bal oldalon, HTML és CSS stílusok együtt a jobb oldalon

Végül, a **JavaScript** dinamikus funkciókat biztosíthat a weblapodhoz. JavaScript segítségével olyan kis programkódokat írhatsz a weblaphoz, amelyek a kliens számítógépén fognak lefutni, és nincs szükség semmilyen speciális szoftver telepítésére a kiszolgálón. A JavaScript lehetőséget ad arra, hogy néhány egyszerű funkcionalitást megvalósíthass a weblapon, és interaktívvá tehesd, de természetesen vannak korlátai, amelyek a szerveroldali programozási nyelvekhez és a dinamikus weboldalakhoz vezetnek tovább.

Dinamikus weboldalak

Néha böngészés közben olyan oldalakra találhatsz, amelyeknél a weblapok kiterjesztése nem .html, hanem .php, .asp, .aspx, .jsp vagy valami más különös kiterjesztés. Ezek mind példák a dinamikus webtechnológiákra, ezekkel olyan weboldalakat lehet készíteni,amelyeknek van egy dinamikusan változó része — egy olyan kód, amely eredményeket szolgáltat a különböző bemenetekre (például adatbázisból, űrlapról vagy más adatforrásból). Ezekről bővebben az 1.3.3. fejezet alatt beszélünk majd.

Formátumok más alkalmazásokhoz és beépülőkhöz

Mivel a webböngészők alapvetően csak néhány megadott webes szabvány értelmezésére vannak felkészítve, így ha az URL egy komplexebb fájlformátumra mutat, vagy egy olyan weboldalra, amelyiknek beépülőkre van szüksége, akkor az ilyen tartalmat letöltheted, vagy a beépülők használatával (és ha szükséges, telepítésével) megnyithatod a böngészőben. Például:

1. Ha találsz egy Word dokumentumot, egy Excel fájlt, egy PDF-et, egy tömörített fájlt (pl. ZIP vagy SIT), egy komplex képet (pl. Photoshop PSD), vagy más olyan fájlt, amelyet a böngésző „nem ért”, akkor a böngésző az esetek többségében fel fogja ajánlani, hogy töltsd le vagy nyisd meg a fájlt a hozzárendelt alkalmazással. Mind a kettő hasonló megoldás, mivel a fájlt mindkét esetben le kell tölteni, és csak egy speciális alkalmazással nyithatod meg a böngészőn kívül.
2. Ha olyan weboldalt találsz, amelyik tartalmaz egy Flash videót, egy MP3-at vagy más zeneformátumot, MPEG vagy más videoformátumot, akkor a böngésző ezt képes neked a weboldalon lejátszani, ha a szükséges beépülő telepítve van. Ha nincs, akkor a böngésző megkérhet a kapcsolódó beépülő letöltésére és telepítésére, vagy az előző ponthoz hasonlóan felajánlja a fájl letöltését és megnyitását egy másik alkalmazásban.

1.3.3 Statikus kontra dinamikus oldalak

Szóval mik is azok a statikus és dinamikus weboldalak, és mi a különbség közöttük? Hasonlóan egy doboz csokoládéhoz, minden azon múlik, hogy mivel vannak megtöltve.

Egy statikus weboldal egy olyan website, ahol a tartalom, a HTML és a képek mindig statikusak, változatlanok — minden látogatónak pontosan ugyanazt küldi el minden alkalommal, kivéve azt az esetet, ha a weboldal készítője gondol egyet, és megváltoztatja a lapot magán a kiszolgálón. A cikk nagy részében erről az esetről volt szó.

A dinamikus oldalak ezzel szemben, bár a tartalom a kiszolgálón változatlan, az egyszerű HTML mellett dinamikus kódot is tartalmaznak, amely bizonyos információktól függően más és más adatokat jeleníthet meg. Nézzünk egy példát: nyisd meg a www.amazon.com oldalt a böngésződben, és keress rá 5 különböző termékre. Az Amazon nem 5 különböző weboldalt küldött neked eredményként; mind az 5 esetben ugyanazt az oldalt küldte át, csak más és más dinamikus információkkal feltöltve. A különböző információkat egy adatbázisban tárolja, amely kérésre kiadja a releváns adatokat, és átadja a webszervernek, hogy az elküldhesse a dinamikus lapot.

Érdemes még megjegyezni a dinamikus oldalakkal kapcsolatban, hogy a kiszolgálón speciális szoftvereket kell telepíteni a használatukhoz. Míg a normál statikus HTML fájlok kiterjesztése .html, addig a speciális dinamikus oldalak kiterjesztése ettől eltér, mivel a kiterjesztés alapján a kiszolgáló látni fogja, hogy további módosításokat kell végrehajtania a fájlon, mielőtt elküldi azt a kliensnek (például fel kell töltenie egy adatbázisból). A PHP fájlok például .php kiterjesztést kapnak.

Több dinamikus programozási nyelvből is lehet választani, megemlítettük például a PHP-t, de többek között ilyen még a Python, a Ruby on Rails, az ASP.NET és a Coldfusion. Ezeknek a nyelveknek nagyjából ugyanolyan lehetőségeik vannak, például adatbáziselérés, információ validálása, de ezeket különböző módokon teszik meg, így vannak előnyeik és hátrányaik is. Minden attól függ, hogy mi felel meg jobban a számodra.

1.3.4 Tesztkérdések

- Mi az a HTML és a HTTP? Mi a különbség közöttük?
- Mire szolgálnak a webböngészők?
- Nézz körül az interneten, és próbálj meg 5-10 perc alatt minél többféle tartalmat találni: egyszerű szöveget, képeket, HTML-t, dinamikus oldalakat (PHP-t és .NET-es aspx lapokat), PDF-et, Word dokumentumokat, Flash videókat, stb. Próbáld ezeket megnyitni, és gondold végig, hogy a böngésződ milyen módon jeleníti meg ezeket a számodra.
- Mi a különbség a statikus és a dinamikus weblapok között?
- Keresd meg a HTTP hibák listáját, válassz ki belőlük 5-öt, és magyarázd meg, hogy ezek mit jelentenek.

1.4. A webes szabványok modellje — HTML, CSS és JavaScript

Az előzőekben röviden érintettük a web alapköveit, a HTML-t (vagy XHTML-t), a CSS-t és a JavaScriptet. Most ideje egy kicsit mélyebbre ásni és megnézni, hogy mire jók ezek,

mit csinálnak, hogyan tudnak együttműködni, és hogyan építhetünk fel velük egy weblapot.

1.4.1 Miért kell szétválasztani?

Általában ez az első kérdés, amelyet a webes szabványokkal kapcsolatban kérdezni szoktak. Csak a HTML használatával megadhatod a tartalmat, a stílust és az elrendezést is, a font elemet használhatod a stílusozáshoz, a HTML táblázatokat az elrendezéshez, szóval miért törődnél ezekkel az XHTML meg CSS dolgokkal? A rendezéshez a táblázat meg az ehhez hasonlók pontosan azok a technikák, amelyeket a régi, rossz időkben használtak a webdesignhoz, és sokan még most is ezeket használják (pedig már nagyon nem kellene). Ez az egyik elsődleges oka annak, hogy ezt a sorozatot elkészítettük. Ezekről a módszerekről nem fogunk beszélni ezen a kurzuson. Itt vannak viszont a CSS és HTML használatának az ellenállhatatlan előnyei a régi módszerekkel szemben:

1. **Hatékony kód**: minél nagyobbak a fájlok, annál tovább tart a letöltésük, és annál többe kerül ez egyes embereknek (jópáran most is a forgalom után fizetnek). Biztosan nem akarod elpazarolni a sávszélességet hatalmas lapokra, amelyeknél minden egyes HTML fájlban külön benne van a stílus és az elrendezés is. Sokkal jobb alternatíva, ha a HTML-t lecsupaszítod a minimumra, és a stílust meg az elrendezést csak egyszer adod meg egy (vagy több) különálló CSS fájlban. Ha kíváncsi vagy egy valódi példára, nézd meg az A List Apart Slahsdot rewrite cikket, amelyben a szerző elővett egy népszerű weblapot, és átírta XHTML/CSS formára.
2. **Könnyű karbantartás**: az előző gondolatmenetet folytatva, ha a stílust és az elrendezést csak egyszer írod meg, akkor ez azt jelenti, hogy elég lesz csak egyetlen helyen változtatni, ha módosítani akarod a teljes website megjelenését. Vagy inkább minden egyes oldalt külön frissítenél? Nem hinném.
3. **Hozzáférhetőség**: azok a felhasználók, akiknek látási problémáik vannak, általában egy „képernyő-felolvasónak” nevezett szoftvert használnak ahhoz, hogy a weblapok információihoz hozzáférjenek. A szoftver felolvassa nekik a weblap tartalmát, ők pedig meghallgatják. Ezen kívül a hangfelismerő szoftvereknek, amelyeket a mozgássérült vagy rokkant emberek használhatnak, ugyancsak sokat segíthetnek a jól megszerkesztett, sematikus weblapok. Míg a képernyő-felolvasót használók billentyűparancsokkal navigálhatnak a lap fejlécei között, addig a mozgássérült hangparancsokat ad ki a navigáláshoz. A megjelenésre összerakott weblapokkal szemben az egyszerű, sematikus weblapokon sokkal egyszerűbb navigálni, és az információkat is könnyebben találják meg a felhasználók. Más szavakkal, minél hamarabb találod meg a lényeget (a tartalmat), annál jobb. A képernyő-felolvasók nem találják meg azokat a szövegeket, amelyek képeken vannak, és a JavaScript egyes felhasználási módjai is összezavarhatják. Mindig bizonyosodj meg róla, hogy a fontos tartalom mindenki számára elérhető.
4. **Eszköz kompatibilitás**: mivel az XHTML oldalad csak egyszerű jelölés, stílus nélkül, így egyszerűen átformázható a különböző tulajdonságokkal rendelkező eszközökön (mint például kisebb képernyő), mert elég hozzá egy alternatív stílust megadni. Ezt több módon is megteheted (ha ez bővebben is érdekel, olvasd el a mobil eszközökről szóló leírásokat a dev.opera.com-on). A CSS natívan támogatja a különböző stílusok használatát a különböző megjelenítési módszerek/eszközök számára (például képernyőre, nyomtatáshoz vagy mobiltelefonhoz).
5. **Webes keresők és keresőrobotok**: valószínűleg te is szeretnéd, hogy az oldaladat könnyen meg lehessen találni a Google-lel vagy más keresőkkel. A keresők

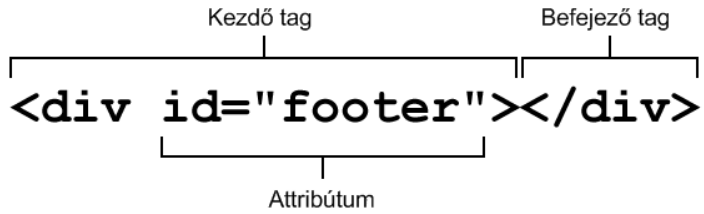
ún. „keresőrobotokat” használnak, amelyek átnézik és letapogatják a weblapjaidat. Ha a keresőrobotnak gondja van a hasznos tartalom megtalálásával a weblapodon, vagy félreérti az információkat, mert azokat nem jelölted megfelelően (például a címeket nem jelölted címként), akkor valószínűleg vissza fogsz esni a találati listában.

6. **Egyszerűen jó módszer**: ez most egy kissé olyan „mert én úgy mondom” indok, de kérdezz csak meg egy professzionális webfejlesztőt vagy webdesignert, és azt fogja mondani, hogy a tartalom, a stílus és a működés szétválasztása a legjobb módja egy alkalmazás fejlesztésének.

1.4.2 Jelölés, minden weblap alapja

A HTML és az XHTML különböző elemekből álló jelölő nyelvek, amelyek tartalmazhatnak attribútumokat is (egyeseket kötelezően, másokat választhatóan). Ezek az elemek jelölik a különböző típusú tartalmakat a dokumentumokban, és megadják, hogy hogyan értelmezzék a webböngészők a különböző tartalomdarabokat (ilyenek például a címsorok, a bekezdések, a táblázatok, a felsorolások, stb.)

Ahogy az elvárható, az elemek egy bizonyos tartalomtípust definiálnak, míg az attribútumok további információkat adnak az elemekhez, például azonosítanak egy bizonyos elemet, vagy megadják, hogy hova mutat egy hivatkozás. Mindig tartsd észben, hogy a jelölés a lehető legegyértelműbb kell maradjon, és olyan pontosan kell leírja a tartalom rendeltetését, amennyire csak lehetséges.



1.3. ábra: Egy (X)HTML elem felépítése

Egy HTML elem felépítése (1.3. ábra): az első kisebb jel, az első nagyobb jel és a köztük található betűhalmaz a kezdő tag, ezzel kezdődik el az elem. A következő betűhalmaz, amelyet egy egyenlőségjel követ, és azt ezt követő idézőjelekben található betűhalmaz az attribútum. A HTML elemek tartalmazhatnak egy vagy több attribútumot, illetve bizonyos esetekben egyet sem. A második kisebb jel, amelyet egy perjel követ, a második nagyobb jel, és a köztük található betűhalmaz a befejező tag, ezzel végződik az elem. A két tag, valamint a köztük található szöveg maga az elem.

A fentieket észben tartva végül is mi a különbség a HTML és az XHTML között?

Mi az az XHTML?

Az „X” az XHTML-ben azt jelenti, hogy „extensible”, vagyis bővíthető. A kezdők egyik leggyakoribb kérdése éppen erre vonatkozik, hogy „most akkor HTML-t vagy XHTML-t használjak, és mi a bánat a különbség a kettő között?”. Nagyjából ugyanarra a dologra szolgál mind a kettő, a különbség a struktúrában rejlik. A következő táblázat megmutatja a legfontosabb különbségeket.

HTML	XHTML

Az elemek és attribútumok nem érzékenyek a kis- és nagybetűkre, a <*h1*> ugyanaz, mint a <*H1*>.	Az elemek és attribútumok érzékenyek a nagybetűkre, mindent kisbetűvel kell írni.
Egyes elemekhez nem szükséges a bezáró tag (például a paragrafusoknál elég a <*p*>), míg más esetekben (az ún. „üres elemeknél”) nem szabad lezárni a taget (pl. a képeknél, <*img*>).	Minden elemet mindig le kell zárni (pl. <*p*>Egy paragrafus</*p*>). A tartalom nélküli elemeket a kezdő tag végére írt perjellel is le lehet zárni (pl. a <*hr*></*hr*> és a <*hr/*> pontosan ugyanaz). Ha az XHTML-t *text/html* formában biztosítod, akkor az összes „üres” elemnél a rövid formát kell használnod, és a lezáró perjel előtt pedig ki kell hagyni egy space-t. Minden más elem esetében a hosszú formát kell használnod (különálló kezdő és befejező tagekkkel), még akkor is, ha nincs benne semmilyen tartalom.
Egyes attribútumok értékeit idézőjelek nélkül is meg lehet adni.	Az attribútumok értékeit mindig idézőjelekbe kell tenni.
Az attribútumokat bizonyos esetekben lerövidítheted (pl. <*option selected*>).	Mindig meg kell adni a teljes attribútumot (pl. <*option selected="selected"*>).
A kiszolgálók a HTML-t a *text/html* médiatípussal küldik a felhasználónak.	Az XHTML az *application/xhtml+xml* médiatípust használja, de használhatja még az *application/xml*, a *text/xml* vagy a *text/html* típusokat is. A text/html esetében követni kell a HTML kompatibilitási útmutatót, mivel a böngészők HTML-ként fogják értelmezni (és ez alapján próbálják a felmerülő hibákat feloldani).

Egyelőre azt ajánljuk, hogy ne foglalkozz sokat azzal, hogy HTML-t vagy XHTML-t használj-e. Tartsd magad a kurzusban található példákhoz, és egyszerűen használj HTML doctype-ot (esetleg olvasd el a 14. leírást a doctype-okról), így nem igazán ronthatsz el semmit.

Mi az a validáció?

Mivel a HTML és az XHTML már szabványosítva van (és egyébként a CSS is), a W3C (World Wide Web Consortium) készített egy nagyszerű eszközt, egy validátort, amely képes leellenőrizni a weblapjaidat, és megmutatja a lapon talált hibákat vagy problémákat, mint például a hiányzó bezáró tagek vagy hiányzó idézőjelek az attribútumokban. A HTML validátor online elérhető a http://validator.w3.org/ címen. A validátor automatikusan felismeri, hogy HTML-t vagy XHTML-t, valamint hogy milyen doctype-ot használsz. Ha a CSS-t is ellenőrizni szeretnéd, akkor ehhez az online validátor a http://jigsaw.w3.org/css-validator/ címen érhető el.

1.4.3 CSS — kell egy kis stílus

A CSS (Cascading Style Sheets) lehetőséget ad a weboldalaid formázására és azok elrendezésének kialakítására. Beállíthatod vagy lecserélheted a színeket, a hátteret, a betűk méretét és típusát, és megadhatod a különböző elemek helyét a weblapon. Három módszer van arra, hogy a CSS segítségével készíts egy stílust: újradefiniálsz egy taget, stílust rendelsz egy azonosítóhoz (ID-hez), vagy stílust rendelsz egy osztályhoz. Nézzük meg ezeket sorban:

Egy elem újradefiniálása. Bármilyen (X)HTML elem megjelenését megváltoztathatod, ha készítesz hozzá egy új stílusszabályt. Ha azt szeretnéd, hogy az összes paragrafusod zöld és dupla sorközű legyen, akkor megadhatod az alábbi deklarációt a CSS fájlban:

```
p {
  line-height: 2;
  color: green;
}
```

Ezután minden tartalom a * * tagek között dupla sorközű és zöld színű lesz.

Egy azonosító definiálása. Egy elemhez hozzárendelheted az *id* attribútumot, hogy egyértelműen azonosítani tudd a weblapon (minden azonosító csak egyszer szerepelhet) — például *id="navigacios_menu"*. Így sokkal könnyebben felügyelheted ennek a bizonyos elemnek a megjelenését. Ha egy bizonyos paragrafust a lapon dupla sorközzel és zöld színnel szeretnél kiemelni, akkor először adj neki egy azonosítót (ID-t):

```
<p id="highlight">Paragrafus tartalma</p>
```

Majd készíts hozzá egy CSS szabályt a következő módon:

```
#highlight {
  line-height: 2;
  color: green;
}
```

Ez a CSS szabály kizárólag csak arra az egy paragrafusra vonatkozik, amelynek az id attribútumában a *highlight* érték van (a kettős kereszt a CSS jelölése az azonosítóra).

Egy osztály definiálása. Az osztály éppen olyan, mint az azonosító, kivéve, hogy a lapon egyszerre több elem is tartozhat ugyanahhoz az osztályhoz. Maradva a dupla sorközös példánál, ha azt szeretnéd, hogy az első két bekezdés a lapon dupla sorközű és zöld színű legyen, akkor először add meg nekik az osztályt:

```
<p class="highlight">Paragrafus tartalma</p>
<p class="highlight">A második paragrafus tartalma</p>
```

Majd készíts hozzá egy CSS szabályt a következő módon:

```
.highlight {
  line-height: 2;
  color: green;
}
```

Ebben az esetben a *highlight* egy osztály, és nem egy azonosító — erre utal a pont a CSS szabályban.

A következő példában bemutatjuk, hogy hogyan alkalmazza a CSS a stílusokat a HTML dokumentumon.

1.4.4 JavaScript — a weblapok működése

Végezetül, a JavaScript egy szkriptnyelv, amelyet arra használhatsz, hogy különböző funkcionalitást adj a weblapodhoz – például leellenőrizhetsz vele egy beírt szöveget (hogy jól adták-e meg vagy nem), adhatsz a weblapodhoz drag/drop funkcionalitást, lecserélheted vele gombnyomásra a lap stílusát, animálhatod a lap elemeit (például a menüket), lekezelheted a gombok eseményeit, de még millió és egy dologra használhatod. A modern JavaScriptek már úgy működnek, hogy megkeresik a célzott HTML elemet, és ezzel csinálnak valamit, hasonlóan a CSS-hez, de a szintaxis és maga a művelet teljesen más ebben az esetben.

A JavaScript sokkal bonyolultabb és terjedelmesebb téma, mint a HTML és a CSS, ezért az egyszerűség megőrzésének érdekében, valamint a félreértések elkerülése végett a következő példában nem fog szerepelni. Valójában a kurzuson is csak jóval később fogsz a JavaScripttel ismét találkozni.

1.4.5 Egy példa oldal

Több olyan részlet is van, amelyről most nem volt szó, de mindent részletesen meg fogsz ismerni a kurzus folyamán! Most pedig bemutatunk egy valódi példát, amelyen keresztül betekintést kaphatsz, hogy mivel fogunk foglalkozni a további bejegyzésekben.

A bemutatott példa egy referencia oldal, amelyet egy beszéd végén az elhangzott idézetek forrásának bemutatására használhatsz, amelyben a szélessávú internet használatáról volt szó az Egyesült Államokban. Ha ismerős számodra az akadémiai írásmód, akkor elárulom, hogy ez a példa APA formában készült (ki kellett választam valamelyiket).

index.html

```
<!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"
lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
  <title>References</title>
  <style type="text/css">
    @import url("styles.css");
  </style>
</head>

<body>
  <div id="bggraphic"></div>
  <div id="header">
    <h1>References</h1>
  </div>
```

```
  <div id="references">
    <cite class="article">Adams, J. R. (2008). The Benefits of
Valid Markup: A Post-Modernistic Approach to Developing Web
Sites. <em>The Journal of Awesome Web Standards, 15:7,</em> 57-
62.</cite>
    <cite class="book">Baker, S. (2006). <em>Validate Your
Pages.... Or Else!.</em> Detroit, MI: Are you out of your mind
publishers.</cite>
    <cite class="article">Lane, J. C. (2007). Dude, HTML 4,
that's like so 2000. <em>The Journal that Publishes Genius, 1:2,
</em> 12-34.</cite>
    <cite class="website">Smith, J. Q. (2005). <em>Web Standards
and You.</em> Retrieved May 3, 2007 from Web standards and
you.</cite>
  </div>

  <div id="footer">
    <p>The content of this page is copyright © 2007 <a
href="mailto:jonathanlane@gmail.com">J. Lane</a></p>

  </div>
</body>
</html>
```

Nem szeretnék most sorról-sorra mindent végigelemezni a forráskódon, mivel ezekről még lesz szó bőven a következő leírásokban, de néhány fontosabb pontot kiemelnék.

Az első sort nevezzük a dokumentum típusdeklarációjának, vagy röviden doctype-nak. Ebben az esetben ez XHTML 1.0 Transitional. A doctype egy szabályhalmazt definiál, amelyet a jelöléseknek követniük kell, és amelyeket így ellenőrizni lehet.

Az 9-11-es sorok töltik be a CSS fájlt a laphoz, az ebben megadott stílusok lesznek alkalmazva a lap különböző elemein. A CSS fájl tartalmát, amelyben a lap formázását adtuk meg, a következő pontban láthatod.

A referencia különböző elemeihez különböző osztályokat definiáltunk. Ezzel lehetővé válik, hogy minden egyes referenciához más stílust adjuk, ebben az esetben például minden referencia jobb oldalához egy-egy színt rendeltünk, megkönnyítve ezzel a lista áttekintését.

Most pedig lássuk a CSS-t, amely a HTML stílusát adja meg.

styles.css

```
body {
  background: #fff url('images/gradbg.jpg') top left repeat-x;
  color: #000;
  margin: 0;
  padding:0;
  border: 0;
  font-family: Verdana, Arial, sans-serif; font-size: 1em;
}

div {
  width: 800px;
  margin: 0 auto;
}
```

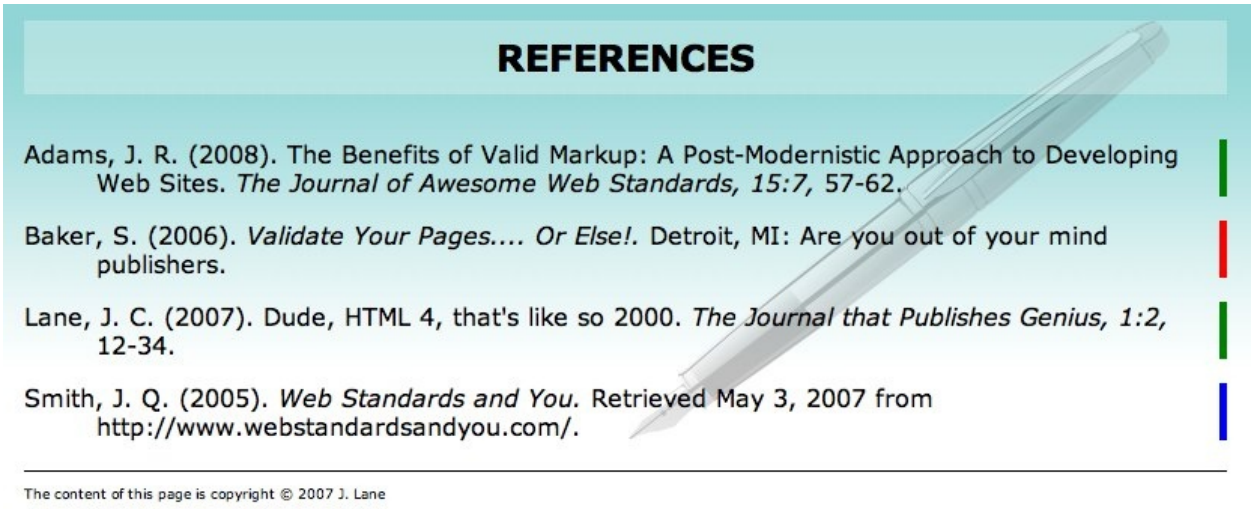
```
#bggraphic {
  background: url('images/pen.png') top left no-repeat;
  height: 278px;
  width: 362px;
  position: absolute;
  left: 50%;
  z-index: -100;
}

h1 {
  text-align: center;
  text-transform: uppercase;
  font-size: 1.5em;
  margin-bottom: 30px;
  background: url('images/headbg.png') top left repeat;
  padding: 10px 0;
}

#references cite {
  margin: 1em 0 0 3em;
  text-indent: -3em;
  display: block;
  font-style: normal;
  padding-right: 3px;
}

.website {
  border-right: 5px solid blue;
}

.book {
  border-right: 5px solid red;
}

.article {
  border-right: 5px solid green;
}

#footer {
  font-size: 0.5em;
  border-top: 1px solid #000;
  margin-top: 20px;
}

#footer a {
  color: #000;
  text-decoration: none;
}

#footer a:hover{
  text-decoration: underline;
}
```

Meglehet, hogy a stílusozással átestem a ló túlsó oldalára, például beállítottam több háttérképet is, de meg akartam mutatni, hogy miket lehet csinálni a CSS segítségével.

Az első sor néhány alapértéket állít be a dokumentumon, mint például a szöveg és a háttér színe, a szöveg körüli keret szélessége, stb. Sokan nem foglalkoznak azzal, hogy ezeket az alapértékeket ilyen módon megadják, és a legtöbb modern böngésző amúgy is definiálja a saját alapértékeit. Mégis jó dolog, ha ezeket megadod, mert nagyobb befolyásod lesz a lap megjelenésére a különböző böngészőkben.

A következő részben a lap szélességét állítjuk be 800 pixelesre (bár ezt megadhattam volna százalékban is, így a lap mérete annak függvényében változna, hogy éppen mekkora a böngésző ablaka). A margin értéke biztosítja, hogy a tartalom középre kerül.

A következőben nézzük meg a lap háttérképeit (ezeket a *background: url* jelölésekkel adtuk meg). Három különböző háttér elemet használtam ezen a lapon. Az első egy gradiens, amely a teljes lapon ismétlődik, egy szép kék átmentet okozva. A második egy félig átlátszó PNG egy tollról, amely kontrasztot képez a fölötte látható szöveggel. (Az ilyen félig átlátszó PNG-k nem működnek az Internet Explorer 6-os és a korábbi verziókban, viszont működnek minden más modern böngészőben; nézd meg Dean Edward IE javító JavaScriptjét egy megoldáshoz IE6-hoz, amely sok más CSS támogatási problémát is javít IE6-ban.) Végül ismét egy félig átlátszó PNG-t használtam a fejléc hátteréhez. Ez nagyobb kontrasztot ad a fejlécnek, és jól is mutat (1.4. ábra).



1.4. ábra: A bemutatott példa a stílusokkal együtt

1.4.6 Tesztkérdések

- Mi a különbség az osztály és az azonosító között?
- Milyen szerepe van az XHTML-nek, a CSS-nek és a JavaScriptnek egy weboldalon?
- Vedd a példában szereplő index.html fájlt, és módosítsd a megjelenését kizárólag CSS segítségével. Ne módosítsd a HTML fájlt.
  - Adj egy ikont a különböző referencia típusokhoz (különböző ikonokat a cikkekhez, könyvekhez és erőforrásokhoz).
  - Tüntesd el a copyright megjegyzést a lap aljáról.
  - Változtasd meg a fejléc megjelenését.
- Miket kell módosítanod a CSS-en ahhoz, hogy a példában szereplő oldal jól jelenjen meg a mobilböngészőkben is?

1.5. Webes szabványok — szép álom, de mi a valóság?

Egészen mostanáig a webes szabványok ideális, szép világáról volt szó. A szabványok segítségével ugyanúgy működnek a weblapok minden böngészőben, minden operációs rendszeren és minden elérhető elektronikus eszközön. De valóban ez a valóság? Valóban minden böngésző 100%-ig támogatja a szabványokat? Vajon minden fejlesztő egyformán jól használja a szabványokat? Szabványosan építik-e fel a webfejlesztők az oldalaikat, és aztán nyugodtan hátradőlnek, mert mindenhol tökéletesen működik az oldaluk?

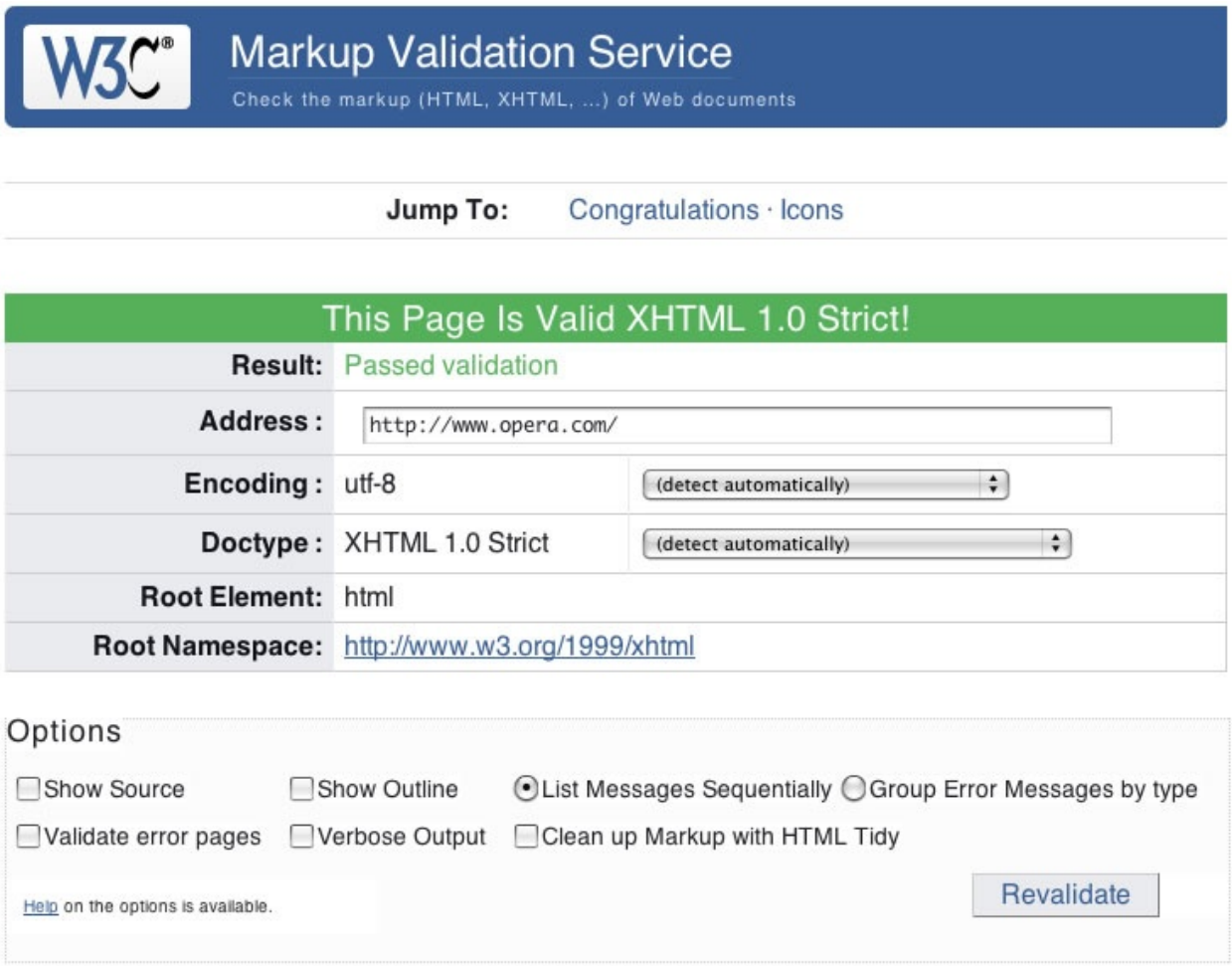
Egy nagyon egyszerű válasz van ezekre a kérdésekre: nem. A fenti kérdések egy ideális helyzetre vonatkoznak, amely távol esik a valóságtól.

1.5.1 Hogyan ellenőrizheted, hogy megfelelsz-e a szabványoknak?

Mielőtt belevágnánk a témába, felteheted magadban a kérdést: „Honnan lehet megtudni, hogy egy weblap használja a szabványokat?” Talán másképp néz ki, mint a többi weboldal?

Igen is meg nem is. A szabványos weboldalak, ha jól készítették el őket, nem néznek ki másképpen, mint az összetákolt, toldozott-foldozott weblapféleségek. Viszont az oldal forráskódja (amit a legtöbb böngészőben úgy nézhetsz meg, hogy jobb gombbal vagy a Ctrl-lal kattintasz egy weblapon, majd kiválasztod a „Forráskód” vagy egy ehhez hasonló menüpontot) már teljesen másképp néz ki. A szabványos weblap szép, tiszta jelölésekből áll, és csak kevés vagy egyáltalán semmilyen formázást nem tartalmaz a weblapra nézve. Ezt talán nem veszed még észre elsőre, de hidd el, a látássérültek, akik képernyő-felolvasókat használnak, vagy a keresőrobotok ezt azonnal észreveszik. A szabványok használatának előnyeiről már beszéltünk az előző leírásokban.

A legegyszerűbb módja a szabványosság ellenőrzésének egy egyszerű eszköz, az ún. validátor használata, amely online elérhető. A W3C (World Wide Web Consortium) ingyen elérhetővé tette a validátort a http://validator.w3.org/ címen (lásd az 1.5. ábrán). Ezt az eszközt használhatod (sőt ajánlott használnod) fejlesztési hibák keresésére az (X)HTML kódodban. A CSS-t egy másik validátorral lehet ellenőrizni, amelyet a http://jigsaw.w3.org/css-validator/ címen találsz. Kattints bátran ezekre a címekre, és ellenőrizd le néhány kedvenc oldalad forrását.



1.5. ábra: A W3C validátora ellenőrzi a weblapokat, és megmutatja a lehetséges hibákat a jelölésekben

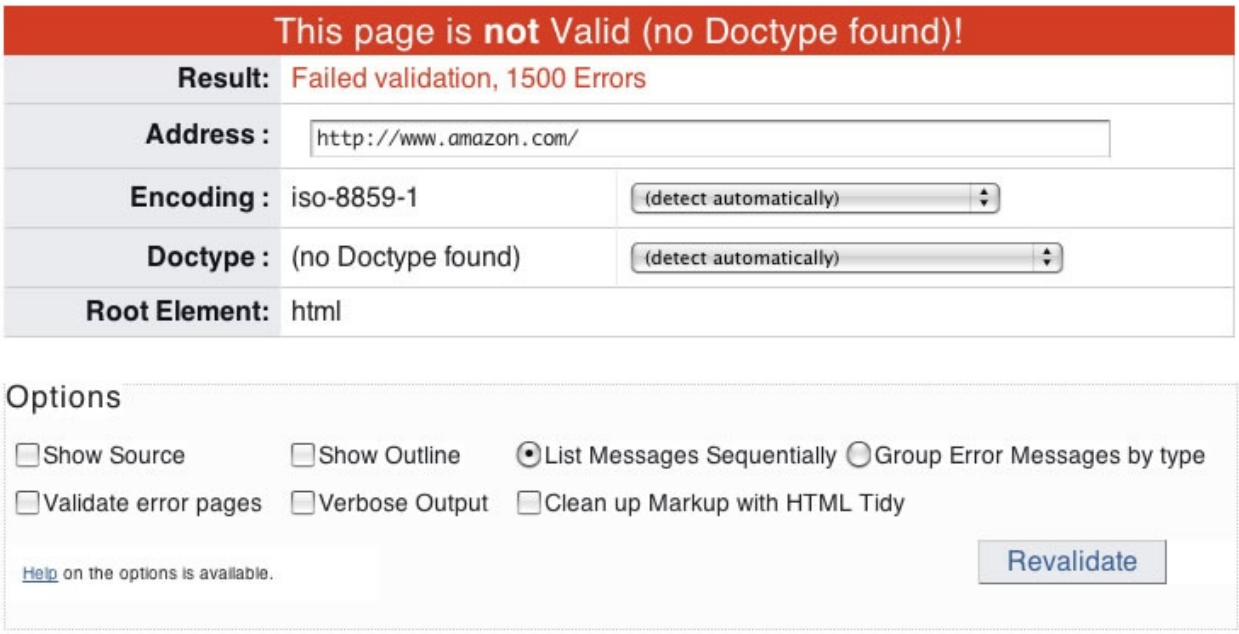
Azzal, hogy meggyőződsz a weblapjaid érvényességéről, még nincs vége a harcnak. Hogyan ellenőrizhetjük, hogy a böngészők jól támogatják-e a szabványokat? A Web Standards Project készített egy tesztsorozatot, amelyeket Acid teszteknek neveztek el. Ezek a tesztek bonyolult HTML és CSS szabályokat tartalmaznak (és még néhány más szabályt is), amellyel ellenőrzik, hogy a böngészők jól jelenítik-e meg a különböző tesztekhez tartozó eseteket. Az Acid teszt utolsó verziója, az Acid3 még most is fejlesztés alatt áll. Az Acid tesztekről többet is olvashatsz az http://www.acidtests.org/ oldalon, ahol kipróbálhatod a tesztet a böngésződben is.

1.5.2 Szabványtámogatás a jelenlegi weblapokon

Vajon a nagyobb oldalak használják-e a webes szabványokat, vagy csak össze vannak dobálva? Nézzünk meg néhány nagyobb vállalatot a weben, hogy lássuk, hogyan értékeli őket a W3C validátora. Meg fogsz lepődni, hogy hány nagy oldal nem megy át a teszten; ettől azonban nem kell elcsüggedned, te még mindig megírhatod úgy az oldaladat, hogy az megfeleljen a szabványoknak. Az alábbi példákat olvasva ne felejtsd el azt sem, hogy minél nagyobb és bonyolultabb egy website, annál nehezebb benne megoldani, hogy átmenjen a validáláson (valamint figyelembe kell venni más szempontokat is, például a felhasznált technológiákat).

Amazon: Vásárlás szabványosan?

Valószínű, hogy ha vásároltál már valamit online, akkor belefutottál az amazon.com oldalba (vagy valamelyik regionális változatába). Az Amazon a kibertér nagykereskedése, ahol mindent megtalálsz a könyvektől a CD-ken át az élelmiszerekig. De vajon átmegy-e az Amazon.com a validáláson?

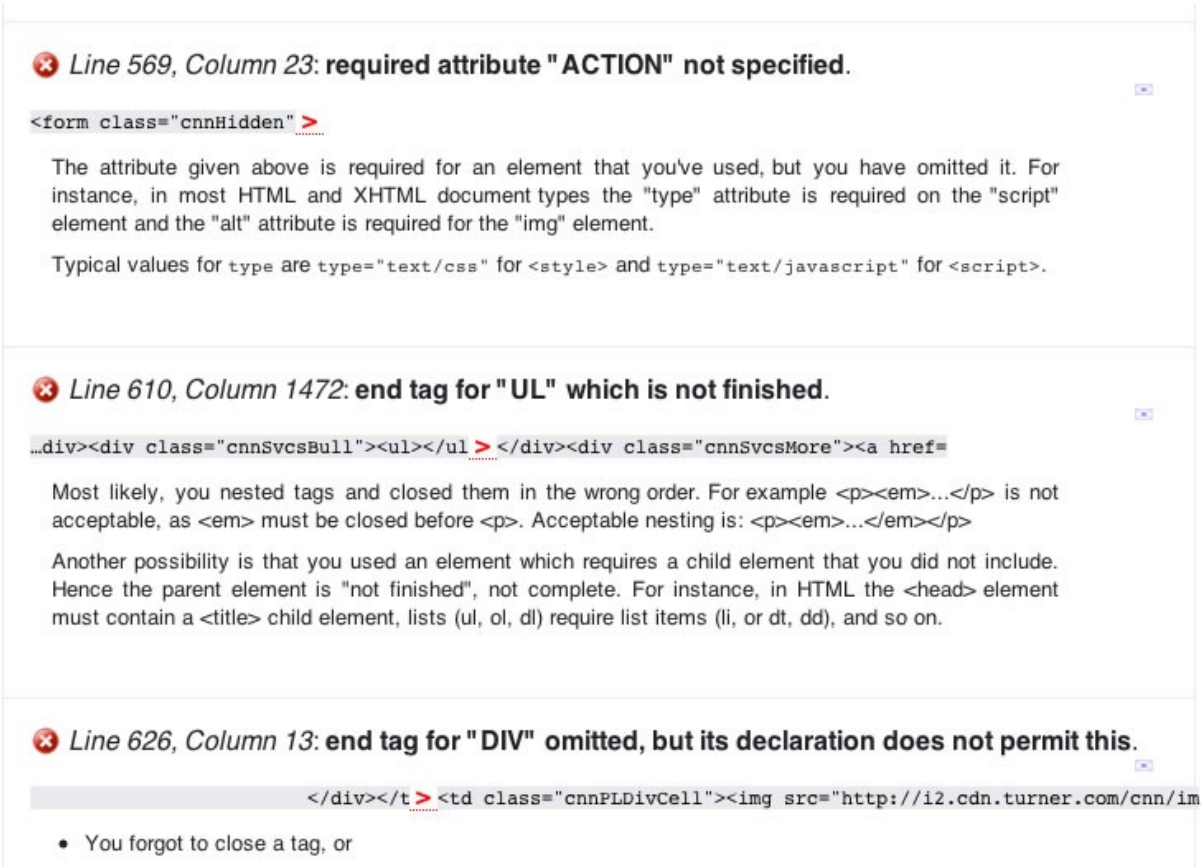


1.6. ábra: Az Amazon sajnos megbukott! Nem csak jelölési hibákat tartalmaz, hanem még a doctype-ot sem adták meg (amely megmondhatná, hogy milyen HTML/XHTML verziót használnak).

Az Amazonnak azonban van mentsége a szabványtámogatás hiányára. Nincs nálam az Amazon fejlesztési naplója, de ha tippelnem kéne, azt mondanám, hogy az Amazon már egy jó ideje köztünk van, és valószínűleg még mindig ugyanazt a szoftvert használják a website futtatására, mint amit régebben. Mivel a webes szabványok csak az ezredforduló után kerültek képbe, valószínűsíthető, hogy az Amazon rendszerét még abban az időben fejlesztették, amikor a webes szabványok csak homályos elképzelések voltak néhány fejlesztő fejében. Azt gyanítom, hogy az Amazon is eleget szenved a régi rendszerük miatt, amit tovább kell foltozzanak minden fejlesztésnél.

CNN: Szabványos hírek?

A hírszolgáltatók vajon valóban rendszerezettek? A CNN.com egyike a legnagyobb médiaoldalaknak a weben. Kiterjedt nemzetközi kapcsolatokkal rendelkeznek, a híreket valós időben kell megjelenítsék, úgyhogy biztosan van egy képzett, belső webfejlesztő csapatuk, amely képes szabványos oldalakat gyártani számukra. Vagy mégsem?



1.7. ábra: A CNN.com (2008. április 15-én) 33 hibával megbukott. HTML 4.01 Transitional doctype-ot definiálnak, de sok helyen mégis XHTML jelölések vannak.

33 hiba egyáltalán nem rossz egy olyan méretű és komplexitású lapon, mint amilyen a CNN oldala. Ez a 33 hiba eredhet onnan (és itt megint csak találgatok), hogy a tartalomkezelő rendszerük nem szolgáltat teljes mértékben szabványos kódot, de a jelölési hibák származhatnak a hírek készítőitől is, akik az íráshoz ugyan értenek, de a webfejlesztéshez már nem; mindkét eshetőség esetén még elfogadhatóak a hibák.

Apple: az elegancia és a design csúcsa … de a validálásé is?

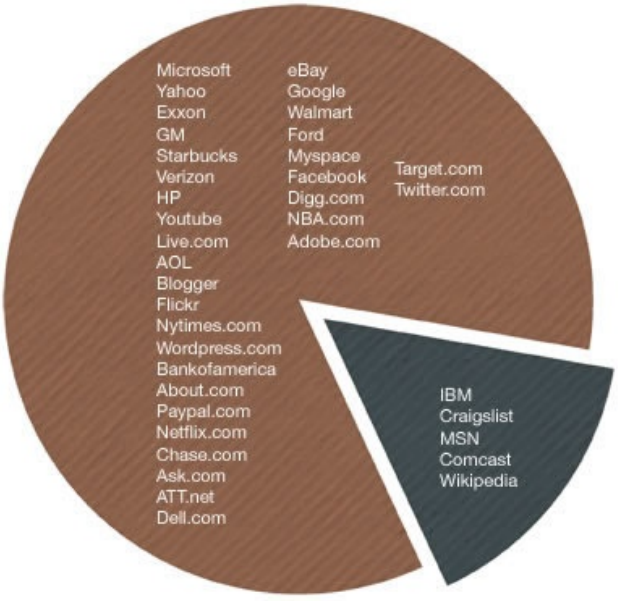
Az Apple széles körben ismert a mutatós és jól használható szoftvereiről és hardvereiről. A termékbejelentéseik sokszor egy isteni kinyilatkoztatással érnek fel a hűséges rajongóhad számára. Az Apple oldalát (lásd az 1.8. ábrát) különösen szép design és rendezettség jellemzi, de vajon átmegy a validáláson is?

1.8. ábra: Az Apple.com már nagyon közel van az érvényes HTML 4.01 Transitional jelöléshez. Az elenyésző 6 hiba az oldalon csak elírásokat és Safari-specifikus tageket takar.

Az Apple weboldala már nagyon közel áll a teljes validáláshoz. Valójában úgy 5 perc alatt ki lehetne javítani ezt a pár hibát, és akkor teljesen tiszta lenne az oldaluk. Egy hibát azért szeretnék kiemelni, mivel az Apple úgy döntött, hogy egy Safari-specifikus attribútumot használ a keresőmezőhöz (a type="search" attribútumot fűzték hozzá). Safariban ez lehetővé teszi, hogy egy kis nagyító ikonra kattintva megnézhessük a korábbi kereséseket. Viszont más böngészőkben, például Operában vagy Internet Explorerben csak egy egyszerű szövegmező jelenik meg.

Egy kis szabványtámogatási vizsgálat

Ahelyett, hogy a fenti példákhoz hasonlóan még megvizsgálnánk néhány tucat weblapot, inkább gyorsítunk egy kicsit. A fennmaradó oldalakból készítettem egy kördiagramot, amelyen 40 vállalati weboldalt soroltam fel, amelyeket a Fortune 500 listáról, valamint az Alexa listája alapján a legforgalmasabb oldalak közül válogattam ki. Az 5. ábrán megnézheted, hogy mit találtam:



1.9. ábra: Az oldalak 85%-a valamilyen módon megbukott a validáláson. Egyes oldalakon több ezer hiba van, míg másokon csak néhány elírás itt-ott.

1.5.3 Miért hiányoznak a szabványos oldalak?

Sírni támad kedvünk: „miért, miért nem képesek validálni?” Talán ez egy kissé drámai, de gondolom valami hasonló futott át neked is az agyadon. Miért csak ilyen kevés oldal szabványos? Már beszéltem korábban néhány lehetséges okról, mint például a régi kereskedelmi rendszerek vagy a tartalomkezelő rendszerek, de vannak még a háttérben más okok is.

Oktatás

Megnéztem néhány oktatási intézményt, ahol vezetői információs rendszer (MIS) tanfolyamot, számítástechnikai és New Media tanfolyamot is tartottak, amelyek közül mindegyik foglalkozott weboldalak készítésével. Bár sok hasznos dolgot lehet ezeken a tanfolyamokon tanulni, azt nem igazán említették egyiken sem, hogy hogyan kell voltaképpen egy weblap kódját megírni. Miután megnéztem több egyetemi kurzus programját is, az volt az általános benyomásom, hogy az olyan webes nyelvek, mint a HTML, a CSS és a JavaScript még nem érik el a számítástechnikai programozás szintjét, viszont már túllépik a MIS vagy New Media jellegű tanfolyamok kereteit.

Mindezzel oda szeretnék kilyukadni, hogy a legtöbb oktatási tananyag nem tartalmaz részleteket ezekkel a témákkal kapcsolatosan. Fogadni mernék, hogy ha megkérdeznénk 10 webfejlesztőt, akik webes szabványokkal dolgoznak, hogy hol tanulták meg a szabványok használatát, legalább 9 azt válaszolná, hogy saját erőből (a maradék 1 fejlesztő meg nem válaszolna semmit, mert éppen azzal lenne elfoglalva, hogy az oldala rendesen működjön IE6-tal is).

A World Wide Web Consortium (W3C), aki a szabványok fejlesztéséért felelős, valamint a Web Standars Project (WaSP) még nem adta fel a küzdelmet, és megpróbálják mind a fejlesztőket, mint a böngészők gyártóit rávenni a jobb szabványtámogatásra.

Ez az egész kurzus éppen azért készült, hogy egy megfelelő technikai szintű tananyagot nyújtson a webes szabványok megismeréséhez és tanításához, és éppen azért ingyenes, hogy bárki fel tudja használni tanulásra. Szeretnénk megszüntetni néhány olyan indokot (bár talán mondhatnánk kifogást is), hogy miért nem használják az emberek a szabványokat. A várható előnyök ismeretében már igazán nem marad semmilyen kifogás a szabványok használatával szemben.

Üzleti okok

Az egyik kedvenc weboldalamon, amely különböző vállalkozások web alapú startupjaival foglalkozik, többször is felmerült az a téma, hogy érdemes-e a webes szabványokat az ún. „Web 2.0 alkalmazásokban" felhasználni. Általában két táborra szakad a társaság, ahol az egyik oldal azt állítja, hogy van értelme (főleg az itt is felsorolt okok miatt), míg a másik tábor egyszerűen csak annyit mond, hogy „kit érdekel".

Való igaz, hogy a böngészők sokszor megbirkóznak az igazán rossz kódokkal is. A weblapod nem kell átmenjen a validáláson ahhoz, hogy minden nagyobb böngészőben helyesen jelenjen meg. Ezért aztán üzleti szempontból, ahol az idő egyenlő pénz, miért kellene validálásra pazarolni az energiát egyáltalán? Ha össze tudsz dobálni egy táblázat alapú weboldalt félóra alatt, vagy megírod az oldalt félóra alatt HTML-ben és CSS-ben, majd újabb félórát fordítasz a validálásra és arra, hogy minden böngészőben egyformán működjön, ráadásul a végeredmény minden nagyobb böngészőben egyforma lesz, akkor szerinted melyik lesz az egyszerűbb út?

Sokan az én generációmból (úgy 30 fölött) még azt tanulták a webfejlesztésről, hogy az elrendezést táblázatokkal, a formázást pedig a font taggel lehet megoldani. Ijesztő lehet újra megtanulni valamit, amikor a jelenlegi módszer továbbra is működik (még mindig jól mutat a legtöbb webböngészőben). A munkáltatók nem ismerik a különbséget; még soha nem hallottam, hogy egy manager a jelölés minőségéről beszélt volna egy performance review alatt. Szóval, hol van a motiváció?

Most már biztosan te is sejted, hogy én melyik oldalt pártolom, úgyhogy elárulhatom, hogy a régi, összedobált kód írása rövidlátásra utal. Tapasztalataim alapján egy szabványos weblap újratervezése sokkal, de sokkal egyszerűbb, mint egy összetákolt oldalé (mindkettőhöz volt szerencsém). Sajnos az az utópia, hogy az újratervezéskor egyáltalán nem kell hozzányúlni az XHTML részhez, az esetek többségében nem igaz, de azért nagyon közel áll hozzá. Ne felejtsd el azt sem, hogy manapság már sokkal több olyan állás-hirdetést látni, amelyben elvárják a webes szabványok ismeretét, mint régebben.

1.5.4 Tesztkérdések

- Megnéztünk néhány „nagy" weboldalt, hogy átmennek-e a validáláson. Futtasd le a validálást azokon az oldalakon is, amelyeket gyakran látogatsz. Átmennek rajta? Ha nem, nézd meg, hogy milyen hibákon buknak meg.
- Mi az a doctype? Mire jó?
- Milyen előnyei vannak a webes szabványok használatának az üzleti életben?

2. Webdesign fogalmak

2.1. Információs Architektúra — egy website tervezése

Hagyományosan egy website tervezése (vagy bármilyen más projekt) elég kimerítő lehet. Mindenkinek van valamilyen véleménye, hogy egy website-ot hogyan kell felépíteni, és ezek a vélemények gyakran ütköznek egymással. Az első számú célod egy website felépítésénél az kellene legyen, hogy az eredmény legyen hasznos a felhasználók számára. Nem számít, hogy mit mond a főnököd, hogy mit mond a srác az alsó szintről a doktori diplomájával, és az sem számít, hogy te mit gondolsz. A végén úgyis csak annak a csoport embernek a véleménye számít majd, akik számára a weblapot valójában készíted.

Ebben a cikkben egy website tervezésének a korai fázisait nézzük át, amelyre sokszor Információs Architektúra, vagy röviden IA névvel hivatkoznak. Ennek keretében át kell gondolni, hogy ki lesz az új website célközönsége, milyen információk és szolgáltatások kellenek a website-ra, és mindezt hogyan rendezd el, hogy elérhetővé tehesd a látogatóknak. Át kell nézd az összes információhalmazt, amelyet az oldalra szánsz, és el kell döntened, hogy hogyan darabolhatod szét ezeket, valamint hogy a darabok hogyan viszonyulnak majd egymáshoz.

2.1.1 Tervezd meg előre a készülő weboldalt

Lehetséges, hogy találkozni fogsz olyan webes projektekkel, amelyeknél azonnal fejest ugorhatsz a kódolásba, de ezek leginkább csak kivételek. A továbbiakban megnézünk egy képzeletbeli bandát, a „Dung Beatles”-t, és megpróbálunk segíteni nekik a készülő website-juk tervezésében. Beszélünk a banda tagjaival, és megtudjuk tőlünk, hogy milyen céljaik vannak a weblappal, mit szeretnének rajta látni. Ezután nekilátunk, és kidolgozunk egy struktúrát az együttestől kapott információk alapján.

2.1.2 A „Dung Beatles” együttes

A Dung Beatles együttesnek van egy kis gondja. Ők a legismertebb Beatles emlékzenekar a kanadai Saskatchewan tartományban, de fel kell építeniük egy profilt a közelgő északamerikai turnéjukra ezen a nyáron. Már lefoglaltak több helyet Kanadában és az Egyesült Államokban is, de a szülővárosukon kívül még nem igazán ismertek. Bárcsak lenne valami egyszerű technikai megoldás, amellyel a lehető legtöbb Beatles rajongót elérhetnék viszonylag kevés pénzért.

A Dung Beatles szerencséjére van egy olyan dolog, hogy világháló, így gyorsan el is döntik, hogy a problémájukra a legegyszerűbb megoldás jelenleg egy website készítése lenne. Szükségük van egy helyre, ahol reklámozhatják a turné időpontjait, rajongói csapatokat építhetnek fel különböző városokban, és növelhetik a banda hírnevét. Te most átnézed velük együtt az eddigi ötleteiket, és megpróbálsz egy vázlatot készíteni a weblapjukhoz az eddigi információk alapján.

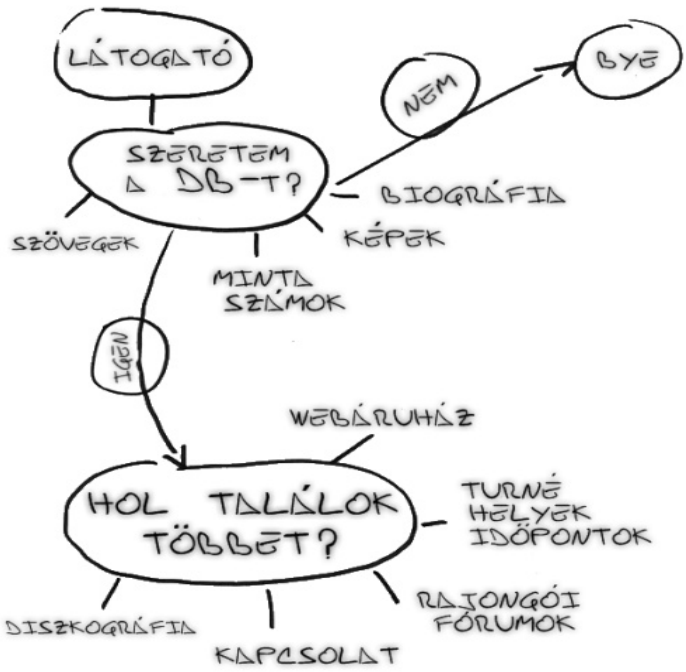
Összehozol a bandával egy találkozót, amelyen megbeszélhetitek a részleteket az igényeikről, valamint megegyezhettek a határidőkben és a költségekben. A beszélgetést azzal kezded, hogy azt javasolod nekik, beszéljenek egy keveset a céljaikról és a terveikről a website-tal kapcsolatban, hogy világos legyen, mit szeretnének vele kezdeni. Milyen reményeket fűznek valójában az online jelenlétükhöz?

A banda ekkor elkezd beszélni a közelgő turnéjukról, hogy szeretnék eljuttatni a turné hírét a Beatles rajongóknak a tervezett helyszíneken. Most február van, és ők öt hónap múlva tervezik elindítani a turnéjukat.

Álljunk itt meg egy pillanatra! Egy website önmagában nem fogja felépíteni a forgalmat, és nem fogja önmagát reklámozni. A beszélgetésből azt a következtetést vonod le, hogy a website egyik fő célja a rajongók online összefogása lenne, egy olyan hely biztosítása, ahol nyomon követhetik a legújabb híreket, újdonságokat, valamint a turnék és találkozók időpontjait és helyeit. A rajongókon keresztül (szóbeszéd útján), és valamilyen reklámfelület által az új embereket ide lehetne irányítani, ahol letölthetnének néhány minta számot, láthatnának képeket az együttesről, és megtudhatják, hogy hol és mikor nézhetik meg őket élőben.

Raul McCoffee, az együttes frontembere rámutat arra, hogy jó lenne, ha egy kis extra bevételre tehetnének szert a turné során néhány CD vagy egyéb cuccok eladásával. Ezután összehozod a bandát, és rajzolsz nekik egy gyors vázlatot arról, hogy a lehetséges látogatók mit akarhatnak, amikor majd meglátogatják a weblapot. Ez még csak az ötletek durva összefoglalása, a struktúra egyelőre nincs igazán megtervezve.

Két nagy csoportra oszthatod azokat az embereket, akik meglátogatják a website-ot: azokra, akik már ismerik és szeretik az együttest (a rajongók), és a többiekre, akik még bizonytalanok. A két csoportot másképpen kell kiszolgálnod: a lehetséges rajongóknak „el kell adnod” az együttest, míg a régi rajongóknak továbbra is fenn kell tartani az érdeklődését. Milyen információk érdekelhetik ezt a két csoportot? A 2.1. ábrán ezt rajzoltuk le: ez tipikusan egy olyan vázlat, amelyet a készülő website jelenlegi stádiumában készíthetsz. Később ennek alapján tervezheted meg, hogy milyen lapokra lesz szükség a website-on, és ezek hogyan kapcsolódnak majd egymáshoz.



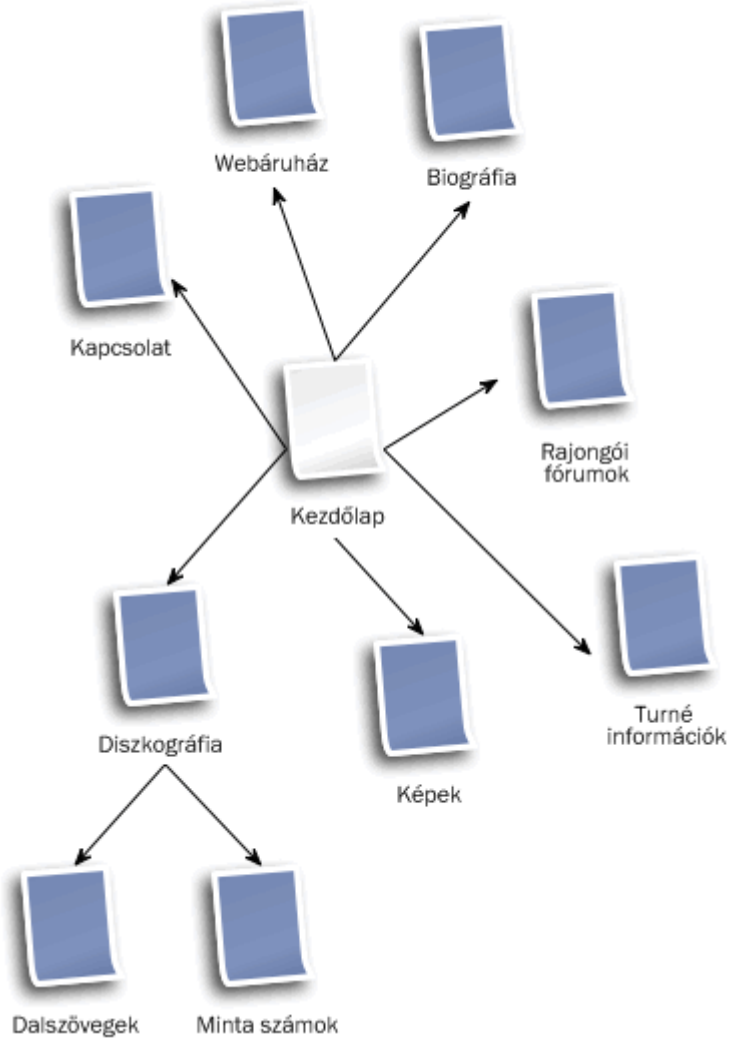
2.1. ábra: Vázlat arról, hogy mire lehet kíváncsi egy látogató

Ezután megegyeztek a költségekben, és hogy a website elindul egy hónapon belül. Megígéred, hogy néhány nap múlva újra felkeresed a bandát, és megmutatod, hogy milyen irányban indultál el.

2.1.3 És most? Rajzoljunk oldaltérképet

Sokan a következő lépésben összedobnak egy egyszerű oldaltérképet, ez úgy néz ki, mint egy organigram (szervezeti diagram). Ezen a diagramon általában csak egyszerű képek vannak, amelyek az egyes oldalak nevét jelölik, valamint hogy hogyan illeszkednek be a website szerkezetébe. Én személy szerint egy kicsivel több részletet adok meg rajta, és röviden beírom azt is, hogy mire szolgálnak az egyes lapok. Például az egyik lap neve „Kezdőlap”, de mi ez a kezdőlap valójában? Vajon csak egy giccses „Üdvözlünk a weblapunkon” szöveg van rajta (borzalom!), vagy egy dinamikus oldal a legújabb hírekkel és mutatós képekkel? Gondolkodj el néhány percig azon, hogy a fenti vázlat alapján milyen oldalaid lehetnek, és ezek nagyjából mit fognak tartalmazni. Próbáld meg elkészíteni most a saját oldaltérképedet, mielőtt tovább olvasod ezt a leírást.

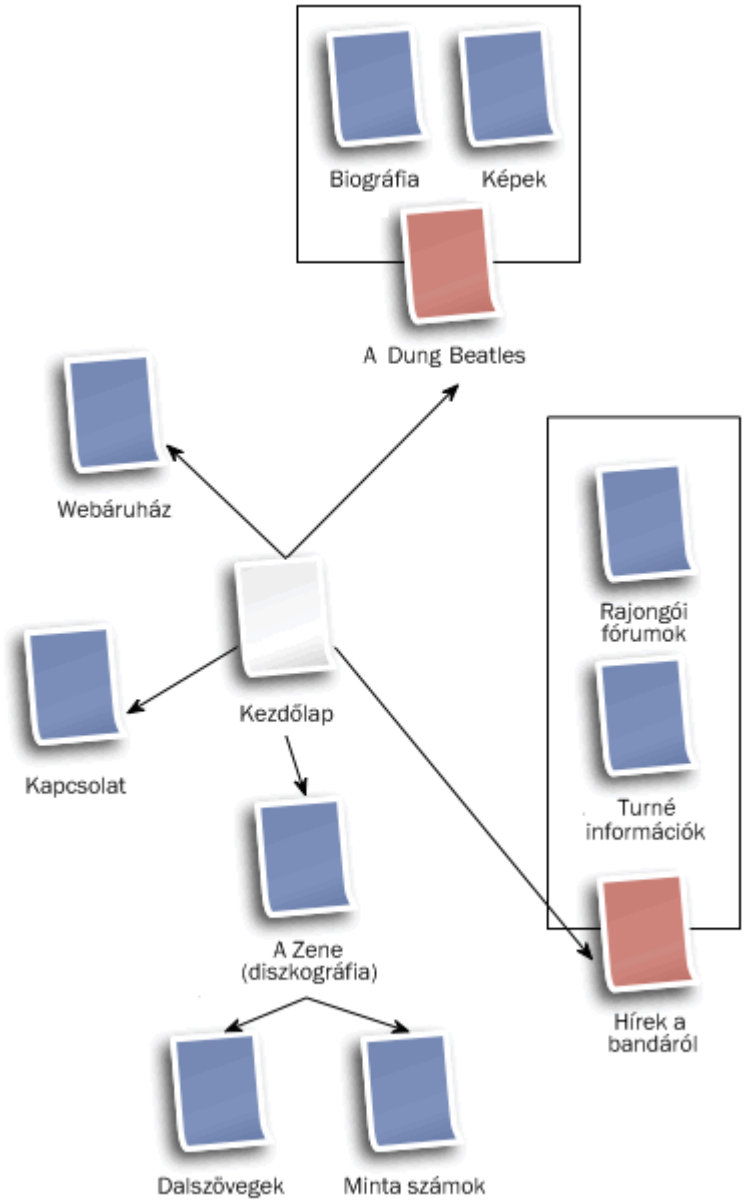
És most kezdjük az alapokkal: készítsünk el egy ilyen oldaltérképet. A 2.2. ábrán az én próbálkozásomat láthatod az ötletelések után. Ilyen lett végül az én oldaltérképem:



2.2. ábra: Az oldal felépítésének első tervezete

Ezen a képen az első próbálkozást láthatod a készülő website struktúrájának ábrázolására. A különböző oldalak egy pókhálószerű diagramon kapcsolódnak egymáshoz, a „Kezdőlap” került középre. Ehhez kapcsolódnak a „Webáruház”, a „Biográfia”, a „Rajongói fórumok”, a „Turné információk”, a „Képek”, a „Diszkográfia” és a „Kapcsolat” oldalak. A „Diszkográfia” alá került még két aloldal, a „Dalszövegek” és a „Minta számok”.

Ebben most már benne van az összes oldal, amire szükségünk lesz, de ezek még nincsenek igazán csoportosítva, csak egyszerűen össze vannak dobálva az oldalak. Ezért a következő lépésben megpróbálom az összetartozó oldalakat valamivel jobban csoportosítani. A 2.3. ábrán láthatod, hogy mire jutottam:



2.3. ábra: Az oldal felépítésének második tervezete

Ezen a képen a készülő website javított struktúráját láthatod. A „Kezdőlap” most a következő oldalakhoz kapcsolódik: „Kapcsolat”, „Webáruház”, „A Dung Beatles” (ami kapcsolódik a „Biográfia” és a „Képek” oldalakhoz), „Hírek a bandáról” (ami kapcsolódik a „Turné infomációk” és a „Rajongói fórumok” oldalakhoz), és „A Zene” (ami kapcsolódik a „Dalszövegek” és a „Minta számok” oldalakhoz).

Több dolgot is módosítottam a második tervezetben. Az új „Hírek a bandáról” oldal lehetőséget ad az együttes számára, hogy bármilyen információt megoszthassanak a rajongóikkal. Ha már a nyári turné végetér, és a „Turné információk” oldal elavulttá válik, akkor is írhatnak még ide új dolgokat. Egy egyszerű blog formátum használatával a rajongók itt kommentezhetnek is a különböző posztoknál, és ez segít egy online közösség felépítésében a Dung Beatles számára is. A hírek és a turné információk valószínűleg

összemosódnak, így ezeket egy csoportba tettem. Másrészt a „Hírek” szó a legtöbb ember számára sokkal egyértelműbb, és hamarabb felfedezik az oldalon, amikor információt keresnek.

Az új „A Dung Beatles” oldal összefogja a banda tagjainak életrajzát és a róluk készült fényképeket. Ezen az úton jó lehetőség adódik a tagok életrajzának elhelyezésére. Az együttes neve itt utal arra, hogy ezen a lapon további információkat találhat az érdeklődő a bandáról.

Végül a „Diszkográfia” egy eléggé technikai jellegű kifejezés. Valószínűleg kevesebben értenék meg ezt, mint „A Zene” kifejezést. Ezen az oldalon később további információkat is el lehet helyezni, például az inspirációk forrása, az egyes számok készítésének története, stb... érted az ötletet. Azt hiszem, az oldaltérképpel így végeztünk is. A következőkben még beszélünk egy keveset a lapok elnevezéséről, és aztán tovább részletezzük az egyes oldalak tartalmát.

2.1.4 Nevezd el az oldalakat

A lapok neveinek kiválasztása a legnehezebb döntéseid között lehet egy website tervezése alatt. Ez nem csak a látogatók szemszögéből fontos, hogy könnyen megtalálhassák a website-on az őket érdeklő információkat, hanem ez befolyásolja azt is, hogy milyen könnyen találják meg az oldalad a különböző keresők (a kurzus alatt többféle keresőoptimalizációs módszerrel is találkozni fogsz).

A keresőmotorok alapjában véve a weblap szövege, az adott lap URL-je és az oldalra mutató hivatkozások szövege alapján döntik el, hogy egy oldal „mennyire fontos”. Ha az oldalaidnak egy ésszerű nevet és egy értelmes URL-t adsz, akkor az arra készteti az embereket, hogy az oldalra mutató hivatkozásoknak is egy értelmes nevet adjanak.

Itt egy példa. Tegyük fel, hogy van egy autós céged, és van egy „Speedster” nevű autómodelletek. Készítettek egy weboldalt, amelyben az autóitokat reklámozzátok, és az egyik oldalon a felsoroljátok az autó jellemzőit. Hogyan fogod elnevezni ezt a lapot? „Jellemzők”, „Legfontosabb tulajdonságok”, „A Speedster jellemzői” vagy „Csengők és sípok”? Én a „A Speedster jellemzői” változatot javasolnám, mivel pontosan lefedi a lap tartalmát, ez jelenik majd meg az ablak fejlécében, és feltűnő (vagyis könnyen indexelhető) lesz a keresőrobotok számára is. Ráadásul felhasználhatod az URL-ben is (például www.autosceg.hu/speedster/speedster-jellemzok/).

2.1.5 Ne felejtsd el a részleteket

Ezen a ponton nem kell még mindent kitalálnod, de legalább egy rövid leírást érdemes készítened mindegyik oldalhoz, amelyben leírhatod, hogy mire kell majd figyelni az oldal készítésekor. Ha megvan a site struktúrája, számozd meg a lapokat, és írj egy rövid leírást mindegyikhez, hasonlóan ahhoz, mint amit a 4. ábrán láthatsz (a cikk végén a tesztkérdések között az lesz az egyik feladat, hogy készítsd el a többi oldal leírását).

2.4. ábra: A Kezdőlap részletesebb leírása

Ezen a ponton még bőven elég ennyi információ a lapokról. Nem szükséges, hogy részletesen leírd a lap funkcionalitását, a technológiát, amellyel elkészíted, vagy a megjelenés részleteit. Csak azt írd le néhány egyszerű szóval, ami az oldallal kapcsolatban most a fejedben van. A cél itt az, hogy az elképzeléseidet meg tudd mutatni a kliensnek, és hogy biztos lehess benne, hogy mindent átgondoltál az oldalakkal kapcsolatban.

Nem ritka az sem, hogy ezen a ponton észreveszed, hogy már túl sok oldalt terveztél a site-ra, és nem tudsz mindegyikhez tartalmat rendelni. Meg lehet őszülni a lapok hierarchiájának készítése közben. Például ha a banda minden tagja szeretne írni magáról néhány mondatot, akkor nem kell minden taghoz külön lapot csinálni, elég, ha egyetlen lapra felteszed mindet.

2.1.6 Tesztkérdések

- Nézd meg újra az 1. ábrát, és készíts el egy hasonló vázlatot egy autóról szóló website-hoz (válassz bármilyen jelenlegi vagy képzeletbeli autót).
  - Mit szeretnének megtudni a látogatók?
  - Van bármi olyasmi a létező autós website-okon, amit elengedhetetlennek tartasz? Vagy amit komolytalannak?
- Vedd elő az ötleteidet, és rendezd el az információkat. Milyen csoportosításnak van értelme?
- Egy másik hasznos foglalkozás egy website tervezésekor az, hogy megnézed a konkurenciát. Keress rá egy együttes honlapjára (bónuszhoz keress egy elismert zenekart), és nézd meg, hogy mit nyújt a weblapjuk. Van rajta olyasmi, amit mi kifefelejtettünk?
- Nézd meg a 4. ábrát, és készíts hasonló leírásokat a többi laphoz, amelyeket a website-hoz terveztünk.

2.2. Mi kell egy jó weblaphoz?

Részletesebben megnézzük, hogy milyen tartalom kerüljön a Dung Beatles weboldalára, közben megtudhatod azt is, hogy mi szükséges egy jó website-hoz.

Még most sem fogunk foglalkozni a kódolással, csak megvizsgáljuk a különböző lapokat, és közben megpróbáljuk kitalálni, hogy mi jelenjen meg rajta, figyelembe véve a következetességet, a használhatóságot és a hozzáférhetőséget.

2.2.1 A kezdőlap

Ezen a ponton sokan biztosan erre gondolnak: „kezdjük azzal a lappal, amellyel a legtöbb felhasználó az elején találkozik, a kezdőlappal. Végül is ez a logikus, nem?”

Bár elég logikusan hangzik, de nem biztos, hogy ezzel a lappal a legjobb kezdeni. Gyakori hiba, hogy a figyelem túlságosan is a kezdőlapra terelődik az elején. Az oldalak kezdőlapja sokszor túl vegyes, mindent megpróbál az oldalról kiemelni, és mindent meg akar mutatni mindenkinek. Ennek az eredménye viszont általában csak egy rakás érdektelen szemét.

Hogy megértsd, mire gondolok, pillants rá az MSN kezdőlapjára (lásd még a 2.5. ábrán). Elképesztő ez a hatalmas tartalom és hivatkozás dömping. Az MSN websitehálózat óriási: az utazástól a tévéig, a randikeresőtől az útvonaltervekig, az elektromos kütyüktől a környezetvédelemig minden tartalom megpróbálja felhívni magára a figyelmedet.

2.5. ábra: Az MSN kezdőlapja - linkek garmadájával

Ez a fajta „dobjunk bele mindent a mosogatórongyon kívül, aztán dobjuk bele a rongyot is” hozzáállás akár még jó is lehet egy link gyűjtemény esetében, de a mi együttesünknél egy ilyen kezdőoldal valószínűleg sokkal több látogatót riasztana el, mint ahányat lenyűgözne.

Egy másik gyakori tévhit, hogy a kezdőlap az az oldal, amelyikkel a látogatóid először találkoznak. Talán ha csak hallanak a bandáról, vagy kapnak valamilyen tollat, matricát, kitűzőt, akármit, amin rajta van a webcímük, akkor előfordulhat, hogy csak egyszerűen beírják a címet a böngészőbe, és a kezdőlapon kötnek ki.

Sokkal valószínűbb azonban, hogy a látogatók egy keresés alapján találják meg az oldalad. Ha a banda nevére keresnek rá, akkor lehetséges (de nem garantálható), hogy az oldal kezdőlapja kerül hozzájuk első találatként. Ha például valaki arra keres majd rá, hogy „Beatles emlékkoncert”, akkor előfordulhat, hogy egyből a „Turné információk” oldalra jut, mivel ez lesz az első találata. Vagy ha arra keresnek rá, hogy „Saskatchewan együttes”, akkor az első találat az oldalról valószínűleg a bandáról szóló oldal lesz, mivel azon megemlítik azt is, hogy hol élnek, míg a kezdőlapon ez nincs rajta.

A New York Times egyik cikkében[5], amelyben azt elemzik, hogy miért nem kérnek pénzt többé a régi cikkekhez való hozzáférésért, megjegyzik, hogy a látogatók viselkedése megváltozott:

... egyre több olvasó érkezett a keresőkön, vagy más oldalakon közzétett hivatkozásokon keresztül, és egyre kevesebben egyenesen a NYTimes.com kezdőlapjáról. Ezek az indirekt olvasók nem fértek hozzá a cikkek tartalmához a fizetős szint miatt, és a közvetlenül érkezett felhasználókkal szemben ők csak nagyon ritkán fizettek elő a zárt tartalomra. Az ingyenes hozzáféréssel megnőtt a régi cikkek forgalma, és ezáltal emelkedtek a reklámbevételek.

Mit jelent mindez a mi oldalainkra nézve?

Mindez azt jelenti számunkra, hogy bár szükség van arra, hogy a tartalmat több különböző oldalra darabold szét, nem szabad elfelejtened azt sem, hogy a látogatók hogyan érkeznek majd meg hozzád, mi lehet az, amit éppen keresnek, és vajon hová szeretnének továbblépni, ha fel szeretnék fedezni az oldaladat.

Bár eléggé csábító, hogy azért mindennek szorítsunk egy kis helyet a kezdőlapon, valójában sokkal jobb, ha itt csak kiemeljük a többi tartalmat az oldalról, és a forgalmat egyenesen oda irányítjuk. Kezeld úgy a kezdőlapot, mint a többi oldalt a website-on, és adj neki egy határozott célt (például itt megjelenítheted az újdonságokat, adhatsz egy áttekintést az oldalról, bemutathatod a bandát és továbbküldheted a látogatót az aloldalakra, stb.). A lapnak szüksége van még valamilyen navigációra a többi oldal irányába, valamint egy márkajelzésre.

Most egy kicsit mélyebbre ásunk, hogy jobban megismerhessük ezeket a dolgokat...

2.2.2 Navigáció

A webfejlesztés során a navigáció a különböző oldalak között egy fontos, sőt talán az egyik legfontosabb témakör. Meg kell találnod a leggyakoribb célpontokat a weblapodon, és ezeket teheted bele az oldalad fő navigációjába.

Van egy elég gyakori téves elképzelés a website-ok navigációjával kapcsolatban, amelyről már talán te is hallottál, mégpedig az, hogy minden oldal legfeljebb három kattintásnyi távolságra lehet egymástól. Ennek az elméletnek a terjesztése a felelős a legrosszabb és legbonyolultabb navigációkért az interneten. Nézd csak meg példának a népszerűbb online piactér- és árösszehasonlító oldalakat: a legtöbbjük megpróbál annyi linket bezsúfolni a navigációba, amennyit csak tud, mantraként ismételgetve, hogy a felhasználók a lehető legkevesebb kattintással tudjanak valamit vásárolni, máskülönben lelépnek és átpártolnak a konkurenciához. Ez valójában oda vezet, hogy a felhasználót túl sok információ éri, és nem lesz képes ezeket hatékonyan használni. A túl sok választási lehetőség éppen olyan bénító lehet, mint a túl kevés.

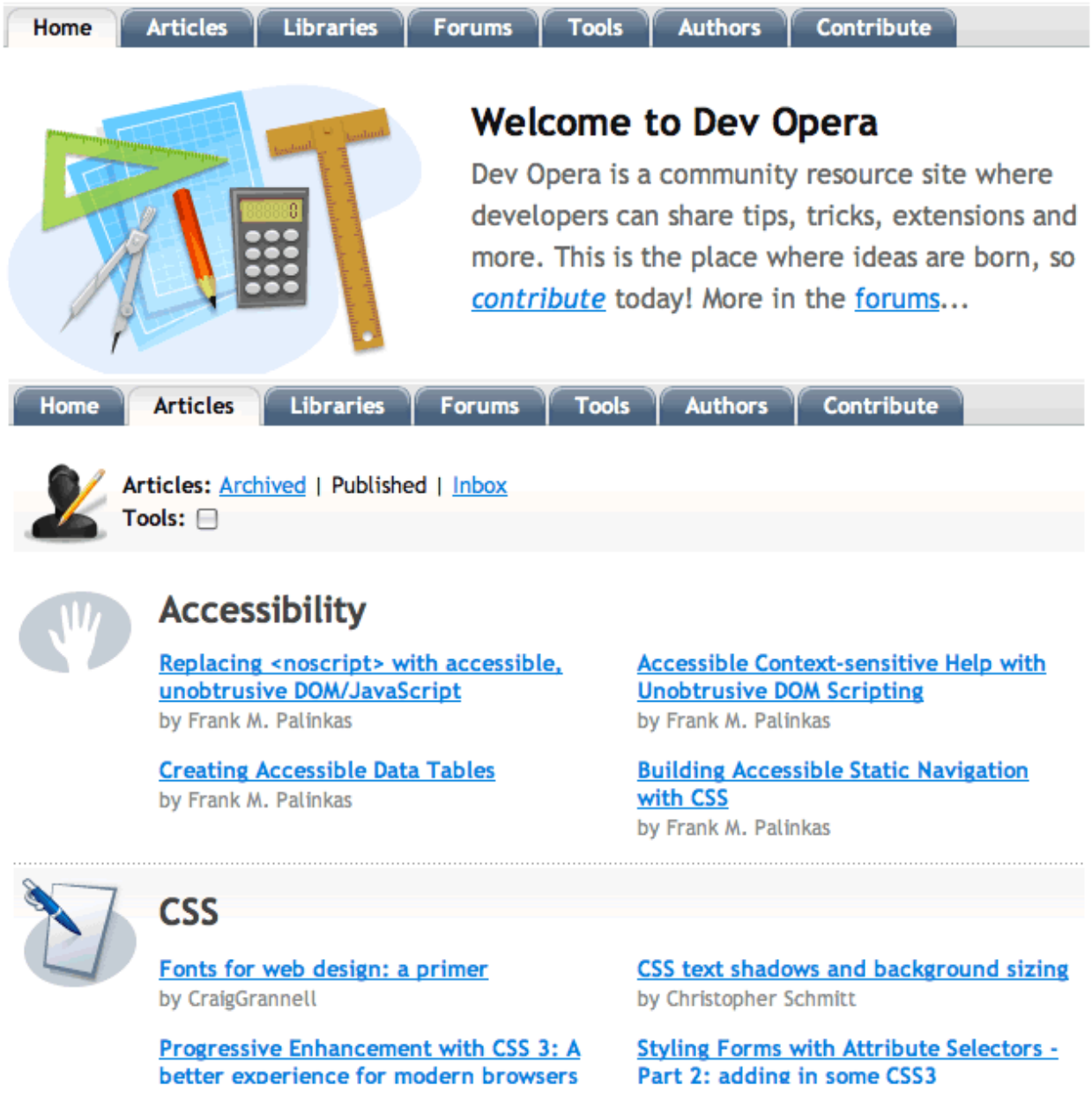
Amíg világos lépések vannak egy linktől a másikig, és a felhasználó megkapja a visszajelzést, hogy továbbra is a jó úton jár, addig nem fogja elhagyni az oldalt emiatt.

Az előző cikkben készített IA struktúrát alapul véve a DB website-jának navigációjában a következő lapokra mutató hivatkozások kaphatnak helyet: „Webáruház", „A Dung Beatlesről", „Kapcsolat", „A Zene", „Banda hírek", valamint egy link vissza a Kezdőlapra. Az ez alá tartozó lapokra, mint például a „Turné információk" vagy a „Dalszövegek", már nem szükséges linkelni. Ezeket a linkeket a megfelelő kategóriában bárki megtalálhatja:

[5] http://www.nytimes.com/2007/09/18/business/media/18times.html

ha valaki egy dalszövegből azonnal a turné dátumokhoz ugrana, az képes lesz ugyanezt a „Banda hírek” navigációs linken keresztül is elérni a „Turné információk” linkre kattintva.

A legkritikusabb része egy sikeres website navigációnak a konzisztencia. Nézd meg például a navigációs füleket ezen az oldalon (mint például „Home”, „Articles”, „Forums”). Ha körülnézel az aloldalakon, a navigációs sáv végig ott marad, viszont megmutatja neked azt is, hogy éppen melyik részén tartózkodsz a site-nak, és további linkeket ad az adott területen belül. Például az „Articles” (leírások) linkre kattintva a leírások főoldalát nyitod meg, ahol megtalálod a legújabb leírásokat, valamint hivatkozásokat az alkategóriákra, mint például a hozzáférhetőség, a CSS vagy a mobil tartalom (lásd a 2.6. ábrát).



2.6. ábra: A dev.opera navigációja a website különböző részein is egyforma

2.2.3 Más gyakori elemek az oldalakon

A navigáción kívül vannak még más olyan elemek is a lapokon, amelyek ismétlődően megjelennek.

Sok oldalnak van logója, céges emblémája vagy valamilyen fejléce, amellyel a tulajdonost jelölik. Például szinte az összes Yahoo! oldalon látni fogsz egy logót a bal felső sarokban, és ez a logó tartalmazza annak a területnek a nevét is a Yahoo! hálózatán belül, amelyikben éppen tartózkodsz (például „Travel", „Movies", „Autos", stb.).

A fejléc (amit a lap teljes felső részén találhatsz) azonban több dolgot is tartalmazhat a logón kívül. Például tartalmazhatja, vagy magához kapcsolhatja a navigációt. Nem ritka a keresőmező sem, amellyel a felhasználók az egész oldalon kereshetnek, elkerülve így a navigációt és a kattintgatást a menükön és hivatkozásokon keresztül. Ezeket az elemeket, vagy legalább egy részüket a website összes lapjára érdemes betenned.

A lábléc (amit a lap alsó részén találsz) további információkat tartalmazhat, mint például a copyright vagy licencmegjegyzések, linkek a website néhány mellékoldalára, ha vannak ilyenek (például „Erről az oldalról", „Felhasználási feltételek", „Kapcsolat", stb.).

A színek, az elrendezés, az ikonok és árnyalatok használata, a tipográfia és a képek kombinációja együttesen azt a benyomást adhatja, hogy ez a lap „része" az egész websitenak: a kulcs itt is a konzisztencia. A konzisztens megjelenés és elrendezés lehetővé teszi, hogy a felhasználó tudja, merre jár, és otthonos hatást kelt. Tudhatod, hogy a lap, amelyen vagy, kapcsolatban áll az egész website-tal, és ugyanazt az élményt nyújtja, mint az előző lap, mivel vizuálisan kapcsolódnak egymáshoz. Amikor a website-ot tervezed, ne feledkezz meg erről, és ne készíts minden lapnak másféle megjelenést.

A mi DB website-unkon a lap fejléce tartalmazni fogja az együttes logóját és nevét, ez megerősíti a látogatókat abban, hogy ugyanazon a website-on járnak, és biztosak lehetnek benne, hogy még mindig az együttes oldalán tartózkodnak. A lábléc copyright információkat fog tartalmazni az oldalról, a dalszövegekről, a képekről és a hanganyagokról, ezen kívül lesz benne egy hivatkozás a kapcsolatokhoz, valamint az együttes megrendeléséhez.

2.2.4 A környezet minden

Minden lap, leszámítva a közös elemeket, egyedi kell legyen. Egy jó website egyik oldala csak egyetlen dolgot, vagy több kisebb dolgot csinál, de azt jól.

Releváns tartalom

A tartalom szétválasztása releváns részekre az egyik legfontosabb dolog, ami egy jó weblapot a legjobbak közé emel. A tartalom egyértelműen címezhető kell legyen (van egy biztos hely, ahol mindig megtalálható, egy egyedi URL-en), és logikusan rendezett (a website-on és az oldalakon belül is), hogy könnyen megtalálható legyen.

A banda közelgő fellépései lehetnek például egy „Fellépések" dobozban is, amelyet minden lapra kitehetsz, de az információnak adhatsz egy saját oldalt is, ahol mindig elérhető lesz. Egy egyszerű „Következő fellépés" modul, amely hivatkozik a Turné dátumok lapra, éppen olyan hatékony, és nem duplázza meg az információkat, valamint nem zavarja össze a látogatókat és a keresőrobotokat.

Fejlécek

Amikor legközelebb a kezedbe kerül egy újság, nézd meg alaposan. Vedd észre, hogy egyes cikkek nagyobbak, vastagabb betűvel vannak írva, képek is vannak benne, vagy feltűnő címeket adtak nekik. Így mutatják meg neked, hogy melyek azok a legfontosabb cikkek, amelyeket érdemes elolvasnod, amikor sietsz, és nem érsz rá mindent végigmazsolázni.

Ugyanez igaz a weboldalakra is. A lap minden részét egy címsor kell bevezessen, amely megmutatja az adott rész relatív fontosságát a lapon belül (ez a rész egy alrésze az előzőnek, vagy ugyanolyan fontosságú?).

Például, ennek a cikknek ebben a részében láthatsz két kiemelt sort: „Releváns tartalom” és „Fejlécek”. Ezek címsorok, és alacsonyabb szinten vannak, mint „A környezet minden” rész, így jelölve, hogy ezek alrészei a környezetről szóló résznek a lapon.

2.2.5 Használhatóság

A használhatóság alatt azt értjük, hogy egy website a józan ész szerint, az elvárt módon működik.

Próbáltál már olyan cikket megnyitni egy hírportálon, amelyikhez először regisztrálnod kell, hogy elolvashasd? Próbáltál már online lefoglalni egy repülőutat vagy rendelni egy vonatjegyet, miközben azt gondoltad, hogy ez is csak maximum két percig tart, mintha csak személyesen vagy telefonon rendelnéd? Írtál már be valaha egy postai címet vagy egy kártyaszámot, hogy aztán később megtudd, hogy más formában kellett volna megadni? Próbáltál már keresni egy oldalon úgy, hogy biztosan tudtad, hogy a keresett szó rajta van az oldalon, de mégsem kaptál találatot?

Ezek mind példák a rossz használhatóságra, amikor nem veszik figyelembe az oldal felhasználóinak igényeit. Ha ezeket az igényeket a design folyamat középpontjába helyezed, akkor nagy valószínűséggel egy megfelelő és kifizetődő weblapot fogsz kapni.

Jól használható weboldalakat viszont nem könnyű készíteni, és a tudás nagy része egyszerűen a tapasztalatból származik. Készíts egy listát azokkal a dolgokkal, amelyek idegesítenek téged más oldalakon, és tanuld meg, hogy hogyan kerülheted el ezeket a sajátodon. Az igazi próba viszont az, ha az oldaladat valódi emberekkel teszteled le. Miután elkészítetted, figyeld meg, hogyan használják az emberek a weboldaladat:

- megtalálják azokat a lapokat, amelyeket keresnek?
- kereséskor azokat a találatok kapják, amelyek a keresésnek megfelelnek?
- a képek, hangok, videók rendben működnek a böngészőjükben?
- mérgelődtek valamin a használat közben?
- van olyasmi, ami különösen tetszett nekik?

A dedikált használhatósági teszt egy olyan dolog, amelyért a professzionális cégek nagyon sokat szoktak fizetni, de valójában egy gyors, kötetlen tesztelés is a barátok és családtagok által nagyon sok hasznos információhoz juttathat, és ráterelheti a figyelmedet olyan problémákra, amelyeket korábban nem vettél észre. Ez azért van, mert te készítetted a weblapot, így mindenről tudod, hogyan kellene használni, hogyan kell működnie - viszont mások ezt nem tudják.

2.2.6 Hozzáférhetőség

A hozzáférhetőség alatt gyakran sajnos csak annyit értenek, hogy „olyan weblap, amit el tudnak olvasni a vakok is”. A valóságban a hozzáférhetőség sokkal, sokkal több emberre vonatkozik.

A „kisegítő technológia” kifejezést az olyan kiegészítő számítógépes eszközökre vagy hardverekre használják, amelyek segítik az embereket abban, hogy könnyebben használhassák a számítógépüket. Ezek a legtöbbször a képernyőolvasókat juttatják eszünkbe, amelyet a vakok és gyengénlátók használhatnak, valamint a hangfelismerőt, amelyet a mozgáskorlátozottak, akik nem tudják használni a billentyűzetet és az egeret. De mi van például a szemüvegesekkel? Azok, akiknek látásjavító eszközre van szükségük, ugyancsak kisegítő technológiát használnak. Mégsem neveznék magukat gyengénlátónak.

Egy jó weblap fejlesztésekor nagyon hasznos tudatában lenni azoknak a problémáknak, amelyekkel az emberek az internet használata közben találkozhatnak. Ne feltételezzük azt, hogy a felhasználóknak van egy egerük; ne feltételezzük azt, hogy a felhasználók látják a lap képeit; ne feltételezzük azt, hogy mindenkinek telepítve van a Flash, hanem biztosítsunk egy alternatív tartalmat is, ha ezek mégsem teljesülnek.

De a kisegítő eszközökkel böngésző emberek mellett mások is vannak, akikre a fenti korlátozások vonatkozhatnak, mégpedig a mobiltelefont használók. A Flash például még egy nagyon idegen technológia a mobiltelefonokon, az Apple iPhone nem támogatja, és valószínűleg nem is fogja, mégis ugyanúgy lehet vele böngészni a netet, mint egy Safarival egy Macen (az Opera Mobile támogatja a Flasht). Az Opera Minihez hasonló technológiák alsókategóriás telefonokon működnek, és egy egyszerűbb, de hatékonyabb módon érik el a webet, amelyben viszont a képek egyáltalán nem, vagy csak sokkal kisebb méretben jelennek meg, mint eredetileg. Ez azt jelenti, hogy a képekre betervezett apró részletek elveszhetnek.

A mi együttesünk esetében mindez azt jelenti, hogy ha sok képet jelenítünk meg (fényképek a bandáról), akkor mindig le kell írnunk a kép tartalmát is. Ha egy beépített Flash zenelejátszót használunk a lapon, amellyel az emberek belehallgathatnak az együttes zenéjébe, akkor ki kell tennünk egy alternatív direkt hivatkozást is a zenére, hogy azok is meghallgathassák, akiknek nincs Flash telepítve.

2.2.7 Tesztkérdések

Ennek a leírásnak a tesztkérdéseihez egyszerűen csak a webet kell használod: látogasd meg néhány kedvenc weboldaladat, és próbáld meg őket most más szemmel nézni. Készíts jegyzetet az alábbi kérdésekre válaszolva:

- Következetesen használják a fejlécet, a láblécet és a navigációs területeket?
- Figyeld meg, hogyan változik a navigáció, ahogy böngészed az oldalt.
- Figyelj arra, ha valami a lapon idegesít vagy összezavar; ha ilyet találsz, próbálj rá megoldást találni, amellyel elkerülhető lenne a probléma.
- Ha tudod hogyan kell, kapcsold ki a képeket vagy a JavaScriptet a böngésződben, próbáld ki a lapot egy mobiltelefonos böngészőben, majd hasonlítsd össze az élményt azzal, mint mikor a lapot a megszokott körülmények között használod.

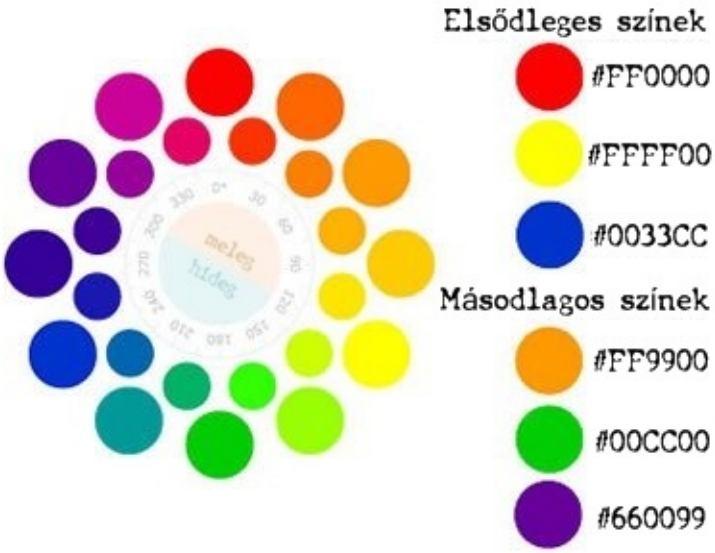
2.3. A színek elmélete

Színek és grafikák nélkül minden oldal Jakob Nielsen[6] álomoldala lehetne. Habár Nielsen filozófiája a lecsupaszított weboldalakkal igencsak hasznos a hozzáférhetőségi és használhatósági szempontok szerint, mégis a legtöbb webdesigner a felhasznált design elemekkel ott szeretné hagyni a keze nyomát azon, amin dolgozott. Szerencsére egy jó design úgy vihet színt egy weboldalra, hogy közben nem veszítjük el a használhatósági és hozzáférhetőségi előnyöket, feltéve, hogy a weblapot eleve így terveztük meg. Bár sok designernek semmilyen gondot nem okoz egy többfelhasználós website tervezése, mégis sokan kényelmetlenül érzik magukat, amikor ki kell választják a színeket és a grafikákat.

Ebben a leírásban átnézzük a színek alapjait és három egyszerű színsémát, amelyek alapján már képes leszel megtalálni a megfelelő színeket az oldaladhoz. Később lesz még egy másik leírás is a témáról, amelyben arról lesz szó, hogy hogyan választhatsz magadnak egyszerűen színeket. Végül is jobb a dicséreteket hallgatni az új webdesignnal kapcsolatban, mint vért izzadni a színválasztással.

2.3.1 Színek és árnyalatok

A színeket és az árnyalatokat már régóta felosztották elsődleges, másodlagos és harmadlagos színekre. Az elsődleges színek a piros, a sárga és a kék, amelyek azért elsődlegesek, mert az előállításukhoz nem szükséges színeket összekeverni. Ha ezeket a színeket webes színekre akarod alakítani, akkor a hexa (hexadecimális) értéküket kell megadnod: #ff0000, #ffff00, és #0033cc, amint az a 2.7. ábrán látható:



2.7. ábra: Az elsődleges és másodlagos színek, és a hozzájuk tartozó hexa értékek

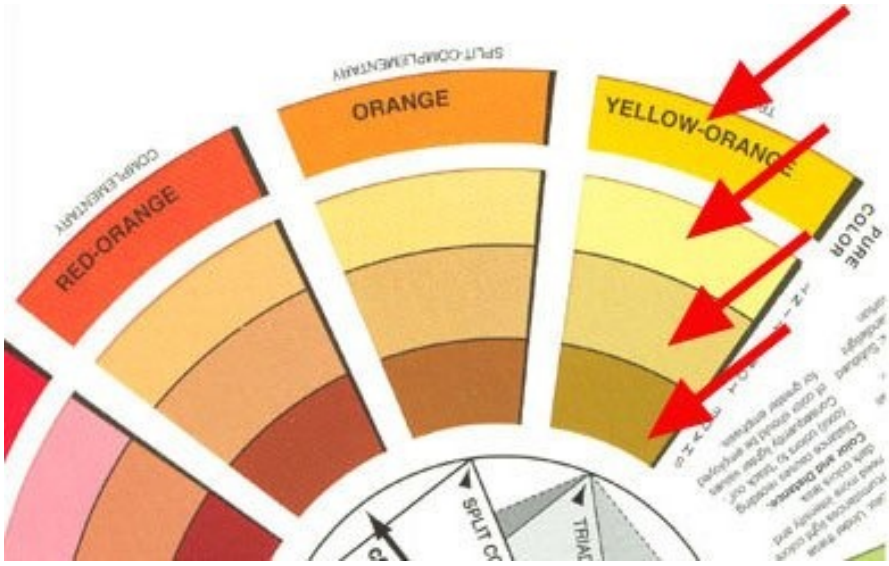
A másodlagos színek az elsődleges színek keverékei, mégpedig a következők:

- piros + sárga = narancssárga (#ff9900)
- sárga + kék = zöld (#00cc00)
- kék + piros = lila (#660099)

[6] http://www.useit.com/

A harmadlagos színek már a másodlagos színek keverékei, és az elsődleges és másodlagos színek között helyezkednek el, amint az az alábbi színkeréken látható. Bár a webes színek mások, mint a hagyományos „festett” színek, még jó lehet, ha kéznél van a színkerék (lásd a 2. ábrán is), amikor a különböző színsémákkal ismerkedsz. A színkerék ezen kívül megmutatja a színek világos, szürke és sötét árnyalatait (tónusait), így képet alkothatsz arról, hogy milyen színek állnak majd a rendelkezésedre. Néhány fontos fogalmat érdemes letisztáznunk:

- világos árnyalat – a kapott szín, amikor fehéret adunk hozzá
- szürke árnyalat – a kapott szín, amikor szürkét adunk hozzá
- sötét árnyalat – a kapott szín, amikor feketét adunk hozzá



2.8. ábra: Egy valódi, nyomtatásnál használt színkerék

A nyilak a 2.8. ábrán különböző dolgokat jelölnek:

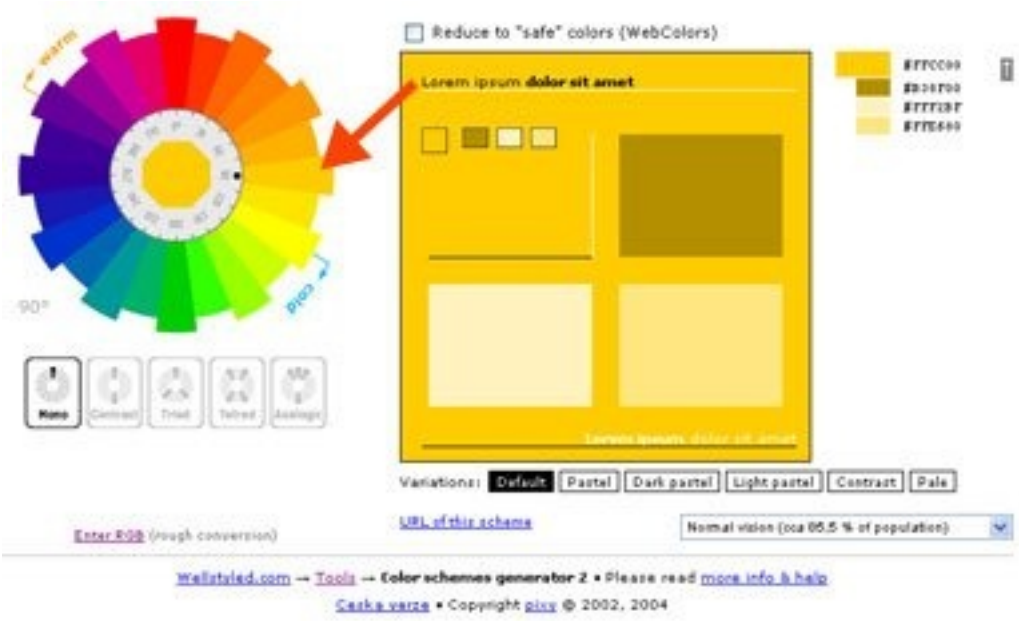
- külső csík – harmadlagos sárga-narancs szín (sárga + narancssárga)
- második csík – az eredeti szín világos árnyalata (fehéret hozzáadva)
- harmadik csík – az eredeti szín szürke árnyalata (szürkét hozzáadva)
- belső csík – az eredeti szín sötét árnyalata (feketét hozzáadva)

Amint az a színkeréken láthatod, elég egy kevés fehér, szürke vagy fekete színt hozzáadni a színhez, hogy azt megváltoztassuk, így hozhatjuk létre a monokróm néven ismert színsémát.

2.3.2 Monokróm színsémák

Színsémákat már évszázadok óta használnak, úgyhogy nem kell újra feltalálnunk a színkereket. Habár a webes színek különböznek a nyomtatási színektől, ugyanazt az alapkoncepciót használják. Egyszerűen cseréld fel a hexa értékeket a színek nevével, és próbáld őket minél jobban behatárolni. Ehhez egy online eszközt tudok ajánlani, ez a **Color Scheme Generator II**[7], amit a 2.9. ábrán láthatsz, ezzel gyorsan és egyszerűen meghatározhatod a színsémákat, valamint azt is ellenőrizheted, hogy a kiválasztott színek biztosítanak-e elegendő kontrasztot a gyengén látó vagy a színvak felhasználók számára is.

[7] http://www.wellstyled.com/tools/colorscheme2/index-en.html



2.9. ábra: Color Scheme Generator II

Ha további információt szeretnél arról, hogy az általad választott színek elegendő kontrasztot nyújtanak-e, próbáld ki a Paciello Group **Contrast Analyser**[8] eszközét. Ezzel az eszközzel ellenőrizheted a kontrasztot a háttérszín és a szöveg színe között.

Ha elő akarod állítani az előbbi sárga-narancs színnek a különböző árnyalatait az online színséma készítővel, először válaszd ki azt a színt, amelyikre a piros nyíl mutat a képen. Ezután a színkerék alatti panelen válaszd ki a Mono gombot, majd a jobb alsó panelen a Default gombot. A jobb alsó lenyíló dobozban válaszd a Normal vision pontot. Ha nem vagy minimalista, akkor ne jelöld be fent a Reduce to "safe" colors (csak „biztonságos webes színek”) pontot.

A „biztonságos webes színek” kifejezés még abból az időből származik, amikor a számítógépek képernyői csak 216 színt voltak képesek helyesen megjeleníteni. Ez a 216 szín ugyanúgy jelent meg a különböző platformokon és a 256 színnel működő (8 bites) böngészőkben, így a használatuk a platformok között is biztonságosnak számított. Néhány minimalista még mindig használja ezt a „webbiztos színpalettát”, bár a modern webböngészők már képesek a 24 bites színek helyes megjelenítésére is. A jelenlegi 24 bites, csatornánként 10-11 bites színábrázolás 16 777 216 színt képes megkülönböztetni. Más szóval, ma már biztonságosan kijelenthetjük, hogy a „webbiztos színpalettára” többé nincs igazán szükség.

Térjünk vissza a monokróm színsémára. A fenti lépések elvégzése után a következő eredményt kell kapnod: sárga-narancs (#FFCC00), világos árnyalat (#FFF2BF), szürke árnyalat (#FFE680), és sötét árnyalat (#B38F00). Ezek a hexa számok jóval megbízhatóbbak annál, mintha egy igazi színkereket próbálnál meg a böngésző színeihez igazítani. Amint a „Mono” előtag sugallja, ezek a színek egy monokróm színsémát állítanak elő, amint az a 2.10. ábrán láthatod.

2.10. ábra: Egy monokróm színséma

[8] http://www.paciellogroup.com/resources/contrast-analyser.html

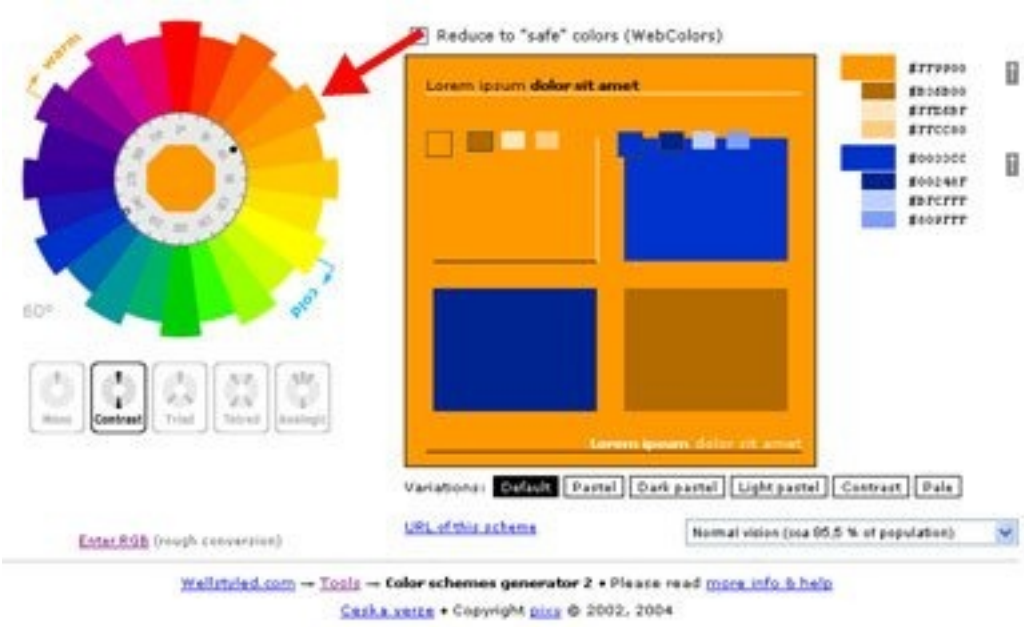
Egy monokróm színséma egyetlen színből és annak világos és sötét árnyalataiból áll. Bár ezt a színsémát elég egyszerű használni, a legtöbb webdesigner nem lelkesedik érte. Nézzünk meg néhány más színsémát is, amelyekkel feldíszítheted a hivatkozásokat, képeket vagy reklámcsíkokat.

2.3.3 Kiegészítő színsémák

A következő színséma család, amelyet jobban megnézünk, a kiegészítő színséma, amelyben mindig a színkerék ellentétes oldalán található színeket párosítjuk össze, ahogy azt a 2.11. ábrán láthatod.

2.11. ábra: Példák kiegészítő színsémákra

Amikor kiválasztasz egy bizonyos színt és az ellentétes párját, akkor velük együtt kiválasztod mindkét szín sötét és világos árnyalatait is. Ez jóval nagyobb választási lehetőséget biztosít, és jól használható az online színséma készítővel is — nézd meg a 2.12. ábrán.



2.12. ábra: Az online eszközzel készített kiegészítő színséma

A fenti képen a nyíllal jelölt narancssárga szín választottam, amelynek az ellentétes színe a kék. Ehhez a színsémához az alsó panelen a Contrast gombot kell kiválasztani, a jobboldali panelen a Default gombot, és a lenyílóban a Normal vision pontot. A kiválasztott fő színt egy kis fekete pötty, míg az ellentétes színt egy kis kör jelöli a színkerék belső részén. Ezekkel a jelölőkkel könnyebben tudod elemezni a választott színsémát.

A színkészítő segítségével könnyen választhatsz színt a normál és a látogatott hivatkozások vagy a képek színeihez, mivel megadja a színek hexa kódját a jobb felső sarokban.

Bármelyik normál színt választhatod a felső színek közül, valamint kombinálhatod őket a színek világos, szürke vagy sötét árnyalatával, így egy megbízható színsémát készíthetsz.

2.13. ábra: Greenpeace oldala jó példa a kiegészítő színséma használatára

A **Greenpeace USA**[9] oldala (a 2.13. ábrán) egyike annak a rengeteg oldalnak, amely kiegészítő színsémát használ. Igen, láthatsz rajta sárgát és narancsot is, de a domináns színek a zöld és a piros — ezek a színek éppen a színkerék ellentétes oldalain állnak. Egy ilyen színsémával nem igazán ronthatod el a dolgokat. A „hideg” és „meleg” színek kombinációja viszont még jobban feldobhatja az oldaladat a színek kontrasztja által.

2.3.4 Hideg kontra meleg színek

A kiegészítő színsémák azért is nagyon népszerűek a weblapokon, mert egyszerre tartalmaznak hideg és meleg színeket. Az ilyen színek használata erős kontrasztot okoz, és nagyon könnyű megjegyezni, hogy melyek a „hideg” és melyek a „meleg” színek, ahogy azt a 2.14. ábrán, vagy a színséma készítő oldalon is láthatod:

[9] http://www.greenpeace.org/usa/

2.14. ábra: Hideg és meleg színek

A meleg színek azok, amelyek a nyárra, a napra vagy a tűzre emlékeztetnek. Ezek a lilától kezdve egészen a sárgáig tartanak. A hideg színek viszont a tavaszra, a jégre vagy a vízre emlékeztetnek. Ezek a színek a zöldessárgától vissza a liláig találhatóak a színkeréken. Hamar felismerheted, hogy nem választhatsz egyetlen színt sem anélkül, hogy ne választanád ki az ellentettjét a másik „hőmérsékletből". Ha kiválasztod a tűzforró vöröset, akkor egy hideg zöldet kapsz a másik oldalon. Ha viszont egy hűvös kékeszöldet választasz, akkor egy csípős piros-narancsot fogsz találni vele szemben.

Az **Ecolution**[10] jó példa egy olyan oldalra, amelyik következetesen használja a hideg és meleg színek kombinációját:

2.15. ábra: Az Ecolution oldala jó példa a hideg és meleg színek használatára

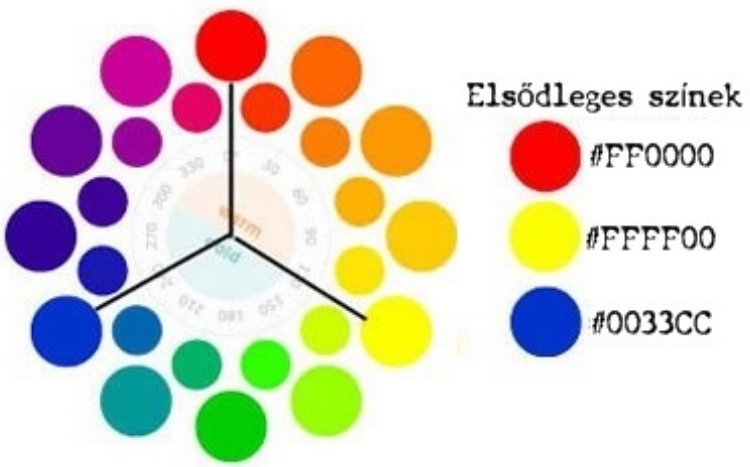
[10] http://www.ecolution.com/

Az Ecolution általában pirosat használ hangsúlyos színnek, amely kontrasztban áll a zöld logójukkal. Az oldalon ezt a két színt és ezek árnyalatait keverik mindenütt. Még a képük sötét részei is „melegek” vagy „hidegek”, hasonlóan a világos részekhez. Összességében a fénykép „meleg”, ami így nagyon jól illeszkedik a határozott, erős zöldhöz. Bár az Ecolution ugyanazokat a színeket használja, mint a Greenpeace, mégsem „fénylik” úgy az oldaluk, köszönhetően a sötét és világos árnyalatok gazdag használatának.

Soha nem gondoltad volna, hogy a színelmélet ilyen egyszerű, nem igaz? Hát akkor… ideje lesz egy kicsit bonyolítani a dolgot…

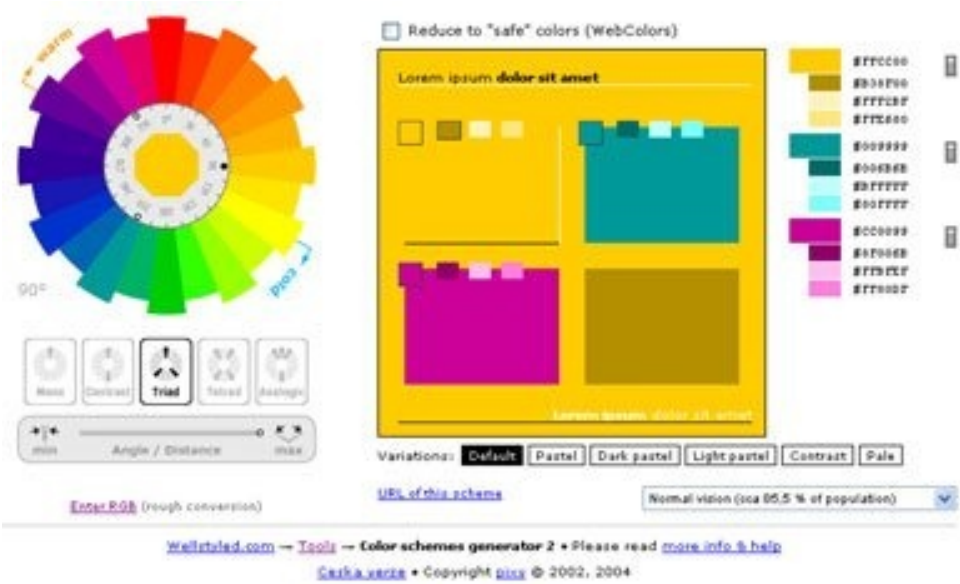
2.3.5 Harmadolós színsémák

Harmadolós színsémát (lásd a 2.16. ábrán) úgy készíthetsz, hogy kiválasztasz egy színt a színkerékről, majd további két színt úgy, hogy a három szín egyenlő távolságra legyen egymástól:



2.16. ábra: Egy harmadolós színséma

Azért választottam az elsődleges színeket, hogy megmutassam, a színsémák egy ragyogó módszert szolgáltatnak az őrülethez. Nem véletlenül kerültek az elsődleges színek éppen ezekre a helyekre a színkeréken, mivel mindegyik elsődleges színnek pontosan ugyanannyi másodlagos és harmadlagos színe van, mint a többi elsődlegesnek. De az elsődleges színekből készített harmadolós színséma már egy elhasznált, vaskalapos megoldás. Válasszunk helyettük inkább egy másik színt az online színséma készítőben, a 2.17. ábra szerint:



2.17. ábra: Egy alternatív harmadolós színséma

A fenti harmadolós színséma sárga-narancssárga, kék-zöld és piros-lila színekből készült. Először a sárga-narancssárga színt választottuk ki (ezt a színkerék belső felén egy kis fekete pötty jelöli), majd az alsó panelen kiválasztottuk a **Triad** gombot. Az eszköz ezután automatikusan kiválasztotta a harmadolós színeket, valamint azok sötét és világos árnyalatait. A másik két választott színt a színkerék belső felén két kis kör jelöli, hasonlóan a monokróm példához.

Ebben az esetben jól jöhet egy valódi színkerék, mivel az online értékek nem felelnek meg annak, amit egy igazi színkerék produkál. A jobb eredményhez a bal alsó részben található **Angle / Distance** csúszkát fel kell húzni a maximumra. Így a fenti képen látható színeket kapjuk, amelyek már megfelelnek az igazi színkerék színeinek.

A harmadolós színsémában ugyancsak vannak hideg és meleg színek is, de az egyik hőmérséklet domináns lesz. Általában az a hőmérséklet lesz domináns, amelyikből az eredeti színt választottad. Ebben az esetben eredetileg egy sárga-narancssárga színt választottunk, amely egy meleg szín. A végeredményben a meleg színek jobban érvényesülnek, és a két ellentétes szín közül az egyik egy hideg kontraszt lett.

2.18. ábra: A Puzzle Pirates jó példa a harmadolós színsémára

A 2.18. ábrán látható **Puzzle Pirates**[11] oldala harmadolós színsémát használ a főoldalán. Ők az elsődleges piros-kék-sárga színsémát használják, és ez a színséma tökéletes egy gyerekoldal számára. Figyeld meg, hogy a kék szín érvényesül jobban, a pirosak és a sárgák hangsúlyozottabbak, ezek vezetik a tekintetet körbe az oldalon.

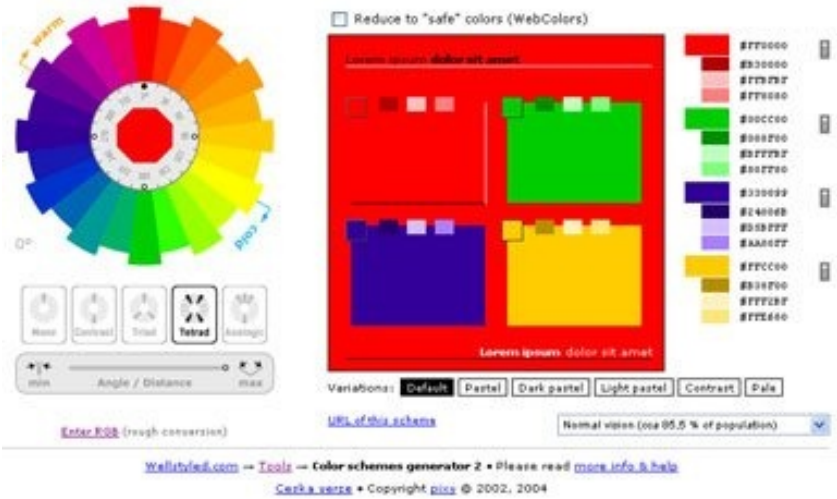
2.3.6 Negyedelős színsémák

Minél több színt választasz, annál bonyolultabb lesz a színsémád. Viszont van egy egyszerű trükk: elég, ha kiválasztasz mindegyik színhez egy világos vagy sötét árnyalatot, és aztán ehhez tartod magad az egész oldalon ahelyett, hogy a sima színeket és azok különböző árnyalatait kevernéd. Ez a módszer nagyon jól működik az olyan színsémák esetében, amelyek már négy színt használnak. A 2.19. ábrán látható séma hasonló a kiegészítő színsémához, viszont két kiegészítő színsémát tartalmaz, amelyek egyenlő távolságra vannak egymástól.

2.19. ábra: Egy negyedelős színséma

A 2.20. ábrán az online eszközzel készítettünk egy ilyen negyedelő színsémát:

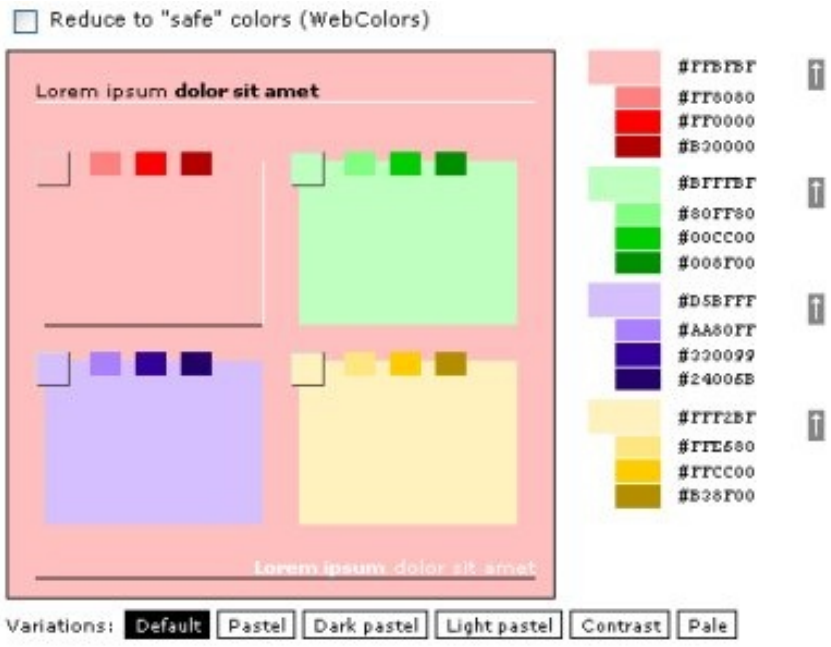
[11] http://www.puzzlepirates.com/



2.20. ábra: Egy online készített negyedelős színséma

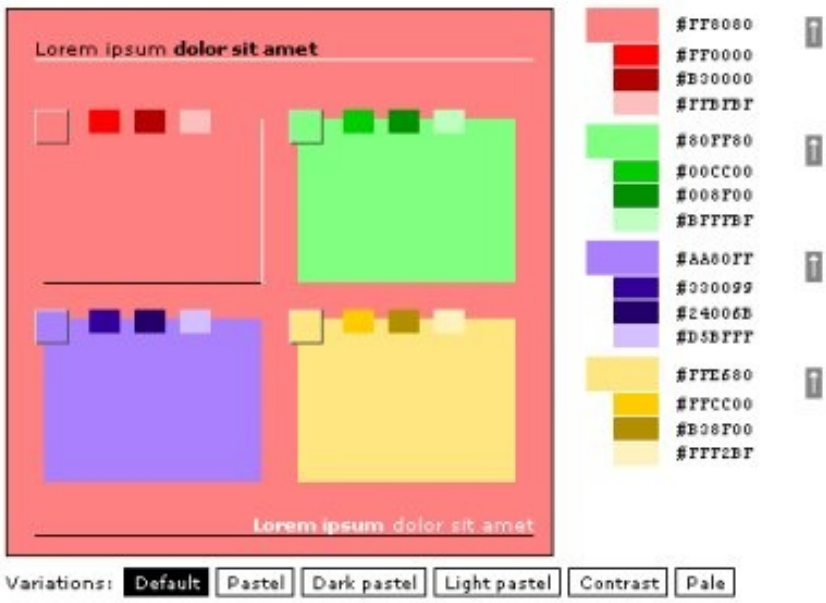
Figyeld meg a színkerék belső felén a fekete pöttyöt a piros alatt. Ez volt az első szín, amit kiválasztottunk, ezután az alsó panelen kiválasztottuk a **Tetrad** gombot. A négy szín, amely ezután megjelent, most sem felelt meg azoknak a színeknek, amelyeket a valódi színkerék mutatott a kezemben, így az **Angle / Distance** csúszkát még fel kellett húzzam a maximumra, ekkor a színek már megfeleltek a valódi színkerék színeinek. Az így kapott színeket láthatod a fenti képen.

Ez a színséma már elég bonyolultnak látszik, így az egyszerűsítés érdekében ki kell választanunk négy árnyalatot a választott színekből az eszköz jobb oldalán. A színeket a mellettük látható felfelé mutató nyilak segítségével váltogathatod. A 2.21. ábrán láthatsz egy példát, amelyben a négy alapszín világos árnyalatait választottuk ki ezzel a módszerrel:



2.21. ábra: Negyedelős világos árnyalatok

A 16. ábrán egy újabb példát láthatsz az eredeti színek szürke árnyalatait választva:



2.22. ábra: Negyedelős szürke árnyalatok

Ha jobban megfigyeled, láthatod, hogy az online eszköz mindegyik színhez előállítja neked a monokróm színsémáját is. Ezeknek a színsémáknak a színeit az eszköz jobb oldalán, vagy színek nagy négyzeteinek a felső részén láthatod.

2.23. ábra: A Jane Goodall Institute oldala jó példa a negyedelős színséma használatára

A **Jane Goodall Institute**[12] oldala (a 2.23. ábrán) egyike azon kevés oldalaknak, amelyek igazán jól használják a negyedelős színsémát. Figyeld meg a lila és sárga szürke árnyalatokat, a piros kiemeléseket a fényképen (az oldal többi, itt nem látható részén még több ilyen van), és a zöldeket. Az itt használt lila szín nem felel meg egészen pontosan az online eszköz által generált színsémában található színnek (amely inkább egy piros-lila),

[12] http://www.janegoodall.org/

de elég közel van ahhoz, hogy megmutassa, hogyan használhatod az online eszközt vagy az igazi színkereket az oldalad színeinek előállítására.

Mostantól, ha a webet böngészed színek és design ötletek után, mindig legyen a kezed ügyében az igazi vagy az online színkerék, mivel ezekkel te is megnézheted, hogy milyen színsémát használnak valójában a kedvenc weboldalaid!

2.3.7 Tesztkérdések

Megjegyzés: az utolsó két kérdésre nincs „jó” vagy „rossz” válasz.

- Nevezd meg a három elsődleges színt, és magyarázd meg, hogy miért nevezik őket elsődlegesnek.
- Nevezd meg a három másodlagos színt, és hogy melyik elsődleges színekből lehet előállítani őket.
- Magyarázd meg, hogyan lehet előállítani a világos, szürke és sötét árnyalatokat.
- Milyen a monokróm színséma?
- Milyen a kiegészítő színséma?
- Magyarázd meg, mik azok a „hideg” és „meleg” színek.
- Milyen a harmadolós színséma? Válassz három színt, amelyek megfelelnek ennek a sémának.
- Milyen a negyedelős színséma? Válassz négy színt, amelyek megfelelnek ennek a sémának.
- Melyik színsémát tudod a legkönnyebben használni?
- Melyik színséma tűnik számodra a legbonyolultabbnak?

2.4. Egy site keretének felépítése

Minden webdesignernek ismernie kellene és meg kellene értenie a websiteok paramétereit, mielőtt egyáltalán belekezdene a kódolásba. Ebben a leírásban meg fogod ismerni az üzleti weboldalak készítésének alapjait. Bár ezek az információk elsősorban a mások számára készített weblapokra vonatkoznak, de felhasználhatod akkor is, amikor a saját oldaladat tervezed meg. Ez a lépés általában azonnal az információs architektúra után következik, össze kell gyűjtened azokat az információkat, hogy mit szeretnének tenni a kliensek a weblapra, hogyan képzelik el a struktúráját, milyen márkajegyeket kell felhasználni, majd ezeknek az információknak az alapján készíthetsz egy vizuális designtervet, amelyet már meg tudsz mutatni a kliensnek még mielőtt elkészítenéd a grafikákat és a használni kívánt színsémákat. Egészen pontosan az alábbiakról fogunk beszélni:

- Bár a színek és a design nagyon fontosak, először tudnod kell, hogy a kliens mit szeretne elérni a website-tal. Ennek az információnak aztán tükröződnie kell a site megjelenésében is.
- Ezért szükséged van egy ellenőrző listára, amelyen végig kell futni a design elkészítése előtt.
- Meg kell ismerned azt is, hogy korábban milyen marketing tevékenységek voltak a cégnél, beleértve a márkajegyeket is. Ez később ugyancsak hatással lehet a website designjára.

- Az klienstől összegyűjtött információk alapján elkészítheted a website első vizuális designtervét, amelyet megmutathatsz a kliensnek, így megalapozhatod a további grafikai- és tartalomterveket.

2.4.1 Mit kell tudnod?

Általában mielőtt döntenének egy website designjáról, a fejlesztőnek szüksége van valamilyen tervre azzal kapcsolatban, hogy az adott website-tal mit szeretnének elérni. Bár a színek és a grafikák is fontosak, előbb szükség van egy tervre a költségekről, a megcélzott piacról, a tervezett célokról valamint a feladatokhoz szükséges erőforrásokról. Vajon a site csak információt fog szolgáltatni a felhasználóknak, vagy valójában az a célja, hogy termékeket és szolgáltatásokat adjon el? Fog később növekedni a website, vagy csak rövidtávon akarják majd használni egy kisebb piaci szegmens elérésére? (Mint például a politikai kampányoldalak, vagy egy olyan oldal, amelyik csak be akar lépni az aktuális trendbe.) Lesz-e az oldalon blog, információs oldal, képgaléria, e-mail kapcsolati űrlap? Mire lehet még szükség rajta? Össze lehet majd hasonlítani az oldalt a konkurenciájával?

Végül, de nem utolsósorban tudnunk kell azt is, hogy van-e a cégnek márkajelzése és marketinges iránymutatója. Ha nincs, akkor erre különösen oda kell figyelni még a design tervezése előtt. A logó, az árucikkek vagy a szolgáltatások márkája egy bizonyos piacra, valamint ennek a piacnak az elérése túlmutathat a képességeiden. Ha még soha nem próbáltad ezt korábban, akkor jobb, ha bevonsz egy szakértőt, hogy jó irányba indulhass el. Másrészt, ha a cégnek már vannak ilyen irányú útmutatásai, akkor nagyon fontos, hogy ezeket szigorúan kövesd, hogy a weboldal jól illeszkedjen a többi marketinges anyag mellé.

Mivel ezeknek az információknak a nagy részét még azelőtt meg lehet szerezni, hogy a tervezett oldal eljutna a designerhez, a válaszok sokat segíthetnek abban, hogy eldönthesd, milyen típusú designra lesz szükség, milyen színsémát és milyen grafikákat használj az oldalon. De egy dolgot mindenképpen ki lehet jelenteni az esetek többségében: a website hozzáférhető és használható kell maradjon, emiatt a kód és a navigáció különleges figyelmet és magasabb prioritást igényel. A hozzáférhetőségről még többet is olvashatsz később ezen a kurzuson, vagy részletesebben is olvashatsz a használhatóságról Jakob Nielsen oldalán.

A cél az, hogy az oldal a HTML és a CSS használatával a kódoláshoz és a designhoz továbbra is egyszerű maradjon. Próbáld meg elkerülni a Flasht, leszámítva az oldal egyes elemeit, ahol esetleg hasznos lehet (sokat tettek már azért, hogy a Flash is hozzáférhető legyen, és bizonyos feladatokhoz nagyon jól illeszkedik, mint például a videó), valamint gondolkodj el azon, hogy hol lesz szükséged JavaScriptre vagy más technikai dologra. Így egyszerűbb lesz az oldal tervezése a designer és a programozó számára (főleg, ha a designer egyben a programozó is), és sokkal inkább kompatibilisebb lesz a különböző böngészők között.

2.4.2 Az első lépések

Hogy jobban megértsd ezeket a dolgokat, készíteni fogunk egy egyszerű üzleti oldalt egy olyan útmutatást követve, amelyet én a saját és a másoknak készített weblapjaim tervezésekor szoktam használni. Ez a feladatlista tartalmaz üzleti vonatkozású pontokat és design témákat is. Az egyszerűség kedvéért egy olyan elképzelt üzleti vállalkozást fogunk használni, amelyik már készített korábban marketing anyagokat. A nyomtatott anyagaikban már szerepel logó és vannak márkajegyeik is. Ha ilyenek még nincsenek, akkor ezeket még azelőtt meg kell tervezned, mielőtt nekifognál a design elkészítéséhez.

Mint webdesigner, egy website designjának elkészítése előtt az alábbi információkat szeretném tudni az üzletről. Szeretnék készíteni egy listát mindarról, amit be akarok venni a website designjába, így később már nem lesz szükség radikális változtatásokra. Ez már nem egy elképzelt szituáció, az alábbi pontokat mindenképpen érdemes átbeszélned a cég tulajdonosaival vagy a döntéshozókkal, hogy biztos lehess benne, hogy a te elképzelésed megfelel az ő elképzelésüknek is.

1. **A website neve**: tükrözi ez a név a vállalatot és az online törekvéseit? A mi esetünkben a website neve egyben a cég neve is, ami „Wiki Whatevers". A cégnek esetleg egy szlogenre is szüksége lehet, ha még nincs nekik. A szlogen aztán a cég nevével és a logóval együtt felkerül a weblapra.
2. **Logó és márkajegyek**: össze szeretném gyűjteni az összes olyan nyomtatott anyagot, amelyet korábban készítettek, logókat, brosúrákat, stb. Így készíthetek egy fájlt, amelyben hasznos információkat, címeket, telefonszámokat tárolhatok. Ezeknek a segítségével jobban megérhetem a cég „hangját", stílusát, márkajegyeit. Ha korábban semmi ilyesmivel nem foglalkoztak még, akkor valószínűleg megbízok egy logókészítő csapatot, hogy készítsenek egy logót (mivel én nem vagyok logó designer, így továbbadom a munkát - ennek az árát ilyen esetben bele lehet építeni a költségekbe).
3. **A website domainneve**: a website neve mellett tudni szeretném, hogy van-e már jelenleg valamilyen domainneve a cégnek. A domainnév a website címe, amellyel azonosíthatóvá válik, ezt a címet kell a felhasználóknak beírniuk a böngészők címsorába, ha meg akarják nyitni a weblapot. Domainneveket használunk akkor is, amikor különböző weblapokat és elemeket linkelünk a weblapokon. A domainnév egy vagy több felső szintű domain regisztrációt is tartalmazhat, mint például „.com", „.hu" vagy „.org", stb. Bár a domainnév kiválasztása nem a designer felelőssége, nem árt tudni arról, hogy választottak-e már egy domainnevet és regisztrálták-e. Egyes esetekben át kellett írjuk a domain nevet, mert később kiderült, hogy már valaki lefoglalta, és ez végül többletköltséget okozott. A problémát el lehet kerülni, ha a domainnevet idejében lefoglaljuk.
4. **Konkurenciakutatás**: mindig jó tudni arról, hogy mi van a konkurencia weboldalán a grafika és a tartalom szempontjából, hogy az új weblap is legalább olyan jó vagy még jobb legyen, amikor elindul a piacon.
5. **Információs architektúra**: szükség van az oldalon webáruházra vagy blogra? Milyen elképzelései vannak a megrendelőnek a website fejlesztéséről? Milyen struktúrában kapcsolódjanak egymáshoz a lapok? Ezek az elemek fontosak, mivel be kell építened a designba és a navigációba is. Tudnod kell, hogy a site milyen irányban fog fejlődni a jövőben, mivel ez befolyásolja a jelenlegi tervet is.
6. **A site tartalma**: elkészült már valamilyen tartalom az oldalra? Ha igen, akkor valószínűleg azonnal hozzáférést szeretnél kapni hozzá, hogy elkészíthesd a navigációt, a designt és az elrendezést. A tartalom kategorizálása a legjobb módszer a navigáció kialakításához. A tartalom segíthet kialakítani a site megjelenését is, így jó ötlet lehet egy időre elhalasztani a design tervezését, ha még nincs tartalom az oldalra. Győződj meg róla, hogy a tartalom még mindig releváns és készíts tervet a frissítésre, mivel a site tartalma az, ami miatt a látogatók leginkább visszatérnek majd az oldalra.
7. **Webes szolgáltatók keresése**: bár lehetséges, hogy a kliensnek már van ötlete a webszolgáltatóra, az nem biztos, hogy neked is meg fog felelni, mivel a szolgáltatók technikai támogatása nem egyezik meg minden esetben. A webszolgáltatás is csak egy üzlet, ami elsősorban a websiteok hostolásával foglalkozik, de egyesek

adnak adatbázis támogatást is, amire szükséged lehet például egy blog esetében vagy az információk és termékek katalogizálására egy webáruház fejlesztésekor. Más szolgáltatók korlátozzák a látogatók számát az oldalra, ami különösen problémás lehet, ha a website valamiért hirtelen népszerű lesz. Ha keresni akarsz egy webszolgáltatót, és kíváncsi vagy rá, hogy miben különböznek egymástól, látogass el a Web Host Database (WhDb) oldalára. Győződj meg arról is, hogy a kliensnek van-e tárhelye a szolgáltatónál, mielőtt nekilátsz a design elkészítésének, hogy láthasd a határokat.

8. **Irányított távozás**: ez azt jelenti, hogy megpróbálhatod az ügyféllel együtt befolyásolni azt, hogy hogyan hagyják el a felhasználók a websiteot. Előbb-utóbb minden látogató elhagyja az oldalad, hát akkor miért ne próbálnád meg irányítani ezt és nyerni belőle, például reklámokkal és linkcserével? Ha már most készítesz erre terveket, az értékesebbé teszi a honlapot, és pénzügyileg is megtérül később.
9. **Határidők**: döntsd el már most, hogy a website mikor fog „beindulni”. Általában egy ilyen kis projektre egy nyolc hetes fejlesztési idő elég szokott lenni, feltéve hogy az ügyfélnél már rendelkezésre áll a tartalom, alkalmasnak találják a színeket, az elrendezést és a designtervet, amit mutatsz nekik, és nincs szükség bonyolult programozásra sem.

Miután ezeken az alapdolgokon túl vagy, ülj le nyugodtan, olvasd el a tartalmakat, amelyek már a rendelkezésedre állnak, tervezd meg a navigációt és döntsd el, hogyan fogod optimalizálni az oldalt a keresőmotorok szempontjából. Ha nem ismered különösebben a SEO-t (Search Engine Optimization, keresőoptimalizálás), akkor beszélj egy SEO szakértővel is arról, hogy hogyan használhatod fel az oldal tartalmát arra, hogy nagyobb forgalmad legyen, és hogyan használd a potenciális keresőszavakat a tartalomban, a fejlécekben és a címekben.

Ahogy egy új házban sem veszel addig szőnyeget meg kanapét, amíg a tervező nem készíti el az első vázlatot, ugyanúgy egy weblap esetében sem foglalkozhatsz a vizuális designnal addig, amíg nem készítetted el a site felépítését. A navigáció és a SEO tervek elkészítése ebben a fázisban sok fejfájástól kímélhet meg a későbbiekben. Mire valóban nekiláthatsz a vizuális terveknek, addigra már jól fogod ismerni a site felépítését, tartalmát, és ez sokkal könnyebbé teszi a munkát a színekkel és grafikákkal is.

2.4.3 Az elképzelt példa oldal

Az elképzelt oldalunk valójában egy üzlet, amely nyílt forráskódú wikiket szolgáltat, és legalább három új kódolási ötlettel jönnek elő hetente. Mivel a kódot ingyenesen lehet használni és módosítani, így a site gazdája úgy gondolta, hogy a bevételeket adományokból, reklámokból és a programozóik által nyújtott extra szolgáltatásokon keresztül próbálja megszerezni. A site neve „Wiki Whatevers”, és a domainnevet már kiválasztották. A tartalmat már elkészítették, amiben főleg kódrészletek találhatóak, amelyeket kategorizálni kell, valamint vannak még különböző leírások, és néhány biográfia a projektben résztvevő programozókról. A webszolgáltatónál van MySQL adatbázis hozzáférhetőség, és képes nagyobb forgalmat is fogadni leállás nélkül. Most már ideje összeszedni azokat a dolgokat, amelyeket a használni fogunk.

1. A már létező céges logó használatához szükségem lesz a logó digitális változatára, amelyet a kliens weboldalán használhatok. A logót egy scanner segítségével beolvasom egy grafikus alkalmazásba, mint amilyen a Photoshop vagy a Gimp. A logó méretét csak később igazítom a website méretéhez. A képet 72 dpi-ben mentem

el, ami gyorsabb letöltési időt tesz lehetővé. Ezt a logót használom majd a 4. lépésben.

2. A biográfiákban és a csapat bemutatásánál (a “Névjegy" lapon) szükségem lesz néhány digitális képre a programozókról. Küldhetnek digitális képeket, vagy hagyományos képeket scannelésre. Ha hagyományos képet küldenek, akkor nagyobb méretben scannelem be, mint szükséges, úgy 300 dpi elég szokott lenni. A digitális képeket is teljes méretben őrzöm meg, így a képeket később a körülményeknek megfelelően kicsinyíthetem.
3. Az ügyfél úgy döntött, hogy szeretnének indítani egy blogot is, mivel már elegendő anyaguk van ahhoz, hogy a blogot néhány hónapig aktívan tartsák. Szerencsére olyan szolgáltatót választottak, amelyik támogatja a blogokat, és megfelel a szükséges feltételeknek, így képes adatbázisokat és megnövekedett vagy ingadozó forgalmat is kezelni. A szolgáltató több lehetőséget is ad a későbbi fejlesztéshez, ami nagyon előnyös, ha a kliens később egy nagyobb weboldalt szeretne. Ha a szolgáltatás rendelkezésre állási ideje is garantált, akkor az ügyfelünk különösen boldog lehet, mivel a website fejlődésével továbbra is maradhat ugyanannál a szolgáltatónál évekig, így megspórolhatja a szolgáltatóváltással járó macerákat.
4. FTP használatával (ilyenből több is van a piacon, mint például a nyílt forráskódú Filezilla, a fireftp Firefox kiterjesztés, vagy különálló kliensként a CuteFTP) feltöltök egy statikus oldalt, amely bejelenti a site indulását. Kerüljük a „fejlesztés alatt” és az ehhez hasonló borzalmas kifejezéseket, mivel a látogatók valószínűleg úgysem térnek vissza soha, ha nem tudják, hogy mikor indul a weboldal. Ehelyett írjuk ki a lapon a cég nevét, hogy mit fognak ezen a weblapon nyújtani, egy dátumot, amikor a weblap várhatóan elindul és egy kapcsolati lehetőséget (az email tökéletes ebben az esetben, de ha pl. egy téglagyárnak készítünk oldalt, akkor megadhatunk egy címet és egy telefonszámot is). Még ennél is jobb, ha a látogatók fel tudnak iratkozni az email címükkel, hogy kapjanak egy értesítést az indulásról, ezzel az ügyfelünk már azelőtt szerezhet néhány potenciális vásárlót, mielőtt még a weblap egyáltalán elindult volna.
5. Az ügyféltől kapott tartalom és a struktúra információk alapján elkészítem a site felépítését, valamint a navigációt és a szöveges linkeket. Nem szabad megfeledkezni arról sem, hogy a kliens minden oldalon szeretne reklámokat elhelyezni. Ezen kívül megpróbálok egy tervet készíteni a site SEO kulcsszavaira is.
6. A logó színeinek felhasználásával kiválasztok két vagy három színsémát, amit megmutathatok később a kliensnek.
7. Ezután kiválasztok még néhány fényképet és illusztrációt valamilyen képgyűjteményből, mint amilyen például az iStock vagy a Comstock. Mivel az ilyen képgyűjtemények általában forgalmazzák a képeket, így a képeket a legtöbb esetben meg kell venned, és nem kerülheted ki a vásárlást. A képgyűjtemények használata egyáltalán nem drága dolog, és sok fejfájástól kímélhet meg később licencelési ügyekben. Ezen felül szükségem van azokra a képekre is, amelyeket a cég korábban már felhasznált, vagy fel akar használni, amelyeket a kódoknál, a leírásokban és a blogbejegyzéseknél lehet majd felhasználni.

Megjegyzés: az utolsó két lépést a következő leírásban részletesebben is kifejtjük. Ne felejtsd el azt sem, hogy még mielőtt elkezdenél színeket és grafikákat használni a weblapon, előtte mindenképpen egyeztesd a vizuális terveket a klienssel, és kérd a beleegyezését!

2.4.4 A logó

A logó a cég márkajegyeinek egyik legfontosabb eleme. A legtöbb cég nem sieti a logó elkészítését, mivel ez a műalkotás fogja képviselni a cégüket még több éven keresztül. Más cégeket viszont egyáltalán nem érdekel az a kis kép, amely a cégüket jelképezi. Tapasztalatból mondhatom, hogy az a cég, amely nem fektet elég pénzt és energiát egy professzionális logó készítésébe, soha nem fogja elkölteni ezt a pénzt, nem számít, hány logikus érvet tudsz felhozni velük szemben.

A Wiki Whatevers cég tulajdonosai mind a Georgia Tech hallgatói, így az Alma Materük színeit használták fel a logójukban: aranyat és feketét. Mivel a logó nagyon egyszerű, így könnyen lehet rajta dolgozni a színeken és a megjelenésen. A 2.24. ábrán látható a logó:

2.24. ábra: A Wiki Whatevers logója

A probléma itt az, hogy éppen most scanneltem be ezt a logót, és szeretném online használni. Viszont a nyomtatási színek, amelyek CMYK (kékeszöld, bíbor, sárga, fekete) alapúak, nem egyeznek meg a webes színekkel, amelyek RGB (vörös, zöld, kék) alapúak. Így először meg kell feleltessem a színeket a webes színeknek, amilyen közel csak lehetséges. Erre több lehetőség is van:

1. Kapcsolatba lépünk a nyomdásszal, hogy milyen színeket használt a Wiki Whatevers logó nyomtatására a legutóbbi nyomtatott anyagnál. A nyomdászok általában Pantone színeket használnak, és a Pantone szerencsére biztosít egy olyan eszközt, amivel meg lehet feleltetni a nyomtatott színeket a webes színekkel. Valószínűleg a nyomdásznak is van ilyen eszköze kéznél, úgyhogy akár anélkül is megkaphatjuk a kívánt webes színeket, hogy fizetni kellene egy ilyen eszközért.
2. Mivel a srácok a Wiki Whateversnél valójában a Georgia Tech színeit használták, egyszerűen elmehetek a Georgia Tech oldalára, és megpróbálhatom ott egyeztetni a webes színeket. Egy grafikus program használatával megszerezheted azt a színt, amelyet a weblapon használtak, ha készítesz egy képernyőképet a weblapról, azt betöltöd a grafikus programba, és a pipettával vagy valamilyen más módon megfeleltetheted a színeket.
3. Hasonlítsd össze figyelmesen a nyomtatott színeket a webes színekkel, és próbáld meg a lehető legközelebbi módon megfeleltetni őket. Egyes esetekben a színek teljesen különbözhetnek, míg máskor talán nem is szükséges rajtuk változtatni.
4. Scanneld be a nyomtatott képet egy olyan scannelő alkalmazással, amely elfogad CMYK-t, majd használd a Photoshop Pantone Colour Swatches eszközét a színek megfeleltetéséhez. Ez a módszer csak akkor működik, ha van egy olyan scannered, amelyik elfogadja a CMYK-t, és van Photoshop szoftvered is.

A mi esetünkben sikerült megszereznem azt a tökéletes arany színt a Georgia Tech Athletic site kabalafigurájáról, amely teljes mértékben megfelel a logó színének. Az arany

#eab200, míg a fekete, hát, egyszerűen fekete (*#000000*). A háttér, ami sötét kékeszöld színű (*#002123*) árnyékként van felhasználva a logóban. Szóval a problematikus részeket egy kis méhecskefigura segítségével (a 2. ábrán) könnyen megoldottuk:

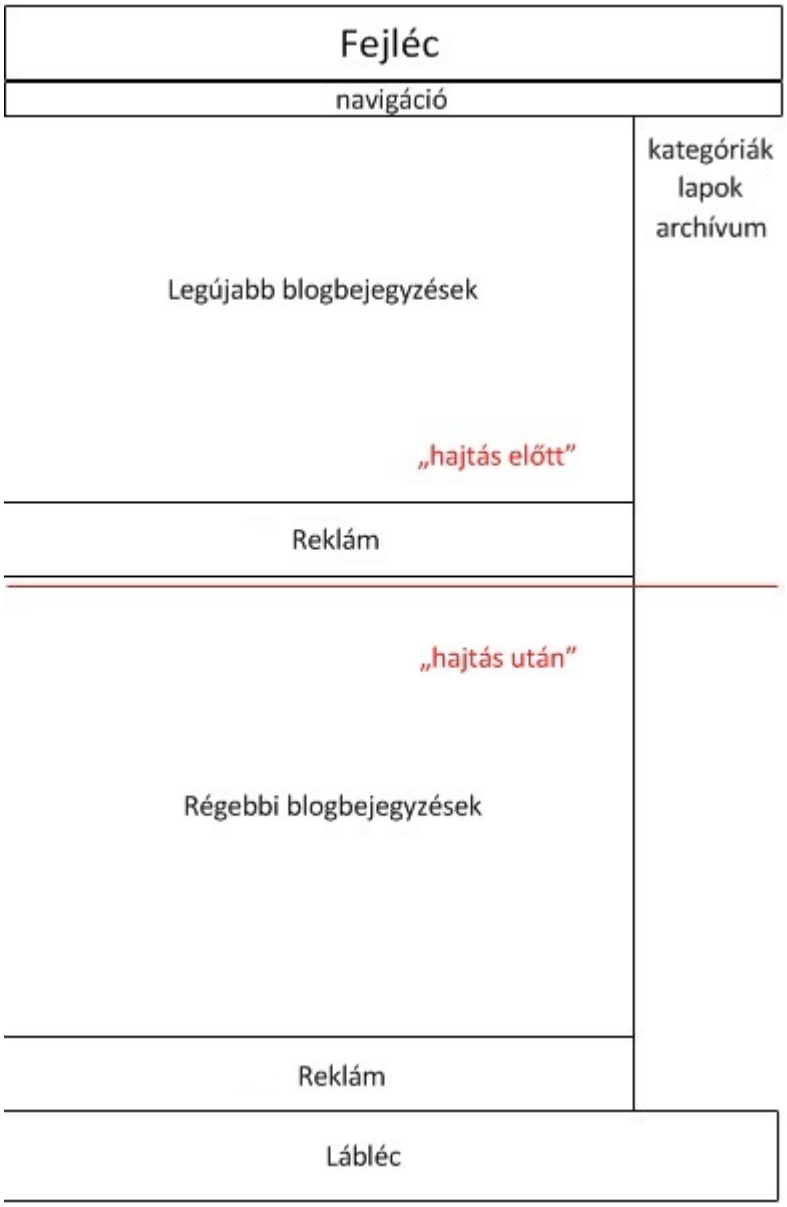
2.25. ábra: A Georgia Tech kabalaállatának színeit feleltettük meg a logó színeivel

Megjegyzés: nagyon ritkán fordul az elő, hogy egy cégnek nincs meg a saját logójuk vagy más márkajelük digitális változata valamilyen formában, például elektronikus névjegy vagy levélfejléc képében. Viszont nem árt tudni, hogy az ilyen cégek túlnyomó része egyszerűen elfogadja a színeket, ahogy azok megjelennek, és nem egyeztetik a színeket a nyomtatott anyagokkal. Szóval ne bízzunk meg mindig azokban a színekben, amelyek már rendelkezésre állnak, főleg akkor nem, ha a színek láthatóan nem egyeznek meg a brosúrákban vagy levélfejlécekben látható színekkel. Ilyenkor kérdezzük meg a megrendelőket, hogy melyik színt preferálják jobban. Lehetséges, hogy eddig észre sem vették, hogy a színek valójában különböznek.

2.4.5 Az elrendezés

Az egyszerűség kedvéért (és hogy ne legyen túl hosszú a mai anyagunk) egyetlen elrendezést fogok bemutatni. Egy blogos elrendezést választottam, amelyik a gyakoribb változásokat a törzs felső részén jeleníti meg, könnyen navigálhatunk a fejléc és a törzsszöveg között, valamint hozzáférhetünk a korábbi bejegyzésekhez „hajtás után” a kezdőlapon. A „hajtás után” kifejezés a nyomtatott sajtótól származik, mivel amikor egy újság kint van a pulton, az olvasó csak azt látja, ami a „hajtás előtt” van (vagyis az újság első lapjának felső felét, ahogy össze van hajtva). Ezt a rész - beleértve a képeket is - rendkívül fontos, mivel rá kell vegye az olvasót, hogy megvásárolja az újságot.

Ugyanez a „hajtás előtt” filozófia érvényes a websiteok designjára is. Minden, ami a képernyőn látszik, amikor egy látogató megnyitja az oldalt, az a „hajtás előtti” rész. Minden olyan rész, amelyért a látogatónak görgetnie kell, már a „hajtás utáni” rész. A trükk az, hogy a látogató tekintetét már az első képekkel és szövegekkel megfogjuk, függetlenül attól, hogy milyen monitoron és milyen felbontásban nyitja meg a weblapot (ezért jó mindig letesztelni a websiteot több különböző monitoron és felbontásban is, erről később fogunk még beszélni). A 2.26. ábrán a Wiki Whatevers kezdeti elrendezésének első vázlatát láthatjuk:



2.26. ábra: A Wiki Whatevers honlap elrendezésének első vázlata

Ez az elrendezés változatlan marad az egész oldalon, de változhat az archívum lapokon, hogy megjelenítsük a cikkek és blogbejegyzések listáját képek nélkül. A következetesség azért jó, hogy ne zavarjuk össze a látogatót. Ha a felhasználók „megtanulják” a site használatát, akkor már nem szeretik, ha az elrendezés lapról lapra változik. Az alábbiakban bemutatjuk, hogy mit tartalmaz ez a design:

1. **Fejléc**: a fejléc kicsi lesz, mivel nem szeretném, ha a logó túl sok helyet elfoglalna a lapokon. Bár a logó csak egy apró része az oldalnak, viszont a színei meghatározzák a teljes website színsémáját. A fejléc felül található, szokás szerint, és a logó a blog kezdőlapjára mutat. A logó linkelése kényelmes megoldás, és a legtöbb látogató már hozzászokott ahhoz, hogy a logóra kattintva mindig visszajut az adott website kezdőlapjára.
2. **Navigáció**: közvetlenül a fejléc alá helyezve a navigáció nyilvánvaló és könnyen használható. A navigációt a láblécben is megismételjük egyszerű szövegként. Ezt azért teszem, mert én még a régi iskolából származom, ahol a navigációt mindig megismételtük a láblécben sima szövegben azoknak a kedvéért, akik képek nélkül

böngésznek. Mivel ezen a ponton még nem tudom biztosan, hogy fogok-e képeket használni a navigációra a fejlécnél, így automatikusan beteszem a szöveges navigációt valahová máshová a lapon, jelen esetben a láblécbe. Ez segíti azokat a vak látogatókat, akik képernyő-felolvasókat használnak a weblapok „olvasására". Az, hogy a navigációs szövegeket felülre vagy alulra helyezed, valójában a design szempontokat leszámítva lényegtelen, mivel a vakok ugyanúgy át tudják fésülni a weblapot fentről le és lentről fel, mint a látók. Ezek függvényében igazából a designeren és a megrendelőn múlik, hogy megismétlik-e a navigációt a lapon vagy nem. Ha a navigációnál képeket használsz, és nem ismétled meg a navigációt szöveges linkekben, akkor figyelj arra, hogy a navigációs képeknél mindig megadj egy beszédes alt attribútumot. Ezen a módon a képernyő-felolvasók vagy a kikapcsolt képekkel böngészők is tudni fogják, hogy az adott képek mire valók. Olvasd el a 17. leírás idevágó részét az alt attribútum megfelelő használatáról.

3. **Legújabb blogbejegyzések**: a legújabb blogbejegyzéseket érdemes kiemelni, és az a lehetőség, hogy ezt lap fókuszába tehetjük a „hajtás elé", előnyös mind az ügyfélnek, mind az olvasóknak. Amint a látogató megnyitja a weboldalt, ezt a szöveget fogja látni először. A nyilvánvaló elhelyezés viszont azt is jelenti, hogy az ügyfél kénytelen rendszeresen frissíteni az oldalt, ellenkező esetben a már meglévő látogatók elvesztését kockáztatja: az emberek nem szeretnek visszatérni egy blogra, ha nem kerül rá új tartalom.
4. **Régebbi blogbejegyzések**: itt lesznek majd a korábbi blogbejegyzések; úgy három vagy öt bejegyzés bőven elég, hogy a látogató kapjon egy rövid benyomást arról, hogy mire számíthat ezen az oldalon a továbbiakban. Ha vannak képek, az nagyszerű, de nem feltétlenül szükségesek, mivel ez a terület már a „hajtás után" található. A döntés, hogy használunk-e képeket függhet attól, hogy a lap letöltési mérete és ideje mennyire számít, valamint hogy az előző bejegyzéseknél van-e szükség képre, amelyre kattintva a látogató tovább olvashatja az adott témát.
5. **Jobb oszlop**: ezen a területen férnek hozzá a látogatók a blogbejegyzésekhez a kategóriákban, az archívumhoz és a site más tartalmaihoz. A többi tartalomra egy példa lehet a cég „névjegy" lapja, egy site index oldal, valamint kapcsolati információk. Fontos eldönteni azt is, hogy hogyan akarod megjeleníteni ezeket az elemeket az oszlopon belül, mivel a blog a létrehozott kategóriákból, lapokból és az archívumból fog felépülni. Ahogy a site növekszik, ezek a listák egyre nagyobbak lesznek, míg végül elérkezhet az a pont, hogy csak a kategóriák listája lesz egyedül a „hajtás előtt". Az ügyfél dönthet arról, hogy a lapok fontosabbak-e, mint a kategóriák. Megemlíteném még, hogy ez a lista nem tartalmaz mindent, amit az oldalsávba vagy az oldalsó oszlopba el lehet helyezni. Egyes ügyfelek úgy érzik, hogy két oszlop jobban megfelelne nekik, így háromoszlopos elrendezést kérnek a fenti kétoszlopos helyett.
6. **Lábléc információk**: a lábléc információk alapvetően fontosak, mivel ezek látják el a látogatókat a lényeges háttérinformációkkal a cégről és a websiteról, amelyet egyébként nehezen tudnának előkeresni. A cég neve, esetleg a logó ismétlése, egy cím, email cím, hivatkozások (a kapcsolat oldalra, adatvédelmi megjegyzésekre, figyelmeztetésekre és jogi tudnivalókra), rövid hírösszefoglalók szerepelnek leggyakrabban a láblécekben. Ahogy azt már korábban is megjegyeztük, a navigációt is megismételhetjük szöveges változatban a láblécben.
7. **Reklámok**: ebben az elrendezésben a reklámok a legújabb blogbejegyzés, valamint a régebbi blogbejegyzések mögött helyezkednek el. Ez lehetővé teszi a kliensnek, hogy válasszon a szöveges reklámok vagy a bannerek között. Ebben az elrendezésben egy reklám van a „hajtás előtt", és ugyancsak egy reklám a „hajtás

után”. A legtöbb website esetében ez a reklámmennyiség elegendő. Ebben az esetben a reklámok egy másodlagos helyre vannak száműzve, a fő tartalom mögé.

Ez az elrendezés lehetővé teszi a látogatónak, hogy gyorsan átválthasson a tartalomról a navigációra görgetés nélkül, valamint láthatóvá teheti a felhasználónak a site további témáit is a kategóriák alatti hivatkozásokon keresztül. Még ha az olvasó le is görget a „hajtás után”, az elrendezés tartalmazza a „hajtás előtt” az összes olyan fontosabb elemet, amire szüksége lehet.

2.4.6 A reklámozásról az oldalakon

Ha az oldalra tartalomfüggő reklámokat teszünk, az egyrészt előny az ügyfélnek és egy hasznos szolgáltatás lehet a látogatónak is. Ha például az oldal virágokról szól, akkor a kapcsolódó reklámok tartalmazhatnak parkosítási szolgáltatást, virágrendelést, stb. Egy olyan oldalra, amely nyílt forráskódú szoftverekkel foglalkozik, olyan reklámozókat kereshetünk, amelyek kapcsolódnak a nyílt forráskódú tartalmakhoz. A Google Adsense egy lehetséges megoldás erre, mivel segíthet a tartalomhoz kapcsolódó reklámokat helyezni az oldalra. Az ilyen típusú reklám jó megoldás egészen addig, amíg a forgalom meg nem nő a megfelelő szintre, amikor más típusú reklámokat is el lehet már helyezni a weblapon. A reklámok használatakor mindig gondolj ezek SEO vonzatára is, mivel a reklámok jelentősen befolyásolhatják a weboldal pozícióját a keresőkben.

Megjegyzés: mint designer, valószínűleg nem te vagy a felelős a reklámozásért az oldalon, kivéve ha saját magadnak készíted az oldalt. De ha valamikor később szeretnél egy hirdetési- vagy design ügynökségnél elhelyezkedni, tudnod kell egyet-mást a reklámozásról is. Minél többet tudsz arról, hogy mi tesz egy websiteot sikeressé, annál több sikerre számíthatsz te is a designer karrieredben. Amikor lehetséges, próbálj minél többet megtudni a marketingről (a saját és az ügyfeleid érdekében), valamint a keresőoptimalizálási trükkökről.

2.4.7 Az elrendezés ellenőrzése validátorral és a klienssel

Mielőtt az elrendezést elkezdeném a kódban megvalósítani, még szeretném megmutatni a kliensnek és a beleegyezését kérni a használatához. Az egyik taktika, amivel megpróbálom rábeszélni az ügyfelet, hogy ez az elrendezés jobb, mint egy másik, hogy emlékeztetem arra, hogy a további elrendezések tervezése pénzbe kerül. Így az ügyfél általában már kiválaszt egy elrendezést, mivel ezeken is lehet még változtatni később, ha szükség lenne rá.

A következő lépés az elrendezés lekódolása, majd a kód validálása. Én a W3C jelölés validáló szolgáltatását, valamint a W3C CSS validáló szolgáltatását használom a HTML és a CSS kódok ellenőrzésére, hogy ne maradjanak bennük hibák. A fájlokat feltöltheted egyenesen a számítógépedről, így nem szükséges emiatt feltölteni őket a kliens websitejára csak a tesztelés miatt. Ezek a tesztek lehetővé teszik a designernek és/vagy a programozónak, hogy már azelőtt kijavíthassák a jelölési hibákat, mielőtt a kód megtelne képekkel, reklámokkal és más elemekkel.

A hozzáférhetőség ugyancsak nagyon fontos, így érdemes meggyőződnöd arról is, hogy a website használható a vak, gyengénlátó vagy mozgássérült emberek által is. Ez nem olyan egyszerű, mint a CSS és a HTML validálása. Vannak ellenőrző eszközök, mint például a TAWDIS, de az ideális az, ha valódi felhasználókkal teszteltetted le a honlapot, és elvégzel egy minőségi ellenőrzést a hozzáférhetőségre, ugyanis egy mechanikus ellenőrző eszköz nem képes megmondani azt, hogy a website hozzáférhető, avagy sem, csak né-

hány javaslatot tehet arról, hogy mi a jó és mi a rossz ebből a szempontból, és gyakran hibáznak. A kurzus során még sokszor lesz szó a hozzáférhetőségről, úgyhogy ne felejtsd el ezt a pontot.

Ugyancsak érdemes végigtesztelned az elrendezést az összes rendelkezésre álló böngészőben, hogy biztos lehess benne, hogy a lehető legtöbb felhasználót elérheted. Ezt könnyen megteheted akkor, ha van Maced, Windowsod és Linuxod és megfelelő mobilod az összes szükséges böngészővel telepítve, de használhatsz emulátorokat, mint például a VMWare Fusion, amellyel egyetlen számítógépen emulálhatod a különböző rendszereket, de ezek használata elég bonyolult és hosszadalmas. Egy másik módszer a böngésző-képmentő használata, mint például a BrowserCam, amelyik egy gyors, kényelmes szolgáltatás, és rengeteg böngészőt támogat (sok régi böngészőt is). Az első 24 órában ingyenesen lehet használni, így nyugodtan kipróbálhatod, hogy megfelel-e neked, utána pedig az ára bőven megéri azt a kényelmet, amelyet biztosít a különböző böngészőkben való tesztelésre.

Végezetül, ismét jó ötlet, ha az ügyféllel tudatod, hogy a kód az elrendezéshez elkészült és sikeresen validálva lett; azt is megmondhatod neki, hogy milyen módosításokat kellett végezni rajta ahhoz, hogy működjön minden böngészőben. Miután a kódot legeneráltad, a validálást elvégezted, és megkaptad az engedélyt a klienstől, csak ezután állhatsz neki a színek, képek és más tartalmak, mint például a reklámok hozzáadásának. Bár ez így fárasztónak tűnik, jobb, ha már az elején elvégzed a validációt és egyeztetsz a klinessel, mielőtt feltennéd a cukormázat a süteményre. Különben azon kaphatod magad, hogy többet foglalkozol a problémák keresésével és több bajod van a különböző böngészőkkel, mint azt korábban gondoltad. Másrészről bármilyen vizuális elem vagy kész tartalom elvonhatja a kliens figyelmét az elrendezésről, amikor be szeretnéd neki mutatni.

Miután befejezted ezt a lépést, nekiállhatsz a munkának a website szövegeivel, képeivel és színeivel. Hogyan kezdhetsz neki ennek? A következő leírásunkból megtudhatod!

2.4.8 Tesztkérdések

- Mikre van szükséged, mielőtt elkezdenéd egy weblap designjának a fejlesztését?
- Miért kell összeszedned azokat az elemeket, amikre szükséged lesz a weblapon?
- Miért fontos utánanézni a webszolgátatóknak?
- A designernek sok dolga van, de mit kell tenned akkor, ha a kliens azt kéri tőled, hogy tervezz neki egy logót, viszont még soha nem csináltál ilyet?
- Nevezz meg két jó okot, hogy miért érdemes átnézni a konkurencia weboldalait.
- Mi az a CMYK, és mit jelent?
- Nevezz meg legalább két módot arra, hogy a CMYK színeket RGB színekké alakítsd.
- Mondj egy okot arra, hogy miért használjon egy webdesigner szöveges navigációt legalább egy helyen a weblap elrendezésében?
- Miért kell az elrendezés konzisztens maradjon az egész websiteon?
- Mondj egy okot arra, hogy miért kell az elrendezést már az egészen korai fázisban validálni.

2.5. Színsémák és designtervek

Miután a webdesigner bemutatja a kliensnek a site vázlatos szerkezetét, felépítését, a következő lépés a színek és grafikák segítségével elkészíteni az oldal megjelenését. Ebben a leírásban bemutatjuk, hogyan lehet ez egyszerűen elvégezni mind a kliens, mind a saját részünkről. Én az egyszerűségben hiszek, aminek két fő oka van: először is, az élet éppen elég bonyolult anélkül is, hogy külön kanyarokat adnánk hozzá, másodszor pedig egy egyszerű terv jobban segít az oldal hozzáférhetőségének és használhatóságának megtartásában.

2.5.1 Első lépés: néhány szó a tipográfiáról

A tipográfiáról részletesebben a következő leírásban olvashatsz majd, most csak néhány szempontot emelnék ki, amelyek hasznosak lehetnek ebben a fázisban.

A betűtípusokat, vagy más néven „betűképeket” a szövegek, számok, karakterek és más szimbólumok megjelenítésére használjuk. Ezek a karakterek, szimbólumon, betűk és számok különböző szempontok alapján vannak kategorizálva, mint például családok (kapcsolódás alapján), stílus (dőlt, normál, rézsútos, stb.), változat (normál vagy kiskapitális), vastagság, nyújtás (szélesség és magasság összenyomása vagy széthúzása) vagy méret (szélesség és magasság pontokban vagy pixelekben). A tipográfia a szöveg elrendezése és megjelenése, így a tipográfia arra vonatkozik, hogy hol és hogyan jelennek meg a karakterek a lapon (oszlopok, paragrafusok, igazítás és egyebek). A lehető legjobb módja annak, hogy befolyásoljuk egy oldal tipográfiáját, az a CSS használata.

A webdesign elkészítésének egyik utolsó lépése az, hogy eldöntjük, milyen betűtípusokat használunk a website különböző részein. Több tanulmány is született már arról, hogy a túl sok betűtípus egy oldalon belül összezavarja a felhasználókat. Másrészről, az olyan weboldal, amelyik csak egyetlen betűtípust használ, unalmasnak tűnhet.

Az én tanácsom az, hogy használjunk egy betűtípust a címsoroknak és az alcímeknek, és egy másik betűtípust a szövegtörzshöz, főleg akkor, ha reklámozni is szeretnél az oldalon. Egy betűtípus a címeknek és egy betűtípus a szövegeknek a teljes websiteon folyamatosságot biztosít a lapok között, és elég változatosságot teremt ahhoz, hogy az olvasó gyorsan megkülönböztethesse mindenhol a címeket a tartalomtól. A reklámok további változatosságot hoznak be, mivel nem tudhatjuk, hogy a reklámozó milyen betűtípusokat használ a megjelenített reklámokban.

Én többnyire Verdana betűtípust használok a szövegtörzsben, és Times New Roman vagy Georgia betűtípust a címsorokban, és ezekhez a teljes designer karrierem alatt tartottam magam. A Times New Roman és a Georgia talpas (serif) betűtípusok, míg a Verdana talpatlan (sans-serif). Hadd magyarázzam meg a különbséget a kettő között, és hogy miért maradtam végig ennél az egyszerű választásnál…

Design okokból viszont előfordulhat az is, hogy egyetlen betűtípust szeretnél használni az egész oldalon, vagy csak két (és nem több) típust. Ahogy rövidesen látni fogjuk, a példa oldalunkon maradtam a Verdananál a szövegtörzs esetében, de mivel a logó Arial Black betűtípussal készült, így végül ugyanezt a talpatlan betűtípust használtam a címsorokban is. Néha meg kell szegned a saját szabályaidat is, és ez az elrendezés remek példa erre az eshetőségre.

Betűképek, betűtípusok

A betűtípusoknak négy nagy csoportja van, mégpedig a következők:

Talpas vagy serif: minden olyan betűtípus, amely egy befejezett vonást, szélesedő vagy vékonyodó vonalvégeket tartalmaz, vagy talpas végei vannak (beleértve a lapos talpakat is), az ebbe a családba tartozik. Korábban a talpas típusokat főleg nyomtatásnál használták, mivel ezeket könnyebb egy nyomtatott szövegben elolvasni. Viszont a web nem olyan, mint a nyomtatott sajtó, így aztán több tanulmány készítése után kiderült, hogy itt bizonyos talpatlan betűtípusokat, mint például a 2. ábrán láthatót, a weblapok törzsében használva könnyebben lehet elolvasni.

Times New Roman, normál, 18 pontos

Talpatlan vagy sans-serif: minden olyan betűtípus, amelyben a vonásoknak egyszerű, sima végeik vannak, nem vékonyodnak vagy vastagodnak el, nincsenek benne talpak vagy más díszítések, azok ebbe a családba tartoznak. Bár egyes szerzők azt állítják, hogy az olvashatósági tanulmányok egy lyukas garast sem érnek, mégis azt vettem észre, hogy ezeken az oldalakon is gyakran talpatlan betűtípust használnak a szövegtörzsnek. Sőt, még azok az oldalak is talpatlan típust használnak a szövegtörzs betűtípusának, amelyek azt állítják, hogy a talpas típus jobban olvasható. Szóval én ebben az esetben egyszerűen követem a tömeget, és talpatlan betűtípust használok, mint amilyen a 2. ábrán látható betűtípus, amely mára már a web hagyományos szövegtörzs betűtípusa.

Verdana, normál, 18 pontos

Írott vagy kurzív: ezek a betűtípusok általában egy kézzel írott, tollal vagy ecsettel készített betűkre hasonlítanak, és nem a nyomtatott betűkre, ahogyan az a 3. ábrán látható. Ebbe a családba beletartoznak a kézírásra hasonlító betűtípusok, még akkor is, ha egyébként nem kurzívak. Mivel a hosszabb szövegrészek az ilyen betűtípussal írva általában nehezen olvashatóak, így nem ajánlott ilyen típusú betűkészletet használni a szövegtörzsben (elég csak arra gondolnod, hogy milyen nehéz volt végigolvasni Margit nagynéni kézzel írott leveleit, vagy azt a 12. századi kéziratot, amit egyszer a múzeumban láttál). Ráadásul nem minden böngésző fogja ugyanazt a betűtípust megjeleníteni, úgyhogy ha te egy írott vagy kurzív típust választasz, egyes böngészők dönthetnek úgy, hogy mégis egy talpas betűtípussal jelenítik meg a szövegedet.

Brush Script, normál, 18 pontos

Különleges, valamint fix szélességű: a fix szélességű betűtípusok jellemzője, hogy minden karakternek pontosan ugyanakkora, rögzített szélessége van, hasonlóan egy írógéppel gépelt oldalhoz. Más fontoknak fantasy vonzatai vannak, mint például a 4. ábrán látható betűtípus. Az ilyen betűkészleteket általában díszítő elemként használják. A fix szélességű típusoknak is megvan a helye a weblapokon, például programkódok megjelenítésére. Ezt a típusú betűkészletet gyakran használják ilyen célokra, mivel így jól láthatóak az egyes karakterek és szimbólumok a kódban.

Jokerman LET,normál, 18 pontos

A betűk képeit megnézve feltűnhet, hogy nem egyforma méretűek, pedig mindegyik ugyanakkora pontmérettel készült. A pontméret egyetlen betű magasságát jelöli, és egyes betűkészletek nagyobbak lesznek 18 pontos méreten, mint mások. Más különbségek is vannak köztük, mint például a távolság a betűk és a szavak között, vagy az a tény, hogy mondjuk a Jokewood típusban nincsenek kisbetűk. Látható az is, hogy a Jokewood, valamint a Staccato írás nehezen lenne olvasható egy szövegtörzsben. Az ilyen betűtípusok is megtalálják a maguk helyét, kisméretű, címsor jellegű területeken vagy a reklámokban.

Érdemes megemlíteni még egy pontot, mégpedig hogy elfordulhat az, hogy ezek a betűtípusok nem fognak egyformán megjelenni a különböző böngészőkben, mivel ezek nem teljesen kompatibilisek egymással. Ezt egyszerűen értsd úgy, hogy nem minden böngésző jeleníti meg ugyanazokat a betűtípusokat. Ennek az az oka, hogy nem minden operációs rendszer támogatja a különböző betűtípusokat. Vagy talán támogatják ugyanazt a fontot, de már a variációi, a vastagsága vagy más tulajdonsága böngészőnként eltérhet. Emiatt ajánlott inkább általános, vagy csak egyszerűen „serif" vagy „sans-serif" betűtípusokat használni a weblap tipográfiájának megjelenítésére. Vagy választhatsz egy általános nevet és mellé annak a fontnak a nevét, amelyet választottál, aztán reméled a legjobbakat, mivel a felhasználóknak egyes esetekben lehetőségük van arra is, hogy átírják a betűtípust vagy annak megjelenését.

Az egyetlen mód arra, hogy egy bizonyos betűképet használj megadott stílussal, változattal, mérettel és vastagsággal, minden böngészőben egyformán, hogy készítesz egy képet ezzel a betűtípussal egy grafikus alkalmazásban. Ez viszont több okból sem ajánlott, egyrészt mivel elrejti magát a szöveget (a képernyő-felolvasók nem tudják elolvasni a képekbe írt szövegeket), valamint nehéz karbantartani (minden alkalommal, ha át akarod írni a szöveget, újra el kell készítened a képet). Higgy nekem, nem éri meg.

Ezek az okai annak, hogy mint webdesigner, néha el kell felejtened azt, hogy a web valójában egy rendkívül rugalmas formátum. Rengeteg dolog felett van befolyásod egy website designjában, ennek egyik eleme a tipográfia, de ez csak akkor hatékony, ha a lehető legegyszerűbben kezeled. Emiatt maradtam én hosszú évekig a szövegtörzseknél a Verdana, a címsoroknál pedig a Times New Roman vagy a Georgia mellett.

Eközben azért a betűkészletek készítői és a programozók folyamatosan dolgoznak azon, hogy szebb és olvashatóbb típusokat készítsenek. Szóval, adj egy csipet sót ahhoz, amit korábban mondtam, és ha úgy érzed, hogy működhet, próbálj ki valami újat. Hamar észre fogod venni, ha nem válik be, amint elkezded tesztelni a kódot a különböző böngészőkben (erről kicsit később fogunk beszélni ebben a leírásban).

Olvashatóság

Amikor megmutatod a weboldal designját a kliensnek, valószínűleg nem fogja tudni, hogy mi a különbség a talpas és a talpatlan betűképek között. Amit tud, az csak annyi, hogy el tudja-e könnyen olvasni a lapot, vagy nem. Így végeredményben egyedül az olvashatóság számít. Emiatt meg kell győződnöd az alábbiakról:

1. **A választott betűtípus elég nagy ahhoz, hogy több felbontásban is olvasható maradjon.** Bár a felhasználóknak általában van lehetőségük arra, hogy megváltoztassák a betűméreteket (például Operában), meg kell próbálnod úgy beállítani a méreteket, hogy azok olvashatóak maradjanak a különböző böngésző felbontásokban. Erre egy jó módszer, ha a pixelméretek helyett százalékokat vagy em méreteket használsz (ezt CSS-sel megteheted).
2. **Elegendő kontraszt van a háttér és a szövegtörzs között**. Egy fehér szöveg fekete háttéren hosszabb szövegek esetén eléggé fárasztó a szemnek, viszont ha mindenképpen ezt a vonalat akarod követni, biztosíts alternatív stíluslapokat, hogy a felhasználók átválthassanak sötét szövegre világos háttérrel, ha szeretnének.
3. **A címsorok valóban különböznek a szövegtörzstől**. A lehetőség, hogy a szövegedet feloszthatod több részre címsorokkal és alcímekkel, vagy például listákkal (mint amilyen ez is), egyszerűbbé teszi a felhasználóknak, hogy gyorsan átnézhessék a weblapot, és kiszűrjék belőle a fontos részeket. Ha képekkel osztod

fel a szöveget, az is rendben van, viszont akkor a képeknek relevánsaknak kell lenniük, ellenkező esetben csak a sávszélességet vesztegeted.

4. **Kerüld a szövegtörzs sorkizárt megjelenítését a teljes képernyő szélességében egy rugalmas elrendezés mellett**. Próbálj meg elolvasni egy olyan bekezdést egy széles képernyőn, amely kihasználja a képernyő teljes szélességét. Hamar kifáraszthatod magad, mivel a szemeid és a fejed folyamatosan ide-oda mozog, ahogy a szöveget olvasod, a képernyő egyik szélétől a másikig. Nézd meg az 5. ábrán látható weboldalt az olvashatóság szempontjából, amely nagyszerű példa egy olyan szélességre, amely olvasható marad különböző felbontásokon is. Ezen a képen láthatod azt is, hogyan olvassák az emberek a lapokat, függetlenül attól, hogy az webes vagy nyomtatott. A képet egy 24 colos képernyőn készítettem 1920 x 1200 felbontásban. Hasonlítsd össze ezt a képet azzal, amit a saját képernyődön látsz, ha a fenti linkre kattintasz. Nézd meg a saját képernyőd felbontását, hogy láthasd a különbséget. Néha a fix elrendezés nagyon hasznos, mivel szabályozza a szövegtörzs méretét, így az könnyen olvasható marad a látogatóknak. Ne aggódj amiatt, hogy a lap körül rengeteg kihasználatlan terület marad (mint a lenti képen is). Használj egy jó hátteret, ami nem vonja el a figyelmet a designról vagy a tartalomról, így gondolhatsz azokra is, akiknek széles képernyőjük van.

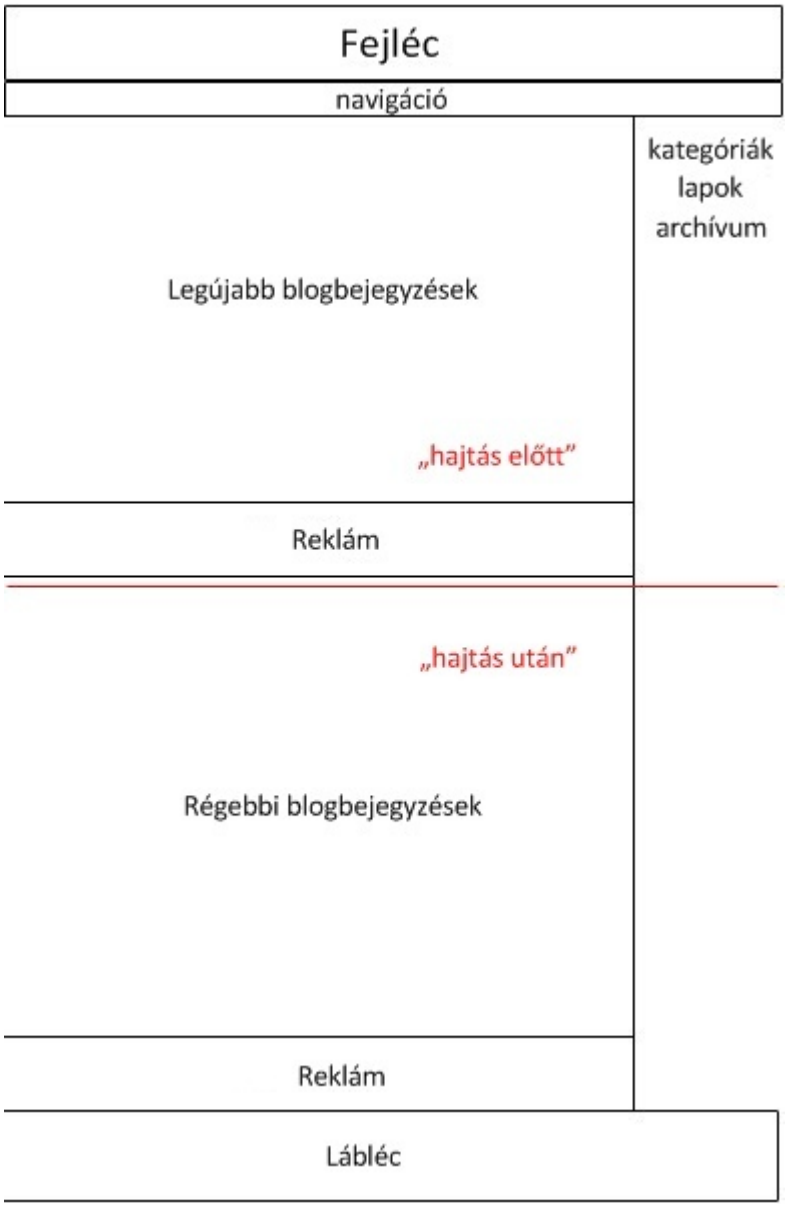
40-60 karakter általában megfelelő szélesség a szövegtörzsnek, de ez a különböző szempontok szerint változhat, mint például a betűméret vagy a célközönség.

2.27. ábra: Egy példa a helyes szélességre, egy széles képernyőn megjelenítve

Végül, de nem utolsósorban mindenképpen ellenőrizd le a szövegeidet és címsorokat az elírások és helyesírási hibák kiszűrése végett. Csak az ebben a cikkben szereplő hivatkozásoknak a felében találtam legalább egy elírást és néhány helyesírási hibát. Habár tévedni emberi dolog, a cég vagy a weboldal hitelessége szenved csorbát a helyesírási hibák vagy az egyszerű elírások miatt, amelyeket egyébként egyszerűen ki lehetne javítani.

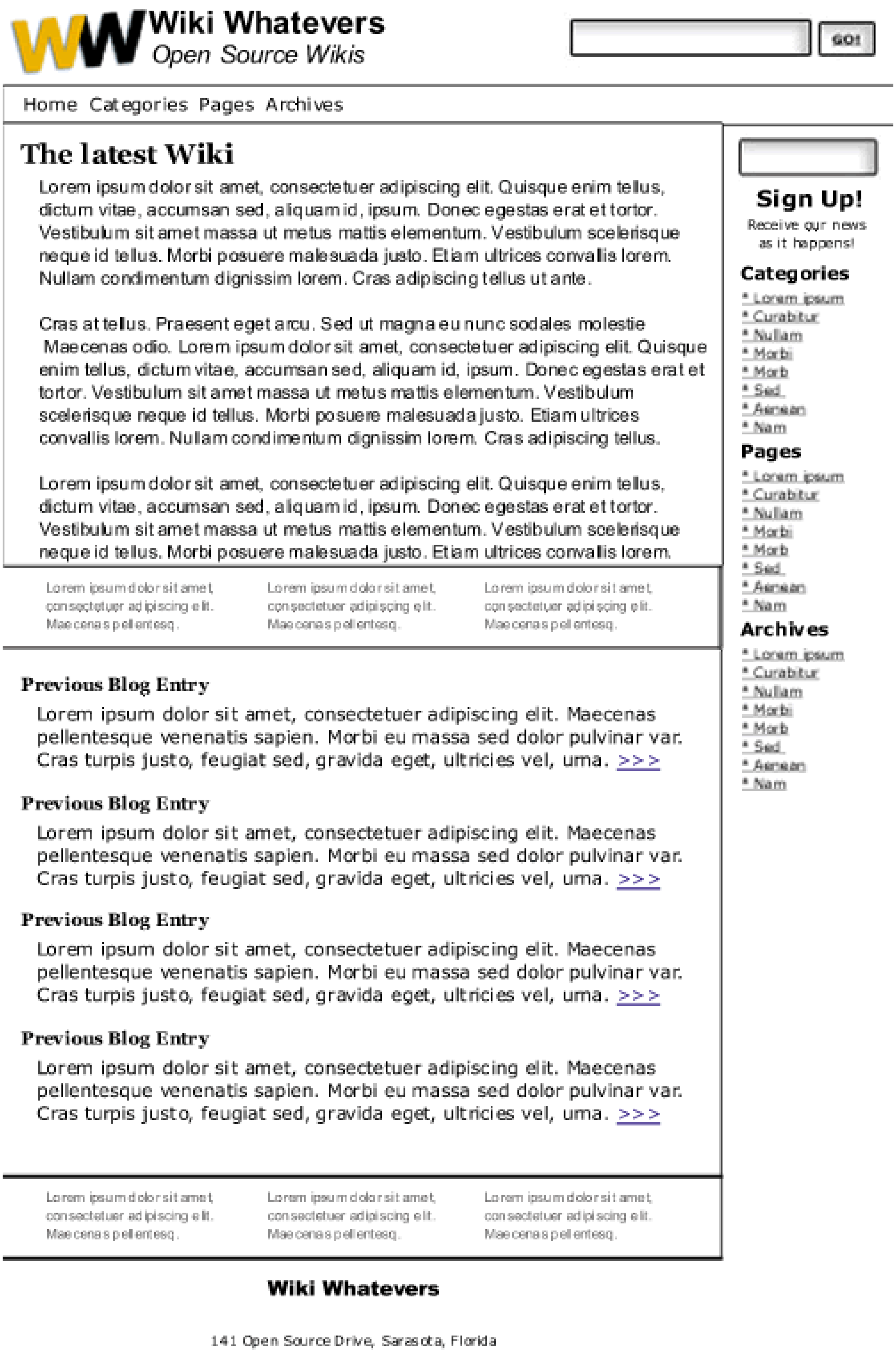
2.5.2 Második lépés: a tipográfia alkalmazása

Miután kiválasztottad a betűtípusokat a websitehoz, el kell helyezned az site korábban elkészített tervében a címsorokat, az alcímeket és szövegtörzseket. Az előző leírásban bemutattam egy képzeletbeli, „Wiki Whatevers” névre hallgató céget, akik meg akarták osztani az általuk írt kódrészleteket nyílt forráskódú anyagként. Elkészítettem a site felépítését, és örömmel jelenthetem, hogy tetszett nekik az elrendezés, amit mutattam! Bár ez a keret elég merev, viszont megakadályozza a klienst abban, hogy előítéletei legyenek azzal kapcsolatosan, hogy a különböző képek és grafikák hová kerülhetnek, beleértve a céges logót. A keret felépítése a 2.28. ábrán látható:



2.28. ábra: A Wiki Whatevers website vázlata

Most pedig hozzá fogom adni a laphoz a logót és néhány olyan szöveget, amelyet a cég bocsátott a rendelkezésemre, hogy megnézhessem, hogyan jelenik meg az elrendezésen belül. Van még egy ok, amiért hozzá szeretném adni a logót és valamennyi tartalmat az oldalhoz már ezen a ponton az az, mégpedig azért, mert ezek befolyásolhatják a színválasztást a későbbiekben. A szövegek, a címsorok és a többi tipográfiai elemek a weblapon saját „színt” visznek az oldalba. Csak hasonlítsd össze a lenti 2.29. ábrát az előző ábrával, és te is látni fogod a különbséget:



2.29. ábra: Az oldal vázlata a logóval és kitöltött szövegekkel

A fenti kép adott egy kis „színt” a weblaphoz, annak ellenére, hogy csak a logóban vannak valójában színek. A belehelyezett szöveg „tartalmat” adott az oldalnak, nem csak a keretet töltötte fel, hanem a megjelenést is feldobta a fehér, a fekete és a szürke árnyalataival. Csak azokat az elemeket adtam most hozzá, amelyek alapvetőek az oldalon: a logó, a website neve (ami egyben a cég neve is), a szlogen („Open Soruce Wikis”), a hivatkozások, a hírlevélre vagy a hírcsatornára feliratkozó link, a lábléc információk a cégről,

szöveges hivatkozások és a reklámrészek. A keresőmező túl nagy ahhoz, hogy az oldalsávba tegyem, így az a fejlécbe került.

Bár kész szövegekről beszéltem, valójában csak egy „tölteléket” használtam, amelyet a Lorem Ipsum generátor készített. Az ilyen generált szöveget nyugodtan behelyettesítheted a weblap szövege helyett, ha az még nem áll rendelkezésre.

Fontos az igazítás

Ezen a ponton megmutatom, miért tettem a tartalmakat éppen azokra a pozíciókra, ahol vannak. Az igazítás azt jelenti, hogy egy bizonyos tartalmat a rendelkezésre álló terület egy bizonyos helyéhez igazítjuk. Igazíthatod balra, miközben a jobb oldal szabadon „lóg” a levegőben, ez a hagyományos elrendezés. Vagy igazíthatod a szöveget középre, használhatod sorkizártan (amikor mind a bal, mind a jobb oldal igazított), esetleg igazíthatod jobbra is, miközben a bal széle szabadon marad.

Én a hagyományos baloldali igazítást választottam, ahol a betűk egy egyenes sort alkotnak a baloldalon. De feltűnhet az, hogy a szövegtörzs kissé jobbra van igazítva a címsorokhoz képest. Ennek az elrendezésnek valójában a logó az oka. Itt egy nagyított illusztráció a 2.30. ábrán, amely megmagyarázza a döntésemet:

2.30. ábra: Illeszkedés a szövegben a logó alapján

Mivel ez nem egy díjnyertes logó, ezért egy kis méretet használtam. Ráadásul az Arial Black betűtípus a logóban még szélesebbé teszi a megjelenést, így arányaiban ez lesz a legnagyobb elem a lapon. Bár jelentősen lekicsinyítettem a logót, azért elég nagy maradt ahhoz, hogy elférjen mellette a rendelkezésre álló magasságon a cég neve és a szlogenje, az „Open Source Wikis”. A piros vonalak alapján láthatod, hogy a cég neve pontosan ugyanabba a magasságba került, mint a logó, míg a szlogen alja a fekete „W” aljához illeszkedik. Ha az első „W”-t nézed a logóban, akkor láthatod, hogy címsor kezdete a betű bal alsó széléhez illeszkedik. Mivel a logó ferde, így az alsó pontja segít rávezetni a tekin-

tetet a címsorra a navigáció alatt. Valójában a navigáció csak másodlagos elemként jelenik meg, mivel a címsor a cég nevéhez hasonlóan ugyancsak félkövérrel van írva.

Ezen a ponton rájöttem, hogy a blogbejegyzések címeihez választott Georgia betűtípus túl élénk ehhez a területhez. Ezért a címsorok típusát átállítottam az Arial Black talpatlan betűtípusra, ami kissé különbözik a szövegtörzsben használt Verdanatól, de nem tér el annyira, hogy teljes káoszt okozzon.

Ez a lehetőség, hogy a különböző elemeket a weblap designjában több helyre lehet illeszteni, CSS használatával nagyon egyszerűen kivitelezhető. Bár egyes böngészők felülbírálhatják az igazítási törekvéseidet, azért a legtöbb esetben, ha pontosan megadod a kívánt pozíciót, akkor az jól jelenik meg. Ezért tudod a keresőmező alját a szlogen aljához igazítani, vagy a jobb szélét az alatta található mezőhöz, ahogy az a fenti példában is látszik.

Igazíthattam volna a szövegtörzset is a logó bal alsó sarkához, mint a címsoroknál, viszont a behúzás lehetőséget ad az olvasónak arra, hogy gyorsan átfussa a címsorokat, és azt a részt olvassa el, amelyik jobban érdekli. De minden weblap más és más, és egy eltérő logó teljesen más designt és igazításokat eredményezne. A lényeg az, hogy minden fontosabb elem a weblapon igazítva legyen, hogy szépen követhessék egymást. Az olvasó nem fogja észrevenni, hogy mennyit bajlódtál ezzel (sőt még a kliens sem, a legtöbb esetben). De ha nem foglalkozol eleget az illesztésekkel, akkor valaki előbb-utóbb meg fogja majd jegyezni, hogy „valami itt nem passzol”.

Ha egy kissé beljebb húzzuk a szövegtörzset, akkor egy üres hely is keletkezik előtte (ezt néha csatornának is nevezik), amely megkönnyíti az olvasónak, hogy szétválassza a szövegtörzset a weblap többi elemeitől. Az üres hely a szövegtörzs bal oldalán többnyire ugyanakkora szokott lenni, mint a jobb oldalán, így egyensúlyt teremthetsz. Az ilyen üres hely egy kis levegőt ad a weblapnak, így az nem lesz olyan zsúfolt.

Amikor legközelebb böngészel az interneten, nézd meg jól a látogatott weblapokat. Figyeld meg azokat a lapokat, amelyet jól designoltnak tűnnek számodra, vagy csak egyszerűen minden a helyén van rajta. Nézd meg a szövegtörzs, a címsorok, a logók és más elemek elhelyezkedését. Illeszkednek valamihez? Így talán észreveheted, ha a webdesigner elég időt fordított az apró részletekre is, hogy a design kevésbé legyen fontos, mint a tartalom; ami tulajdonképpen a jó design célja is egyben.

Azt garantálhatom, hogy az elkészült design problémákat fog okozni a különböző böngészőkben. Lehet, hogy a szöveg nem illeszkedik jól Safariban, viszont teljesen jó Operában és Firefoxban. Ez a munka is vár még ránk a későbbiekben, de csak azután, hogy kiválasztottuk a színeket.

Talán néhányan közületek észrevették, hogy végül mégsem az Arial Black típust választottam az alcímekhez és a cég nevéhez. Azért maradtam végül a Times New Roman mellett, mivel ez a talpas betűtípus egy kis kontrasztot ad az oldalhoz, ezzel mintegy felhívja magára a figyelmet. Az Arial Black átfogó használatával az oldal eléggé unalmas lenne.

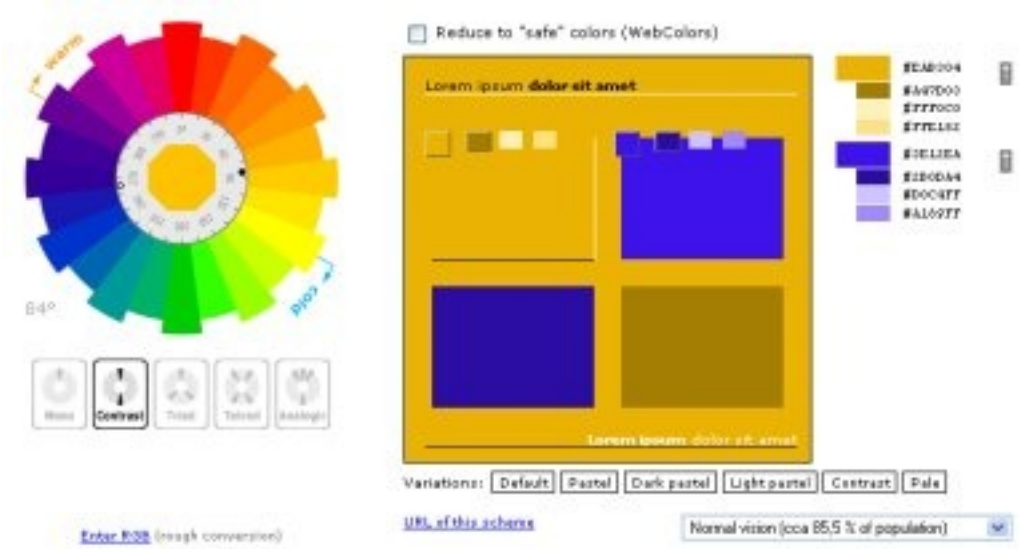
2.5.3 Harmadik lépés: színek

Miután elkészítettem egy website egyszerű vázlatát a kliensnek, rendszerint olyan sokáig várok a következő mintával, amilyen sokáig csak lehetséges, mielőtt újra megmutatnám neki. Ilyenkor már inkább kódot szeretek mutatni, ahelyett, hogy ismét egy képet mutatnék az elrendezésről. Ezzel a módszerrel már beletehetek a mintába több elemet is a websiteról, mint például a logót, szövegeket, sőt még reklámmintákat is, így a kliensnek sokkal jobb rálátása nyílik arra, hogy hogyan is fog a végén az oldal a végleges változatban megjelenni. Egy ilyen részletes elrendezéssel a kliensnek nem lesznek téves elképze-

lései arról, hogy a különböző elemek hogyan jelennek majd meg az oldalon. Emellett a kliens könnyebben dönthet arról is, hogy mit kellene még oldalhoz hozzáadni vagy törölni. Végül pedig, ha a modellt a kliensnek egy számítógépen tudom megmutatni, akkor fogalmat alkothat arról is, hogy hogyan néz majd ki a végleges oldal, amikor majd élesben meglátogatja.

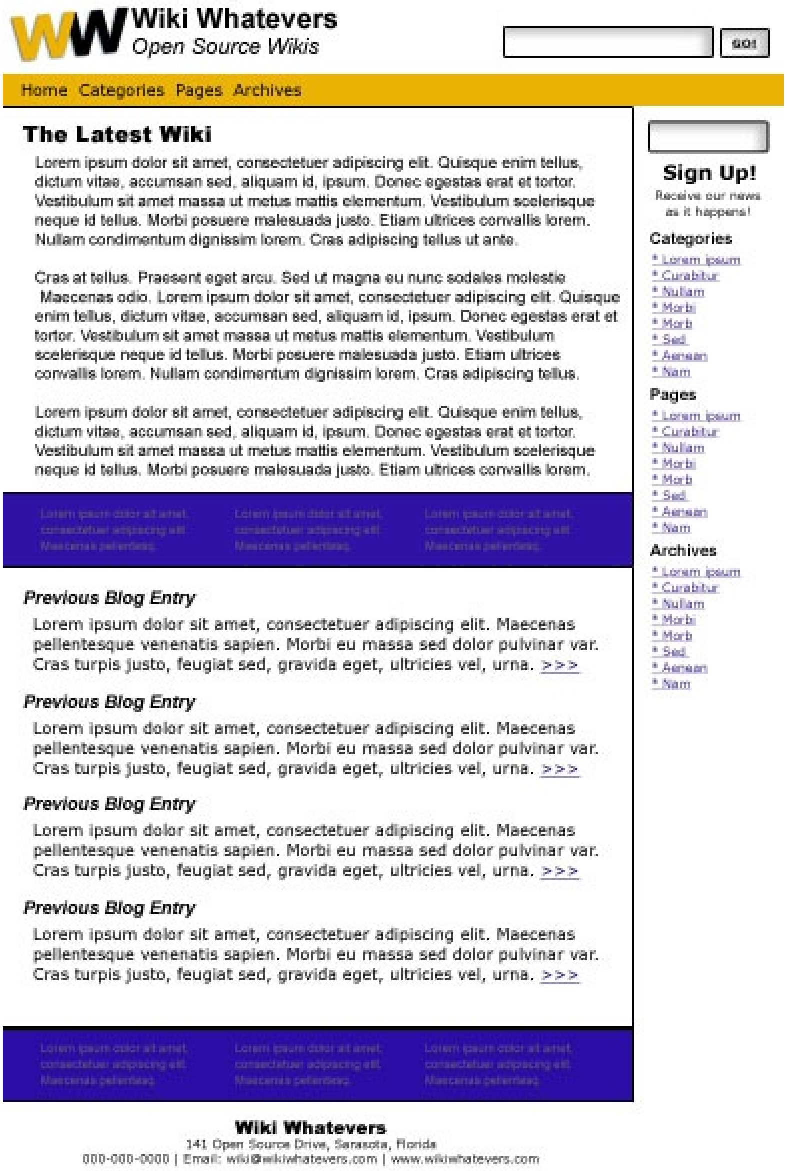
A színek a „minden a helyén van” mentalitás részét képezik. Ennek az az oka, hogy a különböző színsémák jelentősen meg tudják változtatni a teljes website hangulatát, még akkor is, ha minden elem már a helyére került. Ezen kívül a lehetséges színminták számát igyekszem a minimumom tartani, mert a túl sok minta zavaró lehet. Ebben az esetben a kliens korlátozott költségvetéssel dolgozik, ezért rábeszéltem őket, hogy csak egyetlen színsémával dolgozzunk.

Amikor bemutattam a színséma készítő eszközt a 8. leírásban, nem említettem meg azt, hogy egy hexa értéket is megadhatsz alapszínként a színséma generálásakor. Közvetlenül a színkerék alatt találsz egy „Enter RGB” nevű hivatkozást. Mivel ebben az esetben a logóban látható arany volt a legerősebb szín, ezért ennek a hexa kódját adtam meg (#eab304), hogy láthassam a lehetőségeket. A monokróm színséma elég unalmasnak tűnt, viszont a kiegészítő színséma már ígéretesnek látszott. Ez a séma egy kék-lila színt dobott ki (a 2.31. ábrán látható), amellyel már tudok dolgozni, mivel a logó mögötti árnyék ugyancsak kék árnyalatú.



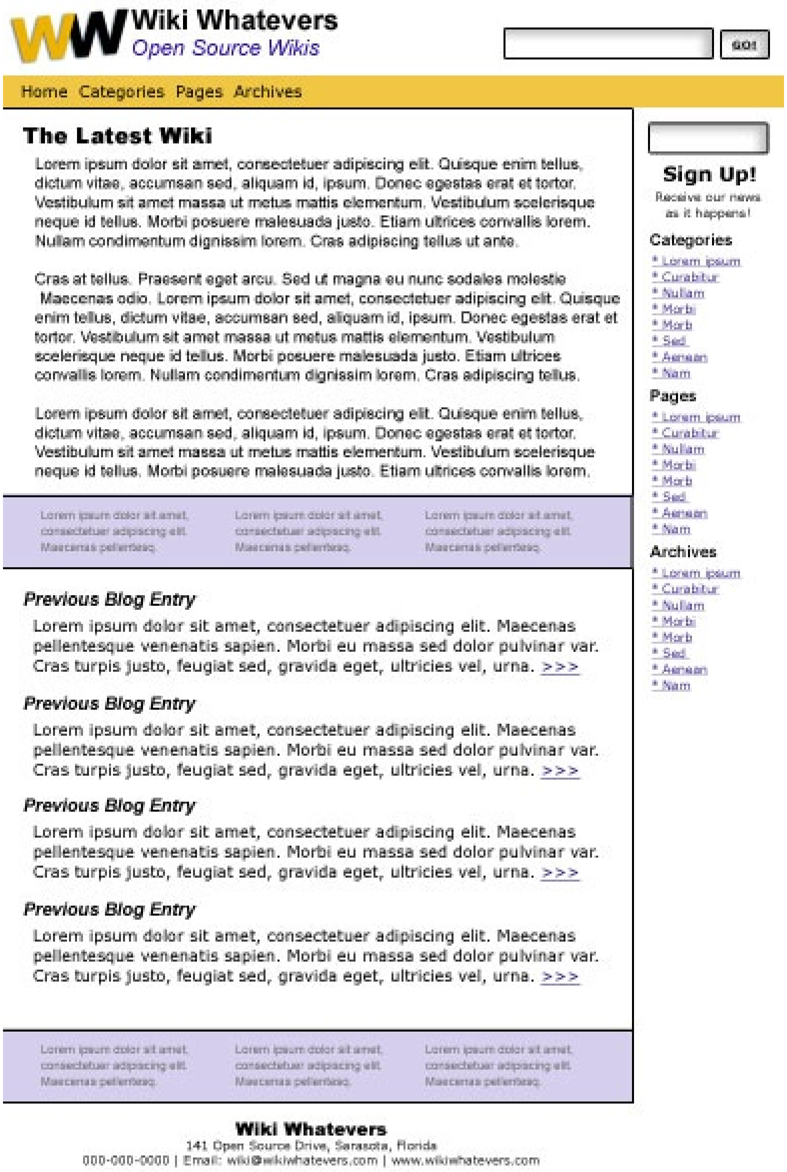
2.31. ábra: Kiegészítő színséma az #eab304 színre alapozva

Az itt látható színek alapján úgy döntöttem, hogy a felső navigáció hátterének a logó fő arany színét fogom használni. Egy sötétebb kéket (majdnem kék-lilát, #2b0da4) használok a hivatkozásoknál (amiket egyébként aláhúzok), és a reklámok hátteréhez ugyanennek a színnek használtam egy világosabb árnyalatát. A színek hozzáadása után az elrendezés jelenlegi állapotát a 2.32. ábrán láthatod:



2.32. ábra: A website vázlata kiegészítő színekkel

A képen jól látható, hogy ezek a színek túl sötétek és „nehezek” ehhez az oldalhoz. Így először a navigációs sáv színének az átlátszóságát beállítottam 75%-ra, majd a reklám hátterének az átlátszóságát egészen 20%-ra vittem le. A különbséget a 2.33. ábrán azonnal észreveheted:



2.33. ábra: A website vázlata a kiegészítő színek világosabb árnyalatával

A csökkentett színek a navigációs sávban jobban illeszkednek a logó színeihez. Az átlátszóbb, világosabb színek a reklámoknál jobban egyeznek a hivatkozások színeivel. Mivel a reklámmező ugyancsak hivatkozásokat tartalmaz, így a színválasztás illik az oldalsávhoz is. Az a tény, hogy színes hátteret választottunk a reklámoknak, jó dolog; ha olyan szolgáltatást használsz, mint a Google Adsense, akkor láthatod, hogy a Google szereti azt, ha a reklámok háttere más, mint a szövegtörzs háttere. A szlogenhez is egy új színt (#2B0DA4) választottam, így ez a kontrasztosabb kék szín végül keretbe fogja az egész oldalt.

Bár ez az elrendezés elég egyszerűnek tűnik, azért eltöltöttem azzal egy kis időt, hogy próbálgattam a színkerékből kapott színeket a hátterek, címsorok és a reklámok színeihez. Minden változtatásnál úgy tűnt, hogy a színezés túllép az egyszerű elrendezésen, így végül az egyszerű feketénél maradtam a betűk esetében (kivéve a szlogent). Még megadhattam volna egy „látogatott hivatkozás” színt is, de azt hiszem jobb lett így, hogy maradtam a két színből álló alapnál, és elkerülhettem a színkáoszt.

Ebben a lépésben láthatod azt is, hogy miért jó az, ha előre felépítjük a website egyszerű vázlatát. Ha már készen van a „váz" első terve, akkor a színek hozzáadása sokkal egyszerűbb, hiszen így már csak „vonalon belül kell maradni". Ilyenkor hagyd, hogy az elrendezés mutassa meg, hogy melyek a legjobb színek ahelyett, hogy az elrendezésen utólag változtatnál. Másrészt, ha a designt meghagyod egyszerűnek, akkor hosszabb távon elegánsabb lesz az oldalad, ráadásul használható és hozzáférhető lesz.

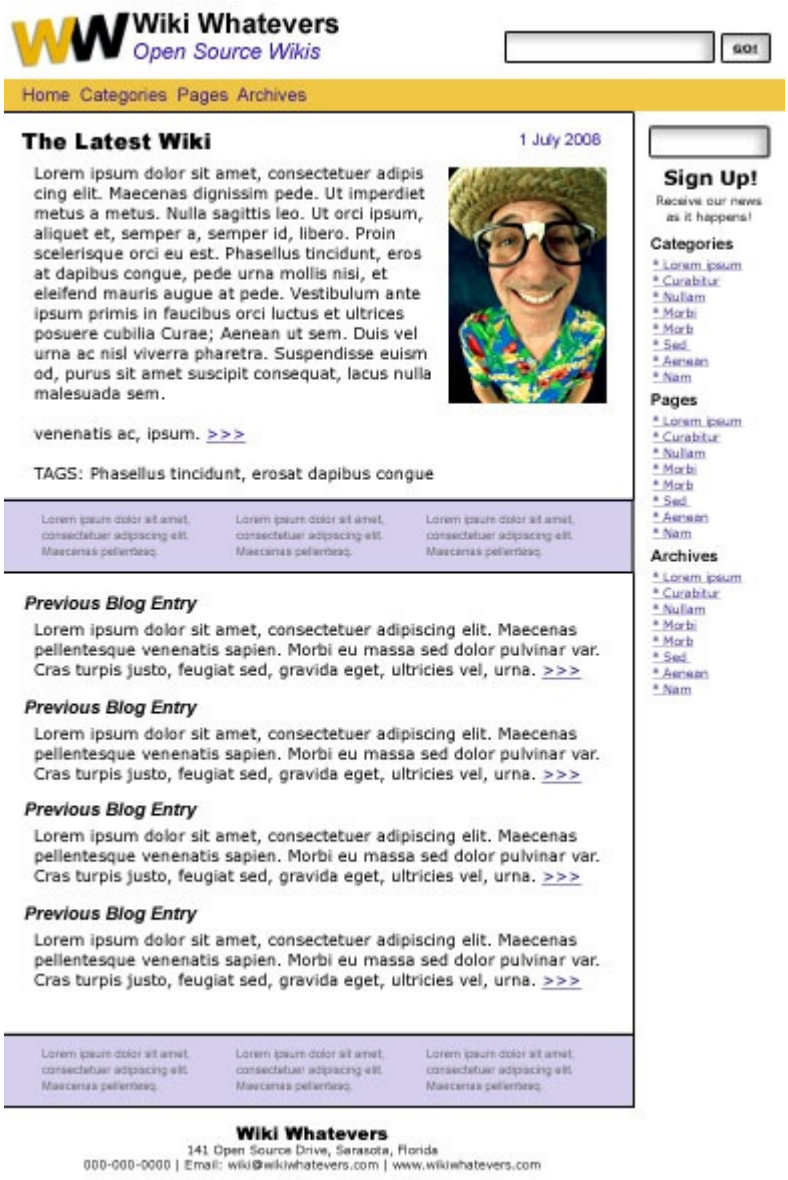
Van még egy utolsó jó indok arra, hogy az elrendezést egyszerűnek hagytam: a belső oldalakon ugyanis több kódrészlet is fog szerepelni, és ezek megjelenítéséhez egy fix szélességű betűtípust fogok használni, a legjobb módszereknek megfelelően. Ez az oka annak, hogy hasonló betűtípusokat választottam a szövegtörzshöz és a címsorhoz is. A fix szélességű típusok és reklámokban használt betűtípusok használata éppen elég változatosságot biztosít az oldal betűtípusaiban. Figyelj arra is, hogy a címsorokhoz és az alcímekhez a megfelelő elemeket használd (h1, h2, h2, stb) ahelyett, hogy vastagítanád (strong) vagy kiemelnéd (em). A megfelelő címsor elemekkel az oldalad használhatóbb lesz. Így a stílusfájlban (CSS) később egyszerre módosíthatod a címsorok és alcímek megjelenését.

Néhány megjegyzés még a fenti oldallal kapcsolatban:

- A cég nevét a lap felső részén azért hagytam feketén, mivel ez továbbviszi a logóból a fekete színt az egész fejlécben.
- Az oldalsáv felső részén a szöveget középre igazítottam, hogy felhívjam a figyelmet a regisztrációs szolgáltatásra. Mivel a szövegmező teljes egészében kitölti az oldalsáv szélességét, így a középre igazított szöveggel egyensúlyban van, és így látszatra is összetartoznak az elemek.
- A középre igazított lábléc először úgy tűnik, mintha elcsúszott volna, de azt akartam, hogy elsősorban a szövegtörzshöz tartozzon, és nem az oldalsávhoz. A site növekedésével az oldalsávban szereplő linkek száma is növekedni fog, így jobb, ha már előre elválasztjuk a láblécet az oldalsávtól. Így a látogatók inkább az oldalsávban fognak további információk után keresni.

Végül utolsó lépésként adok még néhány képet az oldalhoz. Azt leszámítva, amikor a kliensnek már van valamilyen használható képe, olyan képet érdemes választanod, amelyik „passzol" az oldal elrendezéséhez. Más szóval próbálj olyan képet keresni, amelyikben főleg az oldalon használt színek vannak. Ebben az esetben próbáltam egy kis humort vinni a lapba egy képgyűjtőből származó geek fotóval. Azért választottam ezt a képet, mert a fickó éppen szembenéz velünk, így ez a kép már a többi elem előtt meg tudja fogni a látogató tekintetét az oldalon. Nagyon örültem annak, hogy a pólóján a színek között van ahhoz hasonló arany és kék szín, mint amilyeneket az oldalon használtam. Végül a fekete napszemüvege tükrözi a lap fejlécének fekete beütését. Mindezek ellenére adtam Photoshopban egy kis kéket az árnyékokhoz, és egy kis sárgát a fényekhez, hogy még jobban illeszkedjen az általános színsémába.

Hozzáadtam még az oldalhoz egy sort a címkéknek, valamint egy dátumot, hogy a látogatók láthassák, milyen régi egy bejegyzés. Már ezek az elemek is „súlyosabbá" és zavaróbbá tették a siteot. Még egy ok arra, hogy miért kell a fontos információkat a lehető legegyszerűbben tartani: a látogatónak így is éppen elég kattintanivalója van, miután megnyitotta az oldalt. A 2.34. ábrán láthatod a végleges verziót:



2.34. ábra: A végleges design, készen arra, hogy megmutassuk a kliensnek

A webdesign jó oldala az, hogy nem olyan, mint a nyomtatott design. A nyomtatás végleges; a web folyamatosan változik. Vagyis az évek során az oldal teljesen megváltozhat, reagálva a csapat növekedésére. A hibákat ki lehet javítani, a színeket meg lehet változtatni. Mégis jó az, ha a lehető legjobb terméket próbálod átadni a vevőnek, amikor elkészülsz a designnal. A legjobb termék nem csak a te hírnevedet növeli, hanem egyúttal a kliensed hírnevét is.

Most pedig még egyszer, utoljára átnézzük az elkészült terveket, mielőtt újra megmutatnánk a kliensnek.

2.5.4 Negyedik lépés: tesztelés

A weblap tesztelése ezen a ponton azt jelenti, hogy a designer még egyszer átfésüli az oldalakat olyan hibák után kutatva, amelyeket egyszerűen ki lehet javítani még az oldal „élesítése” előtt. Több lehetőséged is van a tesztelésre, és több különféle tesztet is érdemes elvégezned.

- **Tipográfia és hivatkozások tesztelése**: a tipográfiai és a helyesírási hibák kereséséhez használhatsz barátokat, fórumozókat vagy professzionális szövegszerkesztőket. Miközben átnézik az oldaladat, megkérheted őket arra is, hogy ellenőrizzék az oldalon található hivatkozásokat is, hogy biztosan működnek-e. Figyelj arra is, hogy a legtöbb megbízó nem szereti, ha az oldala már azelőtt kikerül valamilyen formában a publikus netre, mielőtt „élesítve" lenne a weben. Ilyen esetben építsd be egy szerkesztőnek az árát a számlába, és a szerkesztővel írass alá egy titoktartási szerződést, amely kötelezi őt arra, hogy a website általa megismert tartalmát nem adhatja tovább.
- **Kód érvényességének ellenőrzése**: használd a W3C validátorait a HTML és CSS kódjaidhoz minden olyan esetben, amikor valamilyen új kódot adsz az oldalhoz. Ha ezt nem végzed el, akkor a következő lépés félrevezethet, hiszen egy validálási hiba problémát okozhat egyes böngészőkben. Idővel megtanulod majd azt is, hogy minden olyan reklámkód, amelyet az oldalhoz adsz, nem lesz valid. Ennek ellenére ne írd át a reklámok kódjait, mert új hibákat vihetsz be a reklámokba. A legtöbb webdesigner már megszokta, hogy a reklámkódok ilyenek, és nem tudnak sokat tenni ez ellen. Szerencsére a reklámok általában nem okoznak problémát a következő lépésben sem.
- **Böngésző kompatibilitás tesztelése**: most talán arra gondolsz, hogy ehhez be kell szerezned több különböző számítógépet és különböző méretű monitorokat, hogy minden operációs rendszeren és felbontásban le tudd tesztelni a weblapot. Ez viszont nem szükséges. A legtöbb webdesigner nem engedhet meg magának ekkora pazarlást, így hát kitaláltak néhány módszert arra, hogyan tesztelhetik le a weblapjukat a különböző böngészőkben. Az egyik ilyen, hogy több különböző verziójú böngészőt töltesz le ugyanarra a gépre. Megkérhetsz barátokat és fórumozókat, de a legjobb, ha olyan szolgáltatásokat használsz, amelyek képernyőképeket készítenek különböző rendszereken. Az ilyen rendszerek alapjában véve különböző rendszereken és böngészőkön készítenek képernyőképet a weboldaladról, így láthatod majd, ha a betűtípusod túl nagy az egyik felbontáson vagy túl kicsi egy másikon. Vagy felismerheted, ha valamelyik felhasználónak vízszintesen kellene görgetnie valamelyik böngészőben. A legtöbb esetben én az ilyen képernyőképes szolgáltatásokat használom, mivel a másik két módszer eléggé esetleges. Ráadásul nem szeretem idő előtt kiküldeni a kliens anyagait, ha ezt megoldhatom másképp, bizalmasan is. Elérhető néhány ingyenes szolgáltatás is erre a célra, de ezekkel azt tapasztaltam, hogy sokat kell várni az eredményre, és a lehetséges környezetek száma eléggé korlátozott. Így inkább fizetek egy keveset a BrowserCam oldalon, mivel ők már bizonyították a megbízhatóságukat az évek során. Mivel van egy rövid ingyenes próbaidejük is, így ki tudod próbálni, hogy hogyan működik anélkül, hogy fizetned kellene.
- **Hozzáférhetőség és használhatóság tesztelése**: a hozzáférhetőség tesztelésére is találhatsz néhány online eszközt a weben. Egyes szolgáltatások segítségével egy képernyő-felolvasón keresztül „olvashatod" el az oldalt. Más eszközök javaslatokat tehetnek neked a kód módosítására vagy a színek javítására, hogy nagyobb kontrasztot biztosíts a gyengénlátó látogatóknak. A használhatóságra is találhatsz hasonló online eszközöket és tennivaló listákat, amelyek segítenek abban, hogy a weboldal designját mindenki ugyanolyan könnyen használhassa.

A tesztelés egy unalmas foglalkozás, és könnyen előfordulhat, hogy a remek kis designterved jól néz ki az egyik böngészőben, viszont a másikban olyan, mint a múlt heti újramelegített spagetti. Ilyenkor a legfontosabb, amire figyelned kell, hogy a tartalom mindig legyen elérhető; ha a teljes tartalom látható és hozzáférhető minden böngészőben,

akkor az a kis üres rész a fejléc tetején, ami csak egyetlen böngészőben jelenik meg, már nem oszt, nem szoroz. Ha a felhasználók túlnyomó többségének nincs gondja az online anyagok elérésével a siteon belül, akkor már elérted a célodat; sok designer elfelejti ezt, amikor azért küzd, hogy a díjnyertesnek gondolt desingja mindenhol pontosan ugyanúgy jelenjen meg.

2.5.5 Tesztkérdések

- Melyek a négy legfontosabb betűtípuscsoportok?
- Milyen betűtípusok a legjobbak a szövegtörzshöz, és miért?
- Miért fontos, hogy elég kontraszt legyen a szövegtörzs betűszíne és háttérszíne között?
- Nevezz meg legalább két módszert egy oldal szövegének megtörésére.
- Mondj egy jó okot arra, hogy miért jó egy lap tipográfiáját még a képek hozzáadása előtt elkészíteni.
- Sorolj fel négyféle igazítási típust.
- Magyarázd meg, hogyan teheti az igazítás átláthatóbbá a weboldalt.
- Mi az a titoktartási szerződés, és mikor érdemes használnod?
- Miért fontos ellenőrizni a helyesírást a weblapon?
- Nevezz meg négy módszert a weblap tesztelésére az „élesítés” előtt.

2.6. Tipográfia a weben

Mi a tipográfia? Egyszerűen fogalmazva ez a szövegek megjelenése, designja és elrendezése (erre típusként is hivatkoznak); egy olyan fogalom, amely még a nyomtatott médiából származik. Legalább annyira szól arról, hogy mit nem tehetsz meg a típusaiddal, mint arról, hogy mit tehetsz meg velük. A weben a tipográfia gyakran túl kevés figyelmet kap, ráadásul több olyan technológiai hiányosság is van, amely miatt a webes tipográfia nem olyan hatékony, mint a nyomtatott. Ennek ellenére a mai rendelkezésre álló eszközök mellett már semmi okod nem lehet arra, hogy ne használd ki a webes tipográfia előnyeit szép és stílusos típusok használatával.

Ebben a leírásban megnézzük alaposabban, hogy miért korlátozott a webes tipográfia a nyomtatotthoz képest, majd adunk néhány jó követendő tippet a webes tipográfia használatára, mindezt egy példán keresztül bemutatva. Ne aggódj, ha ezen a ponton még nem érted meg teljesen a CSS és HTML kódokat – most csak a designra kell figyelned. Az olvasás közben érdemes lehet magadnál tartani egy papírt és egy ceruzát, így közben papírra tudod vetni a szöveg elrendezésével kapcsolatos ötleteidet.

2.6.1 A webes tipográfia korlátai

A hagyományos nyomdászoknak kismillió lehetőség áll a rendelkezésükre, amikor szóba kerül a tipográfia, mint például a betűkészletek puszta száma vagy az elrendezési lehetőségek széles skálája. A webes tipográfia ennél sokkal korlátozottabb, mivel olyan típusokkal és elrendezéssel kell dolgozzunk, amelyről tudjuk, hogy elérhető és használható lesz azokon a gépeken is, amelyeken az olvasók megnyitják a lapot — senki nem fejleszt csak saját magának weboldalt.

A webes tipográfia korlátai többek között a következők:

- Korlátozott betűkészlet.
- Nincs elválasztás, így a sorkizárt elrendezés csúnya lesz keskenyebb oszlopok esetén.
- Nincs befolyásunk az alávágásra (a szóközökkel való feltöltésre).
- Nem lehet tudni, hogy hol és hogyan nézik majd meg a munkát, így a designereknek minden képernyőméretre, felbontásra és környezetre gondolniuk kell.

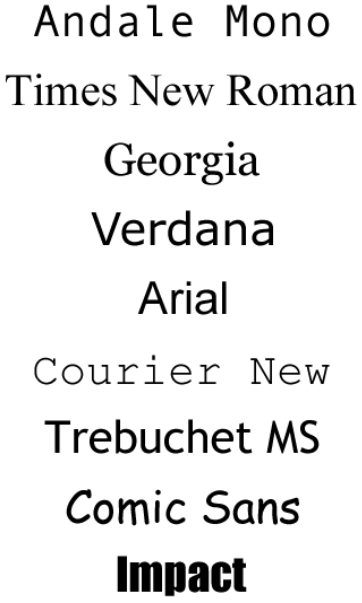
Most pedig lássuk ezeket a pontokat egy kissé részletesebben.

Korlátozott számú betűtípus

A betűtípusok száma általában az első nagyobb korlát, amelybe a szöveg stílusozásakor beleütközhetsz. Bár a CSS-ben olyan fontot adhatsz meg, amilyen csak jólesik, a látogatók viszont csak akkor fogják a szövegeket ezzel a típussal látni, ha az telepítve van a gépükön — ha nincs, akkor a böngészőjük megpróbál egy másik betűtípust választani a CSS alapján, vagy egyszerűen az alapértelmezett típust használja (ami általában a Times New Roman). Szóval ha te olyan különleges fontokkal akarod is megjeleníteni a szövegeidet, mint például a Trump Medieval vagy az Avant Garde, az olvasóid ebből mit sem fognak észrevenni, feltéve, hogy nem megszállott designerek több ezer telepített fonttal. Ebből az okból kifolyólag a legtöbb webdesigner csak a széles körben, több rendszeren is elterjedt fontokat használja, amely a használható betűtípusokat az alábbi listára korlátozza:

- Andale Mono
- Times New Roman
- Georgia
- Verdana
- Arial / Arial Black
- Courier / Courier New
- Trebuchet MS
- Comic Sans (ez egy borzalmasan amatőr típus — ne használd!)
- Impact

Ezeknek a típusoknak a megjelenését a 2.35. ábrán láthatod:



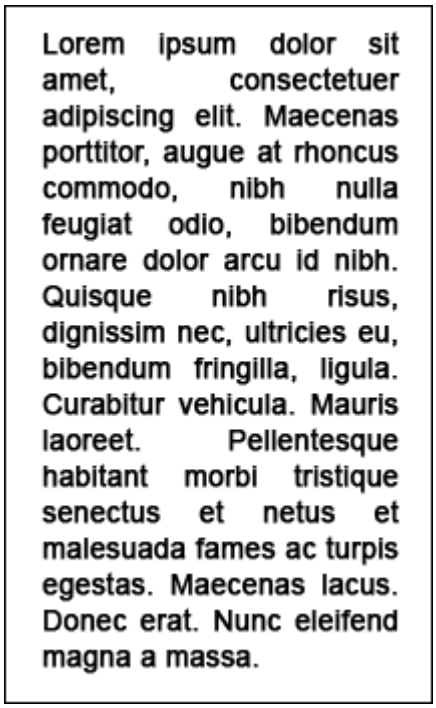
2.35. ábra: A legismertebb és a legtöbb rendszerben elérhető fontok listája korlátozott

Ha ezek közül a fontok közül választasz, akkor az azt jelenti, hogy jó eséllyel a látogatóidnak is meglesz a megadott típus. A Microsoft a fentieken kívül még bevezetett hat új betűtípust Windows Vista és XP alá, amelyek különös módon mindannyian C-vel kezdődnek. Ezek a Cambria, Calibri, Candara, Consolas, Constantia és Corbel. Mégsem ajánlom ezeknek a betűtípusoknak a használatát, mert nem túl valószínű, hogy a Mac és Linux platformokon is telepítve lesznek.

Így aztán a nyomdászok számára elérhető több ezer betűkészlettel szemben a webdesignerek alig néhány betűtípus között választhatnak. De valóban olyan nagy korlát ez? A tipográfia ugyanis sokkal többről szól annál, mint hogy kiválasztunk egy tetszetős típust, sokkal inkább a sormagasságokról, az alávágásról, az üres területekről — ne feledkezzünk meg arról, hogy eleinte a nyomdászok is szembesültek ugyanezzel a korláttal.

Szóelválasztás

Ha a szöveg elrendezéséről van szó, akkor alapvetően négy lehetőség közül választhatsz: balra igazított, jobbra igazított, középre igazított, vagy sorkizárt elrendezés. A sorkizárt elrendezésben a szöveg jobb- és bal oldala is illeszkedik a befoglaló terület függőleges széleihez, és így jobban nézhet ki, mint a „töredezett” balra igazított elrendezés. Gyakran láthatsz ilyet a könyvekben és a magazinokban is. A weben viszont van vele egy kis probléma, mivel hiányzik az automatikus elválasztás, amely a megfelelő helyeken szétválaszthatná a szavakat, hogy így jobban kitöltsék a rendelkezésre álló helyet. A sorkizárt elrendezés érdekében a legtöbb, amit a böngésző tehet, hogy a szavak közötti hézagokat feltölti, amely így függőleges irányú „fehér folyókat” eredményez a szövegben. Ez általában akkor történik meg, ha a sorok hossza a keretben túl rövid, és nincs elég hely a szavak jobb elrendezésére, ahogy a 2.36. ábrán látszik.

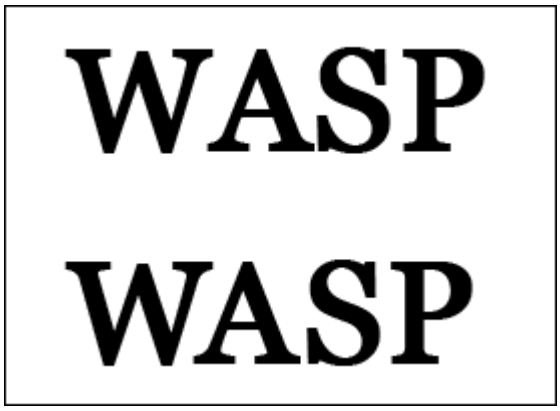


2.36. ábra: A „fehér folyók" elronthatják a sorkizárt elrendezést

Amint azt a fenti képen láthatod, a jobb helykihasználást lehetővé tevő szóelválasztás nélkülözése egyes sorokban túl nagy üres helyet hagyott a szavak között. Ennek az elkerülésére a weben többnyire érdemes balra igazított elrendezést használni.

Alávágás

Az „alávágás" különböző betűpárok közötti távolság módosítását jelenti a változó betűszélességű típusokon belül (mint amilyen a Times New Roman, amelyben a különböző betűk szélessége és a köztük levő távolság betűnként eltérő, ellentétben a Courier típussal, amelyben a betűk mérete és a távolság közöttük mindig ugyanaz). Nyomtatásban arra használják, hogy összehúzzák az olyan betűket, amelyek összeillenek, így helyet takarítanak meg, és a szöveg is jobban olvasható. Ilyen lehet például egy V utáni A, amelyeket ha összehúzunk, sokkal komolyabb megjelenést kaphatunk. A szakszerű fontok tartalmaznak útmutatásokat az alávágások kezelésére, amelyben megadják az egyes betűk közötti távolságokat. A 2.37. ábrán láthatod, hogy hogyan módosítja a megjelenést az alávágás.



2.37. ábra: Az alávágás javítja a szövegek megjelenését

Az előbbi ábrán az első szóban nem alkalmaztuk az alávágást. A második szóban viszont a távolság csökkent a W és az A között, és megnőtt az A és S között.

A weben az ilyen szintű alávágás sajnos nem elérhető. Az egyetlen dolog, amire van befolyásunk, az az általános betűköz, amely a nyomtatott sajtóban a betűk közötti távolságot jelenti, függetlenül attól, hogy milyen betűkről van éppen szó. Így csökkentheted a távolságot a W és az A között, de ezzel egyúttal minden más karakter között is csökken a távolság. A weben ezt a letter-spacing CSS tulajdonsággal állíthatjuk be, amelynek egyik lehetséges megvalósítását a 2.38. ábrán láthatod.

2.38. ábra: Alávágást nem használhatunk a weben, helyette próbálkozhatunk a jóval általánosabb betűközzel is

Az előző képen megnöveltük a távolságot a különböző betűk között egyenlő mértékben. Bár ez segített az A és az S széthúzásában, viszont a távolság a W és az A között még nagyobb lett. A betűköz állítása éppen ezért nem egyszerű CSS-ben, mert mindent-vagy-semmit természete van, így jobb, ha nem használjuk.

Az irányítás hiánya

Bár sokat beszéltünk eddig a nyomtatott sajtóról, egy fontos dolgot azért érdemes kiemelni ezzel kapcsolatban, mégpedig hogy a web nem nyomtatott sajtó. A nyomdászok nem kell azzal foglalkozzanak, hogy az olvasó átméretezi a szövegeit, nincsenek meg neki a választott típusok vagy nincs nála bekapcsolva az élsimítás. A webdesignerek ezt mind figyelembe kell vegyék, bár gyakran megpróbálnak ráerőltetni egy designt az olvasóra merev szövegszélességeket használva, fix szélességű és magasságú dobozokba helyezve a szövegeket, vagy egyszerűen lecserélve a szövegeket képekre, amelyeken az adott szöveg látható.

Az irányítás hiányát viszont nem kell mindenképpen problémaként felfognod — egyszerűen csak azt kell észben tartanod, hogy az általad megjelenített tartalmat több különbö-

ző eszközön, több különböző környezetben, több különböző módon fogják olvasni a látogatók. Ne próbáld őket megállítani és ne nehezítsd meg a dolgukat fölöslegesen. Lehetséges, hogy néhány olvasód a mobiltelefonján akarja elolvasni az oldalad, miközben hazafelé utazik; talán először ki akarja nyomtatni az oldalt, hogy később olvassa el otthon; az is lehet, hogy nem lát olyan jól, mint te, és fel kell nagyítania az oldalakat, hogy könnyebben elolvashassa. Ez az oka annak, hogy a weblap tervezésekor a böngészőnek csak útmutatásokat tudsz adni, hogy te milyen módon szeretnéd megjeleníteni a szövegeket. Az egyes eszközök viszont ezt teljesen figyelmen kívül hagyhatják, és ez rendben is van — a lényeg, hogy ne erőltesd túlságosan a szövegekben a saját elképzeléseidet minden környezetre.

2.6.2 Hogyan készül a tipográfia a weben?

A webes tipográfiát teljes mértékben a CSS segítségével alakíthatod ki. A CSS több lehetőséget is biztosít a szövegek alakítására: nem csak a betűk méretét, színét és típusát adhatod meg, hanem a sorok magasságát, a használt betűközt, a kapitalizációt (minden nagybetű, csak a kezdőbetűk nagyok, kiskapitálisok vagy minden kisbetű), sőt még az első sor első betűjét is külön beállíthatod.

A szöveget tartalmazó doboz stílusozásával további lehetőségeket kapsz a szöveg elrendezésének és a sorok hosszának beállítására. Ezeket a stílusokat elég, ha csak egyetlen helyen adod meg — a saját stíluslapodon —, és ez minden szövegedre érvényes lesz a teljes website-on keresztül (de adhatsz stílusokat csak bizonyos paragrafusoknak vagy dobozoknak is). Ráadásul, ha később úgy döntesz, hogy nagyobb betűket szeretnél a weblapokon, akkor elég csak egyetlen helyen megváltoztatni a betűméretet a stíluslapon.

2.6.3 Gyors tippek

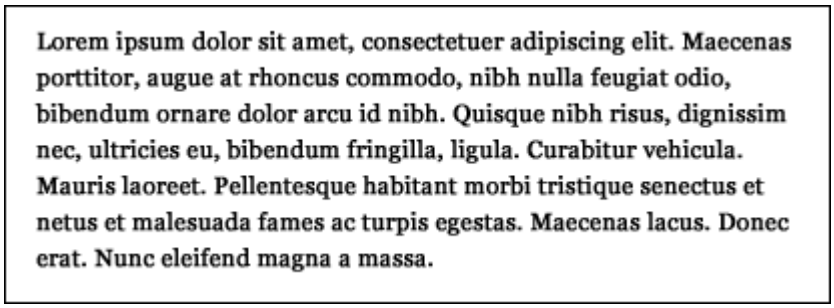
Az alábbi néhány tipp segítségedre lehet a webes tipográfia használatában.

Több betűtípus választása

Jó módszernek számít, ha egy betűtípus megadásakor felsorolsz még néhány további típust is tartaléknak. Vagyis ahelyett, hogy egyszerűen annyit adnál meg, hogy „Georgia”, megadhatsz még néhány más típust is: „Georgia, Cambria, "Times New Roman", Times, serif”. Így a böngésző először megpróbálja használni a Georgia nevű fontot, de ha ez nincs telepítve, akkor megpróbálja a következőt, a Cambriát, majd a Times New Romant, a Timest, és végül azt a fontot, amelyik az operációs rendszeren a „serif” (talpas) típushoz tartozik.

A sorok hossza

Az olvashatóság fenntartásához a sorok átlagos hossza egy szövegdobozon belül 40-60 karakter között kell legyen. Ez persze változhat a célcsoport szerint (a gyerekek a rövidebb sorokat, a felnőttek a hosszabbakat szeretik inkább). Az ideális sorhosszúságot a 2.39. ábrán láthatod:

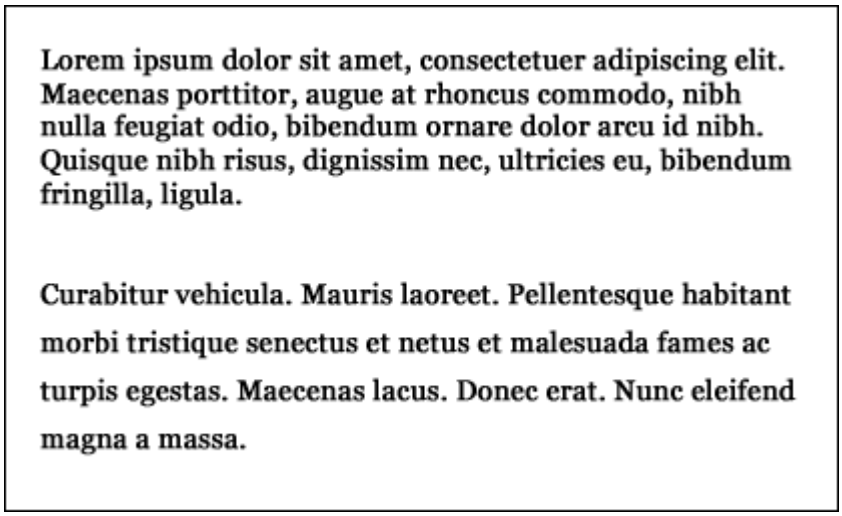


2.39. ábra: 60 karakter egy sorban — az ideális sorhosszúság

A képen látható szövegben nagyjából 60 karakter található soronként. Ha ennél többet használsz, akkor az olvasónak el kell kezdenie mozgatni a szemét, sőt akár a fejét is, ha követni akarja a szöveget. Ez megerőlteti a szemet, és a szöveg is nehezebben olvasható.

A sorok magassága

A sormagasság a sorok közötti függőleges távolságot jelenti. Ha ezt egy kissé növeled a böngésző alapbeállításához képest, akkor ezzel javíthatsz az olvashatóságon (és több helyed marad az alsó- és felső indexekre is). A különbséget a 2.40. ábrán láthatod:

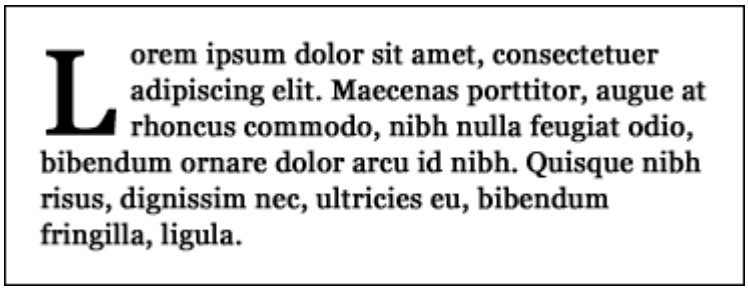


2.40. ábra: A sormagasság módosítása sokat javíthat a szöveg olvashatóságán

A fenti képen az első bekezdésben az alapértelmezett sormagasságot használtuk, és ez egy kissé zsúfoltnak hat. A második bekezdésben ezzel szemben megnöveltük egy kicsit a sormagasságot, így a szöveg levegősebb lett, és valamivel jobban olvashatóbb. Az ennél nagyobb sormagasság viszont már túlságosan is széthúzná a szöveget, így az ismét nehezebben olvashatóvá válna, szóval csak óvatosan a sormagassággal.

Iniciálék

A first-letter elem használatával (például p:first-letter { } formában) módosíthatod a bekezdés első sorának első betűjét a többi betűhöz képest. Ezt a betűt iniciálénak is nevezik, mivel általában 3–4 sornyi helyet foglal el a szövegben, ahogy a 83. ábrán láthatod.


Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor, augue at rhoncus commodo, nibh nulla feugiat odio, bibendum ornare dolor arcu id nibh. Quisque nibh risus, dignissim nec, ultricies eu, bibendum fringilla, ligula.

2.41. ábra: Egy tipikus iniciálé

Kiskapitálisok

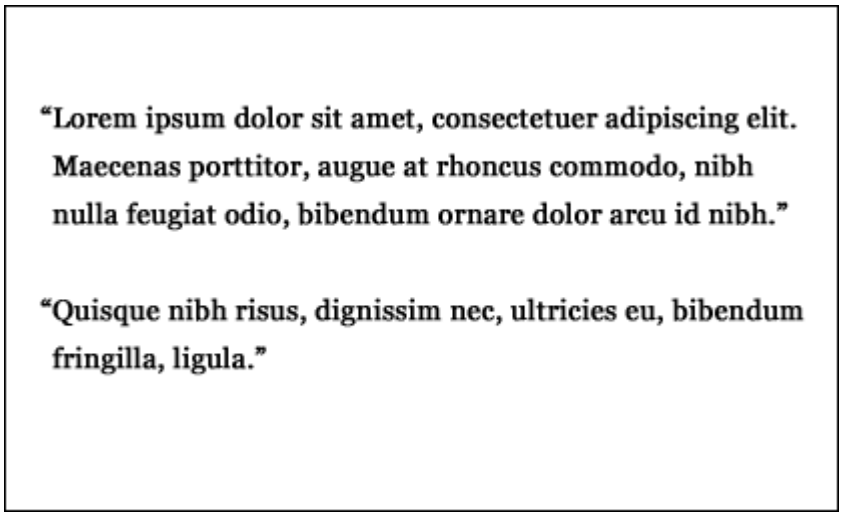
A fontok gyakran tartalmaznak kiskapitális változatokat is — ez azt jelenti, hogy a betűk mind nagybetű formátumúak, viszont a kisbetűk helyén méretben a kisbetűk méretének felelnek meg. Ez akkor lehet hasznos, ha valamit nagybetűvel szeretnél írni, de nem akarod túlságosan ráterelni a figyelmet, például rövidítések esetében. Ha a rendszerben nincs is a típusnak kiskapitális változata, az sem gond — a böngésző ilyenkor generál saját kiskapitálisokat a nagybetűk alapján, lekicsinyítve őket az eredeti méretük 70–80%-ára. A 2.42. ábrán látható a különbség a nagybetűk (minden szó első betűje) és a kiskapitálisok (a többi betű) között.


ABANDON HOPE
ALL YE WHO ENTER HERE

2.42. ábra: Kiskapitálisok akcióban

Kilógó írásjelek

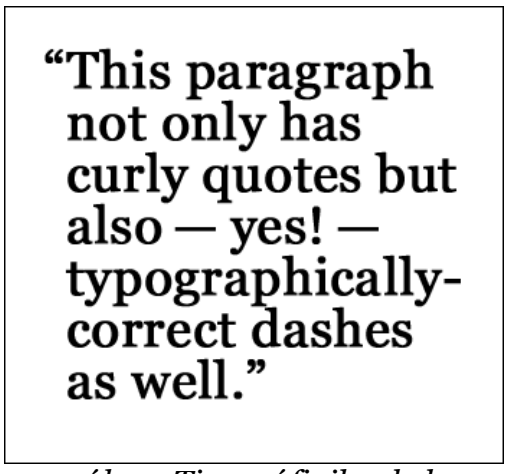
Jó tipográfiai hatást lehet kelteni a text-indent CSS tulajdonság használatával, ha a mondataid idézőjellel kezdődnek. Ha az előbbi tulajdonságnak negatív értéket adsz — akár relatív em értéket (-10em), pontokat (-10pt), pixeleket (-10px) vagy akár százalékot (-10%) —, akkor az idézőjelet kihúzhatod a bekezdés elé, ahogy a 2.43. ábrán látható:



2.43. ábra: Kilógó írásjelek

Tipográfiailag helyes írásjelek és más elemek

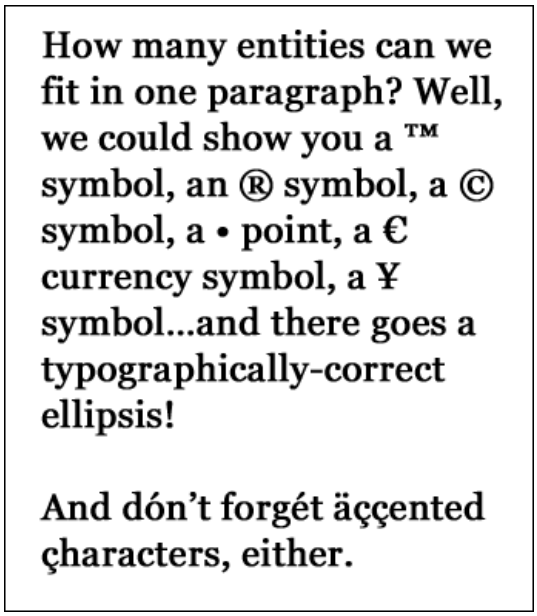
Ha a szövegedet professzionálisabbá és elegánsabbá szeretnéd tenni, akkor érdemes használod a HTML kódok széles skáláját a tipográfiai jelek megjelenítéséhez, például a tipográfiailag helyes „idézőjelek”, valamint kötőjelek (–) és gondolatjelek (—) használatával. Sok blogíró és szövegszerkesztő környezet automatikusan kijavítja ezeket, átalakítva az egyszerű idézőjeleket és elválasztójeleket a helyes formájukra. A 2.44. ábrán egy tipográfiailag helyes jeleket tartalmazó — angol nyelvű — szöveget láthatsz.



2.44. ábra: Tipográfiailag helyes (angol) írásjelek

Amint elkezded a helyes tipográfiai jeleket használni, a szöveged máris sokkal elegánsabb és professzionálisabb megjelenést kap — mintha egy magazin vagy egy könyv lenne, és nem online szöveg. Ne feledd el azonban, hogy ezek a jelek egyes régebbi képernyőkön vagy az élsimítás nélküli megjelenítőkön pixelesek lehetnek, úgyhogy óvatosan használd őket.

Vannak olyan HTML elemek is, amelyeket a kódba illesztve speciális karaktereket is megjeleníthetsz, amelyeket máskülönben csak nehezen vagy egyáltalán nem érhetsz el a billentyűzetről. A 2.45. ábra több ilyen jelet is tartalmaz:



2.45. ábra: HTML jelek

Ezeket beírhatod kézzel, de a legtöbb szerkesztő szoftverben van valamilyen lehetőség az ilyen jelek beillesztésére vagy automatikus átkonvertálására.

Kiemelt idézetek

A kiemelt idézetek rövid kivonatok a saját szövegedből, amelyek a lapon valahol a szöveg mellett vagy a szövegen belül jelennek meg nagyobb betűmérettel és sokszor más típussal is, hogy felhívják magukra a figyelmet. A legtöbb magazinban láthattál már ilyet, mivel ez egy hatékony módszer a szöveg felosztására és a fontosabb részek, kulcsszavak kiemelésére — ráadásul a weben nagyon könnyen alkalmazhatod ezt a módszert néhány egyszerű jelöléssel és stílusozással. Csak állítsd a szöveg méretét nagyobbra, esetleg másik betűtípusra, és állítsd lebegő módra, hogy a normál szöveg körbevegye, és már készen is vagy. A komolyabb megoldások között olyan módszert is találhatunk, amelyben egy JavaScript automatikusan kiemeli az idézetet a szövegből, így nem kell kétszer beírnunk azt a kódba.

2.6.4 Tesztkérdések

- Mi a különbség az alávágás és a betűköz között, és melyiket lehet használni a kettő közül a weben?
- Hogy kerülheted el a szövegben a „fehér folyókat”?
- Nevezd meg a CSS-ben használható négy különböző kapitalizációt.
- Melyik a legjobb sorhossz a szövegeknek, és milyen tényezők befolyásolhatják ezt?
- Mi a különbség a talpas és a talpatlan típusok között? Adj egy példát mindegyikre.
- Miben különböznek a kilógó írásjelek a hagyományos írásjelektől?
- Ha be akarsz helyezni egy copyright jelet a szövegedbe, egy HTML kódot kell használnod. Nézz körül az interneten, és próbáld megkeresni az összes elérhető HTML jel kódját. Legalább 250 van belőlük!

3. HTML alapok

3.1. A HTML alapjai

Ebben a leírásban megismerheted a HTML alapjait — mi az, mire jó, hogyan alakult ki, valamint hogyan épül fel egy HTML dokumentum szerkezete. A következő leírásokban majd részletesebben is megismerheted a HTML különböző részeit.

3.1.1 Mi a HTML?

A legtöbb olyan asztali alkalmazás, amelyik fájlokat ír és olvas, egy bizonyos fájlkiterjesztést használ. A Microsoft Word például a „.doc” fájlokat érti meg, míg a Microsoft Excel az „.xls” fájlokat ismeri. Ezek a fájlok utasításokat tartalmaznak arra, hogy hogyan lehet felépíteni a dokumentumot a legközelebbi megnyitáskor, mit tartalmaz a dokumentum, valamint néhány „metaadatot” a dokumentumról, mint például a szerző vagy a legutolsó módosítás dátuma, sőt akár az utolsó módosítások is benne lehetnek, így visszalehet állni a dokumentum egy korábbi verziójára.

A HTML („HyperText Markup Language”, azaz hiperszöveg-leíró nyelv) a webes dokumentumok leírására szolgáló nyelv. Olyan speciális jelöléseket (más szóval elemeket) tartalmaz, amelyek körülveszik dokumentum szövegeit, és megadják, hogy a kliens eszközök hogyan értelmezzék a dokumentum megjelölt részeit.

A „kliens eszközök” kifejezést használtuk itt a „webböngészők” helyett. Egy kliens eszköz lehet bármilyen szoftver, amely a felhasználó számára weboldalakat ér el. Fontos megemlítenünk még ezen a ponton, hogy az asztali böngészők (Internet Explorer, Opera, Firefox, Safari, stb.) és az alternatív böngészők (mint például a Wii Internet channel vagy a mobil böngészők, mint az Opera Mini vagy az iPhone WebKitje) egyaránt kliens eszközök, viszont nem minden kliens eszköz böngésző szoftver. Az olyan automata programok, mint amilyenekkel a Google és a Yahoo! indexeli a webet a keresőjük számára, ugyancsak kliens eszközök, viszont nem állnak közvetlen emberi irányítás alatt.

3.1.2 Hogyan néz ki a HTML?

A HTML a szövegek és az jelölések egyszerű szöveges megjelenítése. Például a fenti „Hogyan néz ki a HTML?” címsorhoz tartozó HTML kódrészlet a következő:

```
<h2 id="hogynezki">Hogyan néz ki a HTML?</h2>
```

A *<h2>* rész egy jelölő (amire „tag” — ejtsd teg — névvel hivatkozunk), és azt jelenti, hogy „a következő rész második szintű címsornak számít”. A *</h2>* ugyancsak egy tag, amely a második szintű címsor lezárását jelöli (éppen ezért „lezáró tagnek” is nevezik). A nyitó tag, a záró tag és minden, ami köztük van, együtt alkotja az „elemet”. Sokan a tag és elem kifejezéseket felcserélhetőként használják, pedig ez nem teljesen helyes. Az *id="hogynezki"* egy attribútum; ezekről a későbbiekben lesz szó.

A legtöbb böngészőben találhatsz egy „Forrás” vagy „Forráskód” pontot, legtöbbször a „Nézet” menü alatt. Ha megtalálod, válaszd ki most, és tölts egy kis időt ennek a lapnak a HTML forráskódját nézegetve.

3.1.3 A HTML története

Az internet és a web története leírásban már olvashattál arról, hogy hogyan alakult ki a modern web. Amikor Tim Berners-Lee megalkotta a World Wide Webet, akkor elkészítette az első webszervert, az első webböngészőt és a HTML első verzióját is.

Bár a HTML rengeteget változott az első napok óta, sok része az első HTML dokumentációnak még mindig érvényben van a modern verzióban is, és az akkor definiált „HTML tagek" több mint fele most is létezik.

Ahogy egyre többen kezdtek weblapokat és alternatív böngészőket írni, úgy bővült folyamatosan új funkciókkal a HTML is. Sokat általánosan adtak hozzá (mint például az img elemet egy kép beillesztéséhez, amelyet először az NCSA Mosaic valósított meg). Más elemek védettek voltak, és csak egy-két böngésző valósította meg őket. Egyre nőtt az igény a szabványosításra, hogy a böngésző szoftverek készítőinek legyen egy dokumentumuk (más néven specifikáció), amelynek alapján egyértelműen eldönthetik, hogy a HTML hogyan néz ki, és hogy egy HTML elemet helyesen valósítottak-e meg, vagy nem.

Az IETF (Internet Engineering Task Force — a szabványos alap elkészítésére az internet egészéhez) a HTML ajánlásának első vázlatát 1993-ban adta ki. Ez 1994-ben elavult anélkül, hogy szabvánnyá vált volna, de arra késztette az IETF-et, hogy indítson egy csoportot a HTML szabványosítására.

1995-ben elkészült a „HTML 2.0", amely sok ötletet merített az eredeti HTML ajánlásból. Dave Raggett készített egy alternatív javaslatot is HTML+ névvel, amely sok új elem fejlesztésének alapjául szolgált a böngészőkben (például a képek behelyezésének a módszere a dokumentumokba, amelyben az NCSA Mosaic volt az úttörő).

A HTML 3.0 ajánlása még ugyanebben az évben érkezett, de hamar befejezték vele a munkát, mivel a böngészőgyártók teljesen más irányokban kezdtek el fejleszteni. A HTML 3.2 sok funkciót elhagyott a HTML 3.0-ból, és helyettük beépítette a népszerűbb böngészők (mint a Mosaic és a Netscape Navigator) különböző fejlesztéseit.

1997-ben a W3C kiadta ajánlásként a HTML 4.0-át, amelyben még több böngésző-sepcifikus kiterjesztést próbáltak meg ésszerűsíteni és egyszerűbbé tenni. Ezt azzal érték el, hogy több elemet elavultként jelöltek meg — ez azt jelenti, hogy ezek az elemek még léteznek ebben a verzióban (elavultként megjelölve), de a következőben már teljesen törölve lesznek. Ezzel próbálták meg a fejlesztőket rávenni arra, hogy a HTML-t szemantikusabban használják (erről részletesebben A webes szabványok modellje leírásban olvashatsz).

A HTML 4.01 1999-ben jelent meg, majd néhány elírást javítottak benne 2001-ben. Ez az utolsó HTML verzió, bár jelenleg már elérhető a HTML 5 vázlata is.

2000-ben a W3C kiadta az XHTML 1.0 specifikációját is, amely a HTML átdolgozása volt érvényes XML dokumentumra.

3.1.4 A HTML dokumentum szerkezete

A lehető legkisebb érvényes HTML dokumentum valahogy így néz ki:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
  <html>
    <head>
      <title>Példa lap</title>
    </head>
    <body>
      <h1>Helló világ</h1>
    </body>
  </html>
```

A kód a dokumentum típusát megadó (doctype) elemmel kezdődik, erről részletesebben a Megfelelő doctype választása leírásban olvashatsz. Ez az elem adja meg, hogy melyik HTML változatot fogjuk használni, így a kliens eszközök tudni fogják, hogyan értelmezzék a dokumentumot, valamint hogy betartja-e a dokumentum az adott típus szabályait, vagy nem.

Ezután egy nyitó html elemet láthatsz. Ez közrefogja az egész dokumentumot. A záró html tag az utolsó a HTML dokumentumban.

A html elemen belül találjuk a head elemet. Ez a elem információkat tartalmaz a dokumentumról (a metaadatokat). Erről részletesebben a következő leírásban lesz szó. A head elemen belül találjuk a title elemet, amely a lap címét („Példa lap”) definiálja az ablak fejlécében.

A head elem után következik a body elem, amely a lap valódi tartalmát fogja közre — amely ebben az esetben csak egy első szintű címsor elemből (h1) áll, amely a „Helló világ” szöveget tartalmazza. Ez a teljes dokumentumunk.

Ahogy láthatod, az elemek gyakran tartalmaznak más elemeket. A dokumentum törzse mindig tartalmaz további beágyazott elemeket. Az oldal felosztása adja meg a dokumentum struktúráját, és tovább felosztásokat tartalmaz. Ebben lesznek a címsorok, a bekezdések, a listák, stb. A bekezdések ugyancsak több különböző elemet tartalmazhatnak: hivatkozásokat más dokumentumokra, idézőjeleket, kiemeléseket, stb. Ezekről az elemekről többet is meg fogsz tudni a későbbiekben.

3.1.5 A HTML elemek szintaxisa

Ahogy már láthattad, egy egyszerű elem a HTML-ben két jelölő tagből áll egy szövegblokk körül. Vannak olyan elemek is, amelyek nem fognak közre egy szöveget, és a legtöbb esetben az elemek további elemeket tartalmazhatnak (mint ahogy a fenti példában a html tartalmazza a head és a body elemeket).

Az elemek ezen kívül tartalmazhatnak attribútumokat is, amelyek módosíthatják az elem működését, vagy újabb értelmezést adhatnak az elemnek.

```
<div id="masthead">
  <h1>
    A <abbr title="Hypertext Markup Language">HTML</abbr>
alapjai
  </h1>
</div>
```

Ebben a példa div elemben (amely a lap részeinek logikus blokkokba való felosztására szolgál) használtunk egy kiegészítő id attribútumot, amely a masthead értéket kapta. A div tartalmaz egy h1 elemet (első, vagyis legmagasabb szintű címsor), amely tartalmaz egy szöveget. A szöveg egy részét az abbr elem fogja közre (ezzel adhatjuk meg a rövidí-

tések leírását), amely tartalmaz egy title attribútumot, ennek értéke a „Hypertext Markup Language” szöveg.

Sok attribútum a HTML-ben minden elemre használható, míg egyes attribútumokat csak bizonyos elemek mellett használhatunk. Mindig kulcsszó="érték" formában használjuk. Az értéket sima vagy dupla idézőjelbe kell tenni (bár egyes esetekben az idézőjeleket elhagyhatjuk, viszont ez nem jó módszer a előreláthatóság, az érhetőség és a világos, tiszta kód szempontjából — a legjobb, ha mindig kiteszed az idézőjeleket).

Az attribútumok és a lehetséges értékeik a HTML specifikációkban vannak megadva — nem hozhatsz létre saját attribútumokat anélkül, hogy érvénytelenné nem tennéd a HTML dokumentumodat, mivel ez összezavarná a kliens eszközöket és problémákat okozhat a weblap helyes értelmezésében. Az egyetlen kivétel az id és a class attribútumok — ezeknek a lehetséges értékei teljes mértékben a te kezedben vannak, mivel ezekkel tudod a saját értelmezésedet és szemantikádat hozzáadni a dokumentumhoz.

Egy elemre egy másik elemen belül az adott elem „gyerekeként” hivatkozunk. Így a fenti példában az abbr a h1 gyereke, ami viszont már a div gyereke. Fordítva, a div így a h1 elem „szülője” lesz. Ez a szülő/gyerek fogalom nagyon fontos, mivel ez biztosítja a CSS alapjait, és ezt használjuk majd a JavaScriptekben is.

3.1.6 Blokk szintű és inline elemek

A HTML elemeket általánosan két nagy kategóriára bonthatjuk, amelyek megfelelnek az elem által megjelenített tartalom és struktúra típusainak — ezek a blokk szintű és az inline elemek.

A blokk szintű (block level) elem egy felsőbb szintű elemet jelent, amely általában a dokumentum struktúráját jelzi. Úgy gondolhatsz ezekre az elemekre, mint amelyek egy új sort kezdenek, vagy elválasztanak valamit a korábbi tartalomtól. A leggyakoribb blokk szintű elemek a bekezdések (paragrafusok), listaelemek és táblázatok.

Az inline (azaz soron belüli) elemek ezzel szemben azok az elemek, amelyeket a blokk szintű elemek tartalmaznak, és csak a dokumentum kis részeit fogják közre, nem teljes bekezdéseket vagy csoportokat. Egy inline elem nem fog új sort generálni a dokumentumban; ezek azok az elemek, amelyek egy bekezdés szövegén belül találhatóak. A leggyakoribb inline elemek a hiperhivatkozások, a kiemelt szavak vagy mondatok és a rövid idézetek.

3.1.7 Karakter referenciák

Még van egy fontos pont a HTML dokumentumokkal kapcsolatban, mégpedig a speciális karakterek használata. A HTML-ben a <, > és & jelek speciálisak. Ezek adják meg HTML tagek elejét és végét, így a dokumentumon belül nem a kisebb, nagyobb és az és jeleket jelentik.

Az egyik legegyszerűbb hiba, amit egy webfejlesztő véthet, hogy az és jelet felhasználja a dokumentumban, így valami váratlan történik. Ha például azt írja, hogy „az angolszász jelölésben a tömeg stones&pounds”, akkor egyes böngészőkben ez úgy végződhet, hogy „...stones£s”.

Ez azért van, mert a „&pounds;” szövegrész valójában egy HTML karakter referencia. A karakter referencia egy módszer az olyan karakterek beillesztésére a dokumentumokba, amelyeket egyébként csak nehezen vagy sehogyan sem írhatunk be a billentyűzetet használva, vagy a dokumentum kódkészletében nem szerepel.

Az és jel (&) vezet be egy ilyen referenciát, és a pontosvessző (;) zárja le. Ennek ellenére sok kliens eszköz elég megbocsátó a HTML hibák iránt, és nem veszi figyelembe, hogy nincs lezáró pontosvessző, így a „&pound” kifejezést is karakter referenciaként értelmezi. A referenciák lehetnek számok (numerikus referenciák), vagy rövid szavak (egyedi referenciák).

Ha egy és jelet akarunk beírni a dokumentumba, akkor az „&” karakter referenciát kell használnunk, vagy a „&” numerikus referenciát. A karakter referenciák listáját az evolt.org oldalán találhatod meg.

3.2. A HTML *head* eleme

Ebben a leírásban a HTML dokumentumoknak egy olyan részével fogunk foglalkozni, amely méltánytalanul kevés figyelmet kap: ezek azok a jelölések, amelyek a head elemben találhatóak. A leírás végére tudni fogod, hogy mire szolgálnak ennek a résznek a különböző elemei, beleértve a title elemet, a kulcsszót és a leírást (amelyeket meta elemekkel adhatsz meg), valamint a head előtt található doctype elemet is megemlítjük. Fogunk beszélni még a JavaScriptről és a CSS-ről is (külső és belső megvalósításról egyaránt), valamint arról, hogy mit ne tegyünk a head elembe. A demo forrásfájlokat innen töltheted le, erre fogunk hivatkozni ebben a leckében; ebben nyugodtan próbálkozhatsz, miután átolvastad ezt az anyagot. A leckében foglaltakat olvasd el elejétől a végéig, mivel fokozatosan építjük fel a head elem használatának a legjobb módszereit. Minden rész önmagában is érvényes, de a végén az összefoglalóban, ahol a legjobb módszerekről beszélünk, újragondolhatod a korábbi tanácsokat.

3.2.1 Miféle head? Miről beszélsz?

Egy korábbi leírásban már olvashattál arról itt is, hogy az érvényes HTML dokumentumban meg kell adni egy doctype elemet — ez adja meg, hogy milyen típusú HTML-t fogunk használni, hogy a böngésző aztán a megfelelő szabályokat alkalmazza. Minderről bővebben a 14. leírásban fogunk beszélni, egyelőre elég annyit rögzítenünk, hogy a doctype megszabja, hogy a dokumentumnak tartalmaznia kell egy html elemet, ez pedig egy head és egy body elemet. Az időd nagy részét a body („törzs”) elemben fogod eltölteni, mivel ide kerül a dokumentum teljes tartalma. A head („fej”) elemnek látszólag kisebb szerepe van, mivel a title („cím”) elem kivételével semmi nem lesz látható a látogatók számára azok közül, amiket ide írsz. Viszont ebbe a részbe kerülnek a böngészők számára szóló utasítások, és itt tárolhatsz a dokumentumról extra információkat — más néven metaadatokat.

3.2.2 A dokumentum elsődleges nyelvének beállítása

Van egy olyan rövid információ a dokumentumról, amely nem a head elembe, hanem annak a szülőjébe, a html elembe kerül. Ez pedig nem más, mint a dokumentum természetes nyelve. A természetes nyelv alatt az emberi nyelvet értem, mégpedig hogy angol, francia vagy magyar. Ez segít a képernyő-felolvasóknak, mivel például az „eleven” szót másképp kell kiolvasni angolul vagy magyarul, de segíthet a keresőrobotoknak is. Mindig hasznos, ha megadod a dokumentum elsődleges nyelvét, különösen akkor, ha nemzetközi olvasótábornak írsz; mégsem látni sok olyan oldalt, amelyik megteszi ezt. A következőképpen állíthatod be a nyelvet:

```
<html lang="hu-HU">
  …
</html>
```

A dokumentum egyes részeinek is beállíthatsz egy nyelvet a lang attribútum használatával az adott elemeken, például <*span lang="fr">Bonjour</span*>.

Az attribútum, amiben a nyelvet megadhatod, a használt doctype-tól függ. A W3C szerint:

HTML esetében csak a lang attribútumot használjuk; a text/html formában szolgáltatott XHTML 1.0 esetében használhatjuk a lang és xml:lang attribútumokat; és az XML formában szolgáltatott XHTML esetében pedig csak az xml:lang attribútumot.

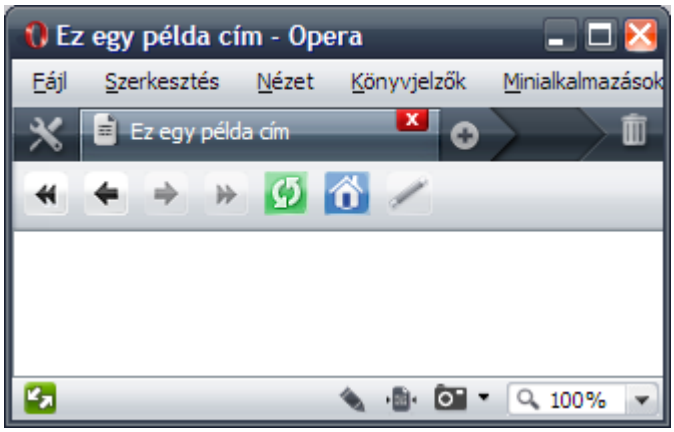
A nyelv kódja lehet egy kétbetűs kód, mint például az en az angolhoz, négybetűs kód, mint például az en-US az amerikai angolhoz, vagy más, ritkán használt kód is. A kétbetűs kódok az ISO 639-1 alatt találhatóak meg.

3.2.3 A dokumentum megítélése a címe alapján

A head egyik legfontosabb része a title elem. Gyakorlatilag az összes böngészőben és kliens eszközök többségén a title elemben található szöveg jelenik meg — a weblap neveként — az ablak fejlécében (ez a böngésző ablakát körülvevő keret felső része). Ez az első dolog, amit a látogatók általában meglátnak egy weblap megnyitásakor, ezért rendkívül fontos. Továbbá, a kisegítő megoldások, mint például a képernyő-felolvasók, ezt adják meg először a felhasználóiknak, így ebből szűrhetik le először, hogy mire számíthatnak majd az oldalon; de a keresőrobotok is nagyon hasonlóan működnek, így ha egy könnyen megérthető és a jó kulcsszavakat tartalmazó címet adsz az oldaladnak, akkor az esélyed arra, hogy a keresőkön keresztül megtaláljanak, drasztikusan megnő. Vegyük például a következő HTML dokumentumot (headexample.html a zip fájlban), és nyissuk meg egy böngészőben.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Ez egy példa cím</title>
  </head>
  <body>
  </body>
</html>
```

Ha megnyitod ezt az oldalt, láthatod, hogy a *title* elem tartalma az ablak fejlécében és navigáció fölött a fülön is megjelenik.



3.1. ábra: A title elem megjelenítése a böngészőben

Sok leírást találhatsz a weben arról, hogy hogyan adhatsz jó címet az oldalaidnak, ezek többsége a keresőoptimalizálással (SEO) kapcsolatos. Ne ess túlzásokba, amikor ezeket látod: a keresőket esetleg becsaphatod egy optimalizált címmel, és talán megszerezhetsz egy ideig egy jobb találati pozíciót, de jobban jársz, ha inkább egy rövid, informatív címet adsz a dokumentum tartalmáról. A „Kutyák tenyésztése — tippek a farkaskutyákról” sokkal jobban olvasható, mint mondjuk a „Kutyák, farkaskutyák, tenyésztés, kutya, tipp, ingyen, kedvenc”.

3.2.4 Kulcsszavak és leírás hozzáfűzése

A következő lépés első látásra talán feleslegesnek tűnhet, mivel ezek az elemek soha nem jelennek meg közvetlenül a látogató előtt: a leírás és a kulcsszavak. Mindkettőt meta elemeken keresztül adhatjuk hozzá a head elemhez, amint az az alábbi példán látszik, a Yahoo! Eurosport oldaláról (headwithmeta.html):

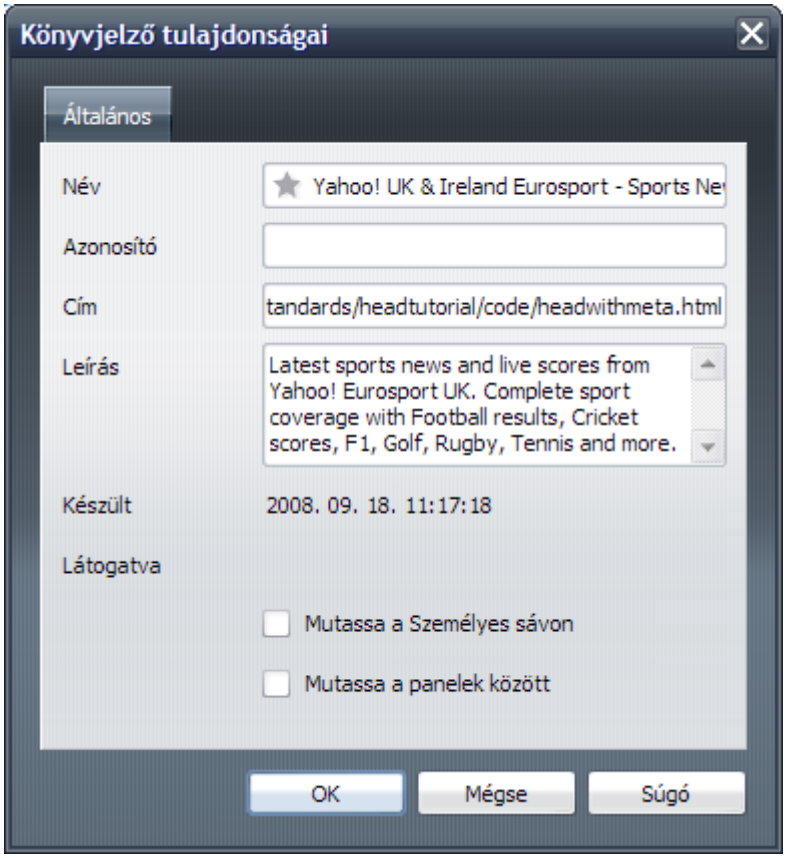
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Yahoo! UK & Ireland Eurosport—Sports News | Live
Scores | Sport</title>
    <meta name="description" content="Latest sports news and
live scores from Yahoo! Eurosport UK. Complete sport coverage
with Football results, Cricket scores, F1, Golf, Rugby, Tennis
and more.">
    <meta name="keywords" content="eurosport,sports,sport,sports
news,live
scores,football,cricket,f1,golf,rugby,tennis,uk,yahoo">
  </head>
  <body>
  </body>
</html>
```

Ha megnyitod ezt az oldalt a böngésződben, semmit nem fogsz látni az oldalon. Ha viszont feltöltöd a lapot a netre, és egy kereső beindexeli, akkor a megadott leírás megjelenik majd a hivatkozás alatt a találati listában, amint az a 3.2. ábrán látható:

Yahoo! UK & Ireland **Eurosport** - Sports News | Live Scores | Sport
Latest sports news and live scores from Yahoo! **Eurosport** UK. Complete sport coverage with Football results, Cricket scores, F1, Golf, Rugby, ...
uk.**eurosport**.yahoo.com/ - 35k - Cached - Similar pages - Note this

3.2. ábra: A leírás megjelenik a keresők találati oldalán

Ez az információ így már kritikus lehet a lehetséges látogatók szempontjából, mivel ennek alapján dönti el, hogy rákattint a találatra vagy nem. A leírásoknak van egy másik haszna is — egyes böngészőkben a leírás extra információként jelenik meg, ha a felhasználó az oldalt elmenti a kedvencek közé, ahogy az a 3.3. ábrán látható:



3.3. ábra: A leírás egyes böngészőkben akkor is megjelenik, ha az oldalt a kedvencek közzé teszed

Szóval, bár nincs azonnali haszna a meta leírás beillesztésének, mégis ennek nagyon fontos szerepe van a weblap sikeressége szempontjából. Ugyanez — bár kisebb mértékben — vonatkozik a hozzáadott kulcsszavakra is.

A szemetelőknek sokéves munkával sikerült elérniük, hogy a keresők már ne vegyék többé túl komolyan a megadott kulcsszavakat, ennek ellenére még mindig hasznosak lehetnek akkor, ha gyorsan át akarsz nézni több dokumentumot egyszerre a tartalom elolvasása és értelmezése nélkül. A meta kulcsszavakat használhatod például egy tartalomkezelő rendszerben, ahol egy script leindexeli őket, így a kereséd gyorsabban fog működni. Soha nem árthat, ha biztosítasz egy módszert az oldalaid keresésére a tartalom átvizsgálása nélkül. Ha több kulcsszavat adsz a meta elemhez, akkor lehetőséget adsz magadnak arra, hogy a jövőben egy gyors és okos keresőt készíthess az oldalaidhoz. Gondolj úgy a kulcsszavakra, mintha kis könyvjelzők lennének, amelyeket egy vastag könyvben hagysz,

hogy ezekkel gyorsan megtalálhasd, amit keresel anélkül, hogy a teljes fejezetet át kellene olvasnod.

3.2.5 Mi a helyzet a megjelenéssel? Stílusok hozzáadása

A következő dolog, amit a head elemhez hozzáadhatsz a dokumentumban, az a stílusozás, amelyet CSS (Cascading Style Sheets) segítségével valósíthatsz meg. Ezt beépítheted közvetlenül a head elembe a style elem segítségével, mint az alábbi példában (headinlinestyles.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Kutyatenyésztés - Tippek farkaskutyákról</title>
  <meta name="description" content="Hogy tenyésszünk
farkaskutyákat, tippek a helyes tenyésztésről és információk a
tenyésztéssel kapcsolatos gyakori esetekről.">
  <meta name="keywords"
content="kutyák,farkaskutya,tenyésztés,kutya,tippek,ingyen,kedve
nc">
  <style type="text/css">
    body {
      background: #000;
      color: #ccc;
      font-family: helvetica,arial,sans-serif;
    }
  </style>
</head>

<body>
Teszt!
</body>
</html>
```

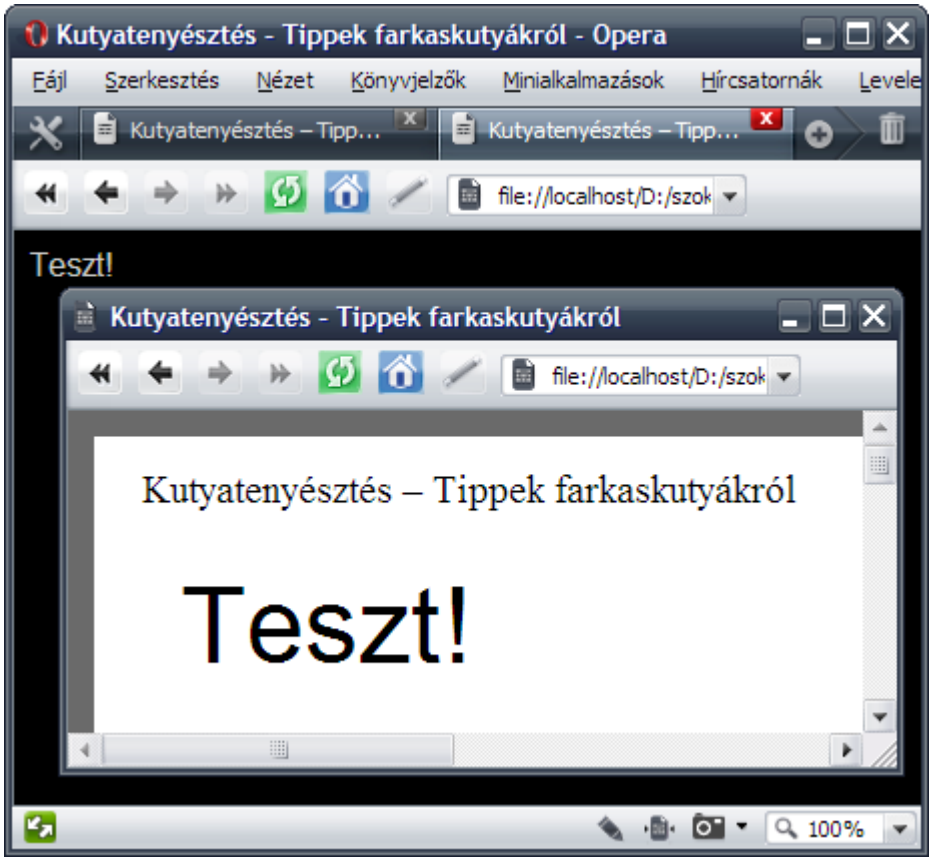
Ha ezt a kódot megnyitod a böngésződben, csak egy szürke „Teszt!” feliratot fogsz látni fekete háttéren, amelynek a betűtípusa Helvetica vagy Arial lesz, a használt rendszertől függően. A style elem tartalmazhat egy media attribútumot is, amelyben megadhatod, milyen típusú eszközön akarod ezt a stílust alkalmazni; például akkor akarod használni ezt a stílust, ha az oldal egy képernyőn jelenik meg, egy kézi eszközön vagy nyomtatáskor? Több média típus közül is választhatsz, amelyek között az alábbiak a leghasznosabbak:

- **screen** — képernyőn való megjelenítéskor használhatod.
- **print** — ebben megadhatod, hogyan nézzen ki a dokumentum nyomtatáskor.
- **handheld** — ezzel adhatod meg a dokumentum megjelenését mobil- és más kisméretű eszközökön.
- **projection** — HTML prezentáció készítéséhez használhatod, amelyet például az Opera Show is támogat.

Ha például nyomtatáskor más színeket és nagyobb betűket akarsz használni, mint a képernyőn, akkor az első style blokk után megadhatsz egy másikat, amelyben megadod a media attribútumot print értékkel, amint az alábbi kódban látható (a teljes kódot a headinlinestylesmedia.html fájlban találod):

```
<style type="text/css" media="print">
body {
    background: #fff;
    color: #000;
    font-family: helvetica, arial, sans-serif;
    font-size: 300%;
  }
</style>
```

Mostantól, ha ki akarod nyomtatni a lapot, a böngésző a print típusú stíluslapot fogja használni a dokumentumhoz a screen típusú helyett. Te is kipróbálhatod, ha megnyitod a headinlinestylesmedia.html fájlt, és megnézed a nyomtatási előnézetet. Az eredményt a 4. ábrán láthatod:



3.4. ábra: Ugyanaz a lap képernyőhöz és nyomtatáshoz való stílussal

3.2.6 Dinamikus funkciók hozzáadása JavaScripttel

Egy másik dolog, amelyet a head elemhez hozzáadhatsz, az a böngésző által futtatható scriptek — más szóval „kliens oldali scriptek” —, amelyeket JavaScriptben írhatsz meg. Ahogy már a 4. leírásban olvashattad, a JavaScript dinamikus működést ad a statikus HTML dokumentumokhoz, például animációs effekteket, adatellenőrzést vagy más olyan dolgokat, amelyeket bizonyos felhasználói műveletek válthatnak ki.

A JavaScriptet a script taggel adhatod hozzá az oldaladhoz. Amikor a böngésző találkozik egy ilyen elemmel, azonnal megállítja a dokumentum feldolgozását, és a feldolgozás folytatása előtt megpróbálja végrehajtani az elemben található kódot. Így aztán ha azt szeretnéd, hogy a JavaScripted még a dokumentum betöltése előtt lefusson, akkor a

head elemben kell megadnod. Az alábbi scripttel például figyelmeztetheted a felhasználót, hogy egy bizonyos hivatkozás egy másik kiszolgálóra fogja vinni (headscript.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Kutyatenyésztés – Tippek farkaskutyákról</title>
  <meta name="description" content="Hogy tenyésszünk
farkaskutyákat, tippek a helyes tenyésztésről és információk a
tenyésztéssel kapcsolatos gyakori esetekről.">
  <meta name="keywords"
content="kutyák,farkaskutya,tenyésztés,kutya,tippek,ingyen,kedve
nc">
  <style type="text/css" media="screen">
    body {
      background: #000;
      color: #ccc;
      font-family: helvetica,arial,sans-serif;
    }
    a { color: #fff }
  </style>
  <style type="text/css" media="print">
    body {
      background: #fff;
      color: #000;
      font-family: helvetica,arial,sans-serif;
      font-size: 300%;
    }
  </style>
  <script>
    function leave() {
      return confirm("Ezzel átugrasz egy másik oldalra,\n
biztos, hogy ezt akarod?")
    }
  </script>
</head>
<body>
Teszt!
<a href="http://cukisag.blog.hu" onclick="return
leave()">Cukiság blog</a>
</body>
</html>
```

Ha a fenti példát megnyitod a böngésződben, és rákattintasz a hivatkozásra, akkor felugrik egy ablak, amelyben meg kell erősítened a műveletet. Ez csak egy gyors példa volt a scriptekre, és nagyon távol áll a manapság használt legjobb módszerektől. A későbbi leírásokban fogunk még foglalkozni a JavaScript módszerekkel, és részletesebben is fogunk beszélni a JavaScript technikákról, egyelőre nem kell vele túl sokat foglalkoznod.

3.2.7 Állj! A beágyazott CSS és JavaScript nem jó ötlet!

Kemény szavak, tudom, de egy dolgot mindenképpen az eszedbe kell vésned, mielőtt weboldalakat írnál: próbáld meg a kódodat minél jobban újrafelhasználhatóvá tenni. Ha

site jellegű stílusokat adsz hozzá minden egyes lapodhoz külön-külön, annak egyrészt nincs sok értelme — ráadásul sokkal nehezebb lesz a site-ot karbantartani, és feleslegesen lesznek nagyobbak a dokumentumaid.

Sokkal jobb, ha a stílusokat és a scripteket külső fájlokba teszed, és a HTML fájlokban csak betöltöd ezeket, ahol szükséges, így ha változtatni akarsz rajtuk valamit, elég egy helyen megtenned. JavaScript esetében ezt továbbra is a script elemmel teheted meg, mégpedig úgy, hogy nem írsz kódot az elemen belül, és a link attribútum helyett az src attribútumot használod, ahogy az alábbi példában tettük (externaljs.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Kutyatenyésztés - Tippek farkaskutyákról</title>
  <meta name="description" content="Hogy tenyésszünk
farkaskutyákat, tippek a helyes tenyésztésről és információk a
tenyésztéssel kapcsolatos gyakori esetekről.">
  <meta name="keywords"
content="kutyák,farkaskutya,tenyésztés,kutya,tippek,ingyen,kedve
nc">
  <style type="text/css" media="screen">
    body {
      background: #000;
      color: #ccc;
      font-family: helvetica,arial,sans-serif;
    }
    a { color: #fff }
  </style>
  <style type="text/css" media="print">
    body {
      background: #fff;
      color: #000;
      font-family: helvetica,arial,sans-serif;
      font-size: 300%;
    }
  </style>
  <script src="leaving.js"></script>
</head>
<body>
Teszt!
<a href="http://cukisag.blog.hu" onclick="return
leave()">Cukiság blog</a>
</body>
</html>
```

CSS esetében már nem ilyen egyszerű a dolog. A style elemnek ugyanis nincs src attribútuma, így helyette a link elemet kell használod. Ebben megadhatsz egy külső css fájlt importálásra a href attribútum használatával, majd a media attribútummal — ugyanúgy, mint korábban — megadhatod, hogy milyen eszközön akarod használni ezt a stílust: képernyőn, nyomtatáskor, stb. Ha a CSS-t és a JavaScriptet is külső fájlban adod meg, akkor jelentősen csökkentetted a head elem méretét, ami az alábbi példán is jól látszik (externalall.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Kutyatenyésztés - Tippek farkaskutyákról</title>
  <meta name="description" content="Hogy tenyésszünk
farkaskutyákat, tippek a helyes tenyésztésről és információk a
tenyésztéssel kapcsolatos gyakori esetekről.">
  <meta name="keywords"
content="kutyák,farkaskutya,tenyésztés,kutya,tippek,ingyen,kedve
nc">
  <link rel="stylesheet" type="text/css" media="screen"
href="styles.css">
  <link rel="stylesheet" type="text/css" media="print"
href="printstyles.css">
  <script src="leaving.js"></script>
</head>

<body>
Teszt!
<a href="http://cukisag.blog.hu" onclick="return
leave()">Cukiság blog</a>
</body>
</html>
```

További előnyei is vannak, ha a stílusaidat és a scripteket külön fájlokban tartod:

1. Gyorsabbá teszed vele a letöltést a látogatóid számára, mert bár le kell tölteniük néhány külön fájlt a dokumentum mellé, de ezt csak egyszer kell megtenniük, mivel ezeket a különböző lapokon újra felhasználhatod, ezáltal összességében kevesebbet kell letölteni. Ráadásul a CSS és a JavaScript fájlok többnyire bekerülnek a gyorsítótárba, így amikor legközelebb látogatják meg az oldalt, a fájlok már ott lesznek a gépen, és nem kell újra letölteni őket.
2. Könnyebb lesz a karbantartás és a javítás. Ha a stílusok és a scriptek az egész sitehoz — ami akár több ezer dokumentumból is állhat — egyetlen helyen találhatóak, akkor a módosításokat elég egyetlen fájlban elvégezned, és nem kell hozzányúlj az akár több ezernyi dokumentumhoz.

3.2.8 Tesztkérdések

Szokás szerint az alábbi kérdésekkel ellenőrizheted, hogy sikerült-e megértened a témát.

- Miért hasznos megadni a dokumentum leírását a meta elemben, amikor az úgysem jelenik meg a képernyőn?
- Milyen előnye van annak, ha a JavaScriptet a head elemben adod meg, és nem a body-ban?
- Hogyan használhatod ki a böngésző gyorsítótárát, és mit kell tenned ezért?
- Mivel a keresők előnyben részesítik a dokumentum címét, nem lenne jobb, ha a címben adjuk meg a kulcsszavakat? Mi ennek a módszernek a hátránya?
- Mivel az oldal címe sokszor unalmas, nem lenne jó kiemelni benne egyes szavakat a b elemmel? Lehetséges ez?

3.3. Megfelelő doctype választása a HTML dokumentumokhoz

Ebben a leírásban részletesebben megnézzük a doctype elemet, megmutatjuk, mire szolgál, hogyan segíthet a HTML validálásában, hogyan választható ki a megfelelő doctype típus a dokumentumhoz, és megnézzük az XML deklarációt, amelyre ritkán ugyan, de szükséged lehet.

3.3.1 Első a doctype

A legelső dolog, amit minden HTML dokumentumban definiálnod kell, az egy DTD deklaráció. Ha még soha senkitől nem hallottál a DTD deklarációról, ne izgulj. A dolgok egyszerűsítése érdekében gyakran úgy hivatkoznak erre, hogy „doctype” — én is így fogom hívni a leírás további részében.

Talán elgondolkodtál azon, hogy mi lehet ez a „DTD” vagy „doctype”. A DTD a „Document Type Definition”, azaz a dokumentum típus definíció rövidítése, és sok más dolog mellett ez adja meg azt, hogy milyen elemeket és attribútumokat használhatsz a HTML egyes részeiben — bizony, a mai weben több különböző HTML verzió is használatban van, de ezen ne aggódj egyelőre, végül is csak az egyikkel kell majd közülük foglalkoznod.

A doctype két dologra használható, a szóban forgó szoftvertől függően:

1. A webböngészők arra használják, hogy meghatározzák a renderelési módot a weblap megjelenítéséhez (ezekről később még lesz szó).
2. A jelölés validátorok a doctype alapján határozzák meg, hogy milyen szabályokat kell ellenőrizniük a dokumentumban (erről is lesz még szó bővebben).

Mindkét megközelítés fontos lesz a számodra, de különböző módokon, ezeket fogjuk a leírás további részében bővebben megmagyarázni.

Itt egy példa:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Ez így most elsőre teljesen értelmetlennek tűnhet a számodra, úgyhogy engedd meg, hogy elmagyarázzam a lehető legegyszerűbben, hogy hogyan épül fel a fenti elem. Ha kíváncsi vagy egy sokkal részletesebb magyarázatra, amelyben minden egyes karakter funkcióját megmagyarázzák, akkor olvasd el a !DOCTYPE cikket (angolul).

A doctype legfontosabb részei az idézőjelekkel határolt két darab string. A "-//W3C//DTD HTML 4.01//EN" azt jelöli, hogy ezt a DTD dokumentumot a W3C adta ki, a DTD a HTML 4.01-es verzióját írja le, és hogy a DTD-ben használt nyelv az angol (EN).

A második string, a "http://www.w3.org/TR/html4/strict.dtd" egy URL, amely a felhasznált DTD dokumentumra mutat.

Bár a doctype elég különösnek tűnik, szükség van rá a HTML és XHTML specifikációkhoz. Ha nem adod meg, akkor biztosan validálási hibát fogsz kapni, ha ellenőrzöd a dokumentumod jelöléseinek érvényességét a W3C validátorával, vagy más olyan eszközzel, amely HTML dokumentumokban keres hibákat. Egyes böngészőkben ez a funkció alapból benne van, míg más böngészőkben egy kiterjesztéssel lehet telepíteni.

3.3.2 Doctype sniffing és renderelési módok

Ha nem adod meg a doctype elemet, a böngészők ebben az esetben is megpróbálják értelmezni és feldolgozni a dokumentumot — az összes értelmetlen zagyvaságot is meg kell próbálniuk feldolgozni, amit a weben csak találni lehet, szóval nem lehetnek túlságosan válogatósak. Viszont könnyen lehet, hogy doctype nélkül nem olyan eredményt fogsz kapni, mint amilyenre számítottál, mégpedig a „doctype sniffing", vagyis a doctype „kiszagolása" miatt.

A 21. századi böngészők túlnyomó része minden megnyitott HTML dokumentumnál megpróbálják a doctype segítségével eldönteni, hogy a dokumentum szerzője vajon figyelembe vette-e a webes szabványokat a HTML és a CSS megírásakor, vagy nem foglalkozott ezekkel.

Ha olyan doctype-ot találnak, amely szerint a dokumentum helyesen van kódolva, akkor egy ún. „szabványos módot" fognak használni a lap megjelenítésekor. Szabványos módban a böngészők igyekeznek a lapot a CSS specifikáció szerint elrendezni — ilyenkor ugyanis megbíznak a fejlesztőben, és feltételezik, hogy az oldal készítésekor tudta, hogy mikor mit csinál.

Másrészről, ha egy elavult vagy helytelen doctype-ot találnak, akkor átváltanak egy ún. „Quirks módba", amely sokkal inkább kompatibilis a régi módszerekkel és a régi böngészőkkel. A Quirks mód azt feltételezi, hogy a dokumentum régi, és nem a jelenlegi webes szabványok szerint készült — a weblap így is meg fog jelenni, de jóval több számítási kapacitást igényelhet, és nagyobb valószínűséggel kapsz furcsa vagy ronda megjelenést, amikor nem is számítasz rá.

A különbség elsősorban abban van, hogy a böngésző hogyan értelmezi a CSS-t, és csak néhány esetben vonatkozik a konkrét HTML értelmezésére. Webfejlesztőként vagy webdesignerként akkor kaphatod az összes böngészőben a leginkább megegyező eredményeket, ha megbizonyosodsz róla, hogy minden böngésző a szabványos módot használja a megjelenítéshez, és megfelelő doctype-ot használsz!

3.3.3 Validálás

Amint azt korábban is említettem, a doctype-ot a validátorok is felhasználják; erről többet is meg fogsz tudni a sorozat későbbi részeiből. Jelenleg elég, ha annyit tudsz, hogy a validátor ellenőrzi a HTML dokumentum szintakszisát, és megnézi, hogy van-e benne valamilyen hiba. A validátor pedig a doctype alapján dönti el, hogy milyen szabályokat kell ellenőriznie a dokumentumban. Ez hasonló ahhoz, mint amikor a helyesírás-ellenőrzőnek megmondod, hogy milyen nyelven van megírva a dokumentum. Ha nem mondod meg neki, akkor nem fogja tudni azt sem, hogy milyen helyesírási és nyelvtani szabályokat kellene ellenőriznie.

3.3.4 Doctype választása

Szóval, most már tudod, hogy meg kell adnod egy doctype-ot, és azt is tudod, hogy ez mire szolgál, de honnan fogod tudni azt, hogy melyiket használd? Valójában elég sok van belőlük. Sőt, ha valami igazán komoly dolgot csinálsz, még saját doctype-ot is készíthetsz magadnak. De most nem fogunk belemenni a különböző fajtájú doctype-okba, hanem megpróbálom számodra a dolgokat egyszerű és érthető szinten tartani.

Ha a dokumentumod HTML, használd ezt:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Ha a dokumentumod XHTML, használd ezt:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Megjegyzés: a „valódi” XHTML-t a webböngészőnek XML-ként kell küldeni, de ennek a részletei, hogy mikor és hogyan teheted ezt meg, valamint hogy ez milyen változásokat okoz, túlmutatnak a jelenlegi cikk keretein.

Mindkét doctype használatával a böngésző a szabványos módot fogja használni a dokumentum értelmezésekor. A leglátványosabb hatás, amit a dokumentum készítésekor láthatsz, miután beállítottad a doctpye-ot, hogy sokkal következetesebbek lesznek az CSS változtatása után az eredményeid. Ha más lehetséges doctpye-okra is kíváncsi vagy, amelyeket a dokumentumodhoz használhatsz, akkor nézd meg a W3C listáját a javasolt DTD típusokkal.

Talán észrevetted, hogy mindkét doctype, amelyet itt javasoltam, az „strict” (vagyis „szigorú”). Ez talán ijesztően hangzik a számodra, de nem az.

Mind a HTML-nek, mind az XHTML-nek vannak szigorú (strict) és átmeneti (transitional) változatai is. A „szigorú” jelen esetben azt jelenti, hogy a doctype kevesebb megjelenítési jelölést engedélyez, mint az átmeneti változat. Viszont azoknak a megjelenítési jelöléseknek, amelyek nem engedélyezettek, amúgy sincs semmi keresnivalójuk a dokumentumban, mivel a HTML-t a struktúra és a tartalom meghatározására használhatod, és a CSS-ben definiálhatod ezeknek a megjelenését. A szigorú doctype segít neked ebben, mivel a validátor figyelmeztetni fog, ha véletlenül talál egy becsúszott megjelenítési elemet vagy attribútumot a kódodban.

3.3.5 XML deklaráció

Korábban már említettem, hogy a doctype kell legyen a legelső elem a HTML dokumentumokban. Nos, ez valójában csak egy egyszerűsített változata a valóságnak. Meg kell még említenünk az XML deklarációt is.

Egyes XHTML dokumentumokban a következő kódrészletet láthatod közvetlenül a doctpye előtt:

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

Ezt XML deklarációnak hívják, és ha jelen van, akkor mindig a doctype előtt kell legyen.

Az Internet Explorer 6-os verziójának van ezzel egy kis problémája: ennek hatására ugyanis átvált Quirks módba, ami — mint azt már korábban említettem — nem igazán kívánatos.

Szerencsére az XML deklaráció az esetek többségében nem kötelező, kivéve abban az esetben, ha az XHTML dokumentumaidat valóban XML formában küldöd a böngészőknek (lásd a megjegyzést az XHTML-nél), *és* más karakterkódolást használsz, mint az UTF-8, *és* a kiszolgálód nem küld egy HTTP fejlécet, amely megadná a karakterkódolást.

Annak a valószínűsége, hogy ez mind egyszerre bekövetkezik, eléggé elenyésző, így a legegyszerűbb mód arra, hogy megoldhasd az Internet Explorer problémáját, az az, hogy egyszerűen kihagyod az XML deklarációt. Azért nehogy elfelejtsd a doctype-ot!

3.3.6 Tesztkérdések

Íme néhány kérdés, amelyeket a leírás elolvasása után meg kell tudnod válaszolni:

- Mi a két legfontosabb célja a doctype beillesztésének a HTML dokumentumokba?
- Mik az előnyei annak, ha „szigorú” (strict) doctype-ot használsz az átmeneti (transitional) helyett?
- Miért problémás az XML deklaráció?
- Ebben a leírásban nem említettem a frameset doctype-ot — nézz utána, hogy ez mire jó, és miért kell elkerülni.

4. A HTML felépítése

4.1. Szöveges részek megjelölése HTML-ben

Ebben a leírásban megismerkedhetsz a HTML alapszintű használatával, mégpedig hogy hogyan adhatod meg a jelölésekkel a dokumentumod különböző részeinek az értelmét.

Megnézzük az általános strukturális elemeket, mint például a címsorok, a paragrafusok vagy a beágyazott idézetek és kódok. Ezután néhány belső tartalommal is megismerkedünk, mint a rövid idézetek és a kiemelések, majd a végén még gyorsan átnézünk néhány régimódi megjelenítési elemet is.

4.1.1 Az űr a legvégső határ

Van egy nagyon fontos pont, amit tisztáznunk kell, mielőtt még a szövegekről beszélnénk, mégpedig a szavak közötti üres helyekről. A HTML írásakor a forráskód ún. „fehér karaktereket" is tartalmaz — ezek azok a karakterek, amelyek egy üres területtel elválasztják egymástól a szavakat. A leggyakoribb ilyen fehér karakter a space (szóköz), amelyet a szóköz billentyű lenyomásával tudsz beírni, de ilyen még a tabulátor és az új sor karakter is.

A HTML-ben a fehér karakterek többszörös megjelenése (szinte) minden esetben egyetlen szóköz karakternek számít. Például:

```
<h2>Kezdetben    teremté
                az      Úr </h2>
```

a böngészőben ugyanúgy lesz értelmezve, mint a következő:

```
<h2>Kezdetben teremté az Úr</h2>
```

Az egyetlen hely, ahol ez nem így történik, az a *pre* elem, amelyről még fogunk beszélni ebben a leírásban.

Ez sokszor zavaró lehet az olyanoknak, akik először szerkesztenek HTML dokumentumot, mikor megpróbálnak egy szöveget kitolni néhány extra szóköz karakterrel, vagy nagyobb helyet szeretnének a mondatok, esetleg a bekezdések között. A dokumentum vizuális megjelenését viszont nem a HTML forrásban kell befolyásolni, hanem használj helyette stíluslapokat, ahogy azt a sorozat egy későbbi részében be fogjuk mutatni.

4.1.2 Blokk szintű elemek

Ebben a részben végigvesszük a gyakoribb, szövegek formázására használható blokk szintű elemek(3.1.6. fejezet) szintaxisát és használatát.

A lap szakaszcímei

Miután a lapot felosztottad több logikus szakaszra, minden szakaszt be kell vezetned egy megfelelő címsorral. Erről már volt szó korábban a 2.2. fejezetben.

A HTML 6 címsor szintet definiál, *h1*, *h2*, *h3*, *h4*, *h5* és *h6* (a magasabb fontosságútól az alacsonyabbig). Általánosságban, a *h1* szokott lenni az egész lap fő címsora, ez vezet be mindent. A *h2* ezután szakaszokra bontja a lapot, a *h3* alszakaszokra, és így tovább.

Nagyon fontos, hogy a dokumentumot a különböző szintű címsorokkal oszd fel szakaszokra, alszakaszokra, al-alszakaszokra, mivel ez sokkal érthetőbbé teszi a lapot a képernyőolvasók és az automatikus folyamatok számára (mint amilyen például a Google indexelő robotja).

Ha ezt a fejezetet vesszük alapul, akkor ez is egy jó példa a címsorok felépítésére:

```
<h1>Szöveges részek megjelölése HTML-ben</h1>
  <h2>Bevezető</h2>
  <h2>Space — az űr a legvégső határ</h2>
  <h2>Blokk szintű elemek</h2>
    <h3>A lap szakaszcímei</h3>
    <h3>Általános bekezdések</h3>
    <h3>Más források idézése</h3>
    <h3>Előformázott szöveg</h3>
  <h2>Inline elemek</h2>
  ...
```

Általános bekezdések

A bekezdések (vagy paragrafusok) a legtöbb dokumentum építőkövei. A HTML-ben a bekezdést a p elem jelöli, amelynek nincsenek speciális attribútumai. Például:

```
<p>Ez egy nagyon rövid bekezdés. Csak két mondatból áll.</p>
```

Sok könyvben és cikkben egy paragrafusban csak egy mondat lehet. Bár a „paragrafus” értelme (az írott terminológiát tekintve) elég nyilvánvaló, a weben sokszor sokkal rövidebb szövegeket is meg szoktak jelölni paragrafusként, mivel a szerző szerint ez sokkal „szemantikusabb”, mint a div elem használata (erről fogunk beszélni a későbbiekben az Általános tárolók leírásban).

A paragrafus a weben egy vagy több mondat csoportja, éppen úgy, mint az újságokban vagy a könyvekben. A webes sokkal jobb, ha ezekhez használjuk a paragrafus elemet, és nem hagyunk a szövegben egyedül csak úgy szövegrészeket. Ha csak néhány szóról van szó, vagy nem egy teljes mondatról, akkor ezeket nem kell feltétlenül paragrafusként megjelölnöd.

Más források idézése

Nagyon gyakran a cikkek, blogbejegyzések vagy a webes dokumentumok idéznek kisebb-nagyobb részeket más dokumentumokból. A HTML-ben a hosszabb idézeteket — például teljes mondatokat, bekezdéseket, listákat — a *blockquote* elemmel jelölhetjük.

A *blockquote* elem nem tartalmazhat sima szöveget, hanem csak egy másik blokk szintű elem lehet benne. Ugyanazokat a blokk szintű elemeket használd az idézetben is, mint az eredeti környezetben. Ha egy bekezdést idézel, használd a paragrafus elemet, ha egy lista elemeit, akkor készíts listát, és így tovább.

Ha az idézet egy másik weboldalról származik, akkor megadhatod a forrást a *cite* attribútumban a következőképpen:

```
<p>A HTML 4.01 az egyetlen olyan HTML verzió, amelyet egy új
weblap létrehozásakor használhatsz, a specifikáció szerint:</p>
<blockquote cite="http://www.w3.org/TR/html401/">
  <p>Ez a dokumentum érvényteleníti a HTML 4.0 korábbi verzióit,
    bár a W3C továbbra is elérhetővé teszi ezeket a
    specifikációkat és a hozzájuk tartozó DTD-ket a W3C
    weboldalán.</p>
</blockquote>
```

Előformázott szöveg

Minden olyan szöveget, amelyben a szöveg behúzása és a fehér karakterek (amelyeket a leírás elején említettünk) fontos szerepet kapnak, a pre elemmel kell megjelölnünk.

A legtöbb webböngészőben az előformázott szöveg pontosan úgy jelenik meg a felhasználónak, ahogy a forrásban szerepel, gyakran fix szélességű (monospace) betűtípussal, így a szöveg úgy néz ki, mintha írógépből származna. Ez a programozók hagyatéka, akik fix szélességű betűkkel dolgoznak, és eleinte leginkább ők használták az előformázott szöveget.

Ebben a példában egy példakódot láthatsz a perl programozási nyelvből:

```
<pre><code class="language-perl">
            # vegig beolvassa a megnevezett fajlt
            sub slurp {
                my $filename = shift;
                my $file     = new FileHandle $filename;
                if ( defined $file ) {
                    local $/;
                    return <$file>;
                }
                return undef;
            };
        </code></pre>
```

A fenti példában szereplő code elemről részletesebben is fogunk beszélni a Kevéssé ismert szemantikus elemek részben.

4.1.3 Inline elemek

Ebben a részben átnézzük a leggyakoribb, szövegek formázásához használt inline (soron belüli) elemek(3.1.6. fejezet) szintaxisát és használatát.

Rövid idézetek

A mondaton vagy bekezdésen belüli rövid idézeteket a *q* elemmel jelölhetjük. Hasonlóan a *blockquote* elemhez, ez is tartalmazhat egy *cite* attribútumot, amelyben megadhatsz egy webcímet, ahonnan az idézet származik.

A rövid idézetek legtöbbször idézőjelben szerepelnek. A HTML specifikáció szerint az idézőjeleket ebben az esetben a kliens eszköz (vagyis a böngésző) kell elhelyezze a dokumentumban, a dokumentum által beállított nyelv alapján. A CSS-ben módosíthatod a felhasznált idézőjeleket — erről egy későbbi cikkben lesz szó, a Szöveg stílusozása CSS-sel leírásban.

Egy példa a *q* elem használatára:

```
<p>Ez nem végződött túl jól a számomra. Hát igen,
   <q lang="fr">c'est la vie</q>, ahogy a franciák mondják.</p>
```

Kiemelés

A HTML-ben két módszer van arra, ha egy szövegrészt ki akarunk emelni az olvasónak, például hibákat, figyelmeztetéseket vagy megjegyzéseket. A vizuális böngészőkben ezt egy eltérő színt vagy betűtípust, esetleg félkövér vagy dőlt megjelenést jelent. A képernyő-felolvasók használói számára a kiemelés eredmény egy más hangtónus vagy egyéb hangjelzés lehet.

Az olyan szövegre, amelyet ki akarsz emelni, használhatod az em elemet, mint az alábbi példában:

```
<p><em>Megjegyzés:</em> a bojlert ki kell húzni éjszakára. </p>
```

Ha a teljes bekezdést ki kell emelned, de a bekezdésen belül is van olyan rész, amelyet ismét kiemelnél, akkor használd a strong elemet, amely egy erősebb kiemelést jelent, hasonlóan az alábbi példához:

```
<p><em>Megjegyzés: a bojlert ki <strong>kell</strong> húzni
minden éjszaka, különben felrobban - <strong>és mind meghalunk</
strong></em>.</p>
```

Dőlt kiemelés

Az általános vélekedés szerint a dőlt kiemelés nem a szöveg értelmét jelöli, ilyenformán az *i* elem használata nem ajánlott (hasonlóan a leírás következő szakaszában bemutatott megjelenítési elemekhez).

Van viszont néhány olyan helyzet, amikor a tartalom dőlt kiemelése vitathatóan bár, de helyes lehet. Vannak olyan esetek, amikor egy kifejezés dőlt kiemelése a legegyszerűbb módszer ahelyett, hogy egy egyébként máshol nem használt speciális elemet vezetnénk be rá. Ilyenek lehetnek például a hajók nevei, a tévésorozatok, könyvek vagy filmek címei, technológiai fogalmak és más rendszertani megnevezések.

Az érvelés szerint a dőlt kiemelés ebben a helyzetben azt jelenti, hogy a megjelölt szöveg különleges, és a környezet mutatja meg, hogy mennyire különleges a többi részhez viszonyítva. Valóban, a HTML 5 specifikáció vázlatában jelenleg ez áll:

Az i elem olyan szövegrészt jelöl, amelyet más hangon vagy hangszínnel mondunk, vagy egyéb módon eltér a normál kiejtéstől [...] Az i elemet csak legvégső esetben használjuk, amikor más semmilyen más elem nem megfelelő.

Mivel az *i* elem megjelenését is módosítani lehet CSS-ben, hogy ne legyen dőlt, így a „dőlt kiemelés” jelentése ebben a környezetben csak annyi, hogy „valami más”. Én ezt személy szerint nem tartom elfogadhatónak, de már éppen elég precedens van a használatára ebben a formában.

4.1.4 Megjelenítési elemek — soha ne használd őket

A HTML specifikáció tartalmaz néhány olyan elemet is, amelyet általában „megjelenítési” elemnek neveznek, mivel kizárólag arra vonatkoznak, hogy a megjelölt szövegrész hogyan jelenjen meg, és nem arra, hogy mit jelent.

A legtöbb ilyen elemet már érvénytelenként jelölték meg a specifikációban. Ez azt jelenti, hogy van egy más, újabb módszer is, amellyel ugyanazt az eredményt érhetjük el.

Most részletesebben bemutatjuk ezeket az elemeket, de ennek főleg történelmi okai vannak — a modern weblapokon soha ne használd őket. Ezeknek az elemeknek a hatását más módon is elérheted; erről később még fogunk beszélni a Szöveg stílusozása CSS-sel és a Kevéssé ismert szemantikus elemek részekben.

font face="..." size="..."

A közrefogott szöveget a böngésző egy másik betűtípussal jeleníti meg — viszont a betűtípusokat inkább a CSS-ben add meg.

b

A megjelölt szöveg félkövér — ez az esetek többségében azt jelenti, hogy ki akarod emelni a szöveget, úgyhogy használd helyette az em vagy a strong elemeket, amelyekről már beszéltünk korábban.

s és strike

A megjelölt szöveget áthúzza egy vonallal — ha ez csak egy megjelenítési hatás, akkor inkább CSS-ben add meg. Ha viszont azt jelöli, hogy a szöveg törölve lett a dokumentumból, vagy már nem érvényes, akkor használhatod helyette a del elemet, amelyről még később itt is lesz szó.

u

A közrefogott szöveget aláhúzza — ez szinte mindig egy vizuális hatás, így inkább CSS-ben valósítsd meg.

tt

A jelölt szöveg „írógéppel” készült, vagy fix szélességű betűtípussal jelenik meg — ezt megoldhatod CSS-sel, vagy egy pontosabb szemantikus jelöléssel, mint például a fent bemutatott pre elemmel.

big és small

A megjelölt szöveg kisebb vagy nagyobb lesz — ezt inkább CSS-ben valósítsd meg.

4.2. HTML listák

A listákat arra használjuk, hogy az egymáshoz kapcsolódó információkat egy csoportba gyűjtsük össze, így ezek világosan kapcsolódnak egymáshoz, ezáltal könnyen olvashatóak. A modern webfejlesztésben a listák gyakori igáslovak, sokszor használják például a navigációban, de az általános tartalmakban is.

A listák strukturális szempontból is kiválóak, mivel a segítségükkel egy könnyen kezelhető, könnyen hozzáférhető és jól strukturált dokumentumot hozhatunk létre. Ezen kívül a listák rendkívül praktikusak is — az extra elemek segítenek abban, hogy különböző CSS stílusokat kapcsolhass hozzájuk (a CSS-ről a tanfolyam egy későbbi részében fogunk

beszélni — először nézd meg a Listák és hivatkozások stílusozása leírást, majd a Tartalomjegyzékben megtalálod a többi CSS-ről szóló leírást is).

Ebben a leírásban megnézzük, hogy milyen listatípusokat érhetünk el a HTML-ben, mikor és hogyan használhatjuk ezeket, és hogyan rendelhetünk hozzájuk pár egyszerű stílust.

4.2.1 A listák három típusa

A HTML-ben három típusú lista van:

- rendezetlen lista — egymással kapcsolatban álló elemek csoportosítására, ha nem számít a sorrend.
- rendezett lista — egymással kapcsolatban álló elemek csoportosítására, ha számít a sorrend.
- definíciós lista — név/érték párok megjelenítésére szolgál, mint például kifejezések és azok definíciói, vagy időpontok a hozzájuk tartozó eseménnyel.

Mindegyiknek meghatározott célja és értelme van — a különböző típusú listákat nem lehet egymás között felcserélni.

Rendezetlen lista

A rendezetlen vagy pontozott listákat akkor használják, ha a lista elemeinek a sorrendje nem fontos. Ilyen például egy bevásárlólista:

- tej
- kenyér
- vaj
- kávé

Ezek az elemek mind ugyanannak a listának a részei, viszont bármilyen sorrendben írhatod őket, a lista ettől még nem változik:

- kenyér
- kávé
- tej
- vaj

A CSS-ben lecserélheted a pontokat néhány más beépített típusra, vagy használhatsz saját képeket is, de akár pontok nélkül is megjelenítheted a listát — erről egy kicsivel később fogunk beszélni ebben a leírásban, és egy későbbi leírásban részletesebben is tárgyaljuk.

Rendezetlen lista jelölése

A rendezetlen listákat az *
* tagek jelölik, amelyen belül az elemeket az *
* tagekkel fogjuk közre:

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

Rendezett lista

A rendezett vagy számozott listákat akkor használják, amikor a lista elemei egy meghatározott sorrendben követik egymást. Jó példa erre egy recept utasításainak a listája, amelyeket szigorúan egymás után kell végrehajtanunk, ha azt akarjuk, hogy a recept működjön:

1. Gyűjtsük be a szükséges hozzávalókat
2. Keverjük össze a hozzávalókat
3. Tegyük be a keveréket egy sütőedénybe
4. Süssük a sütőben egy órán keresztül
5. Vegyük ki a sütőből
6. Hagyjuk hűlni tíz percig
7. Szolgáljuk fel

Ha a lista elemeit átrendezzük valamilyen más sorrendbe, akkor az információ értelmét veszti:

1. Gyűjtsük be a szükséges hozzávalókat
2. Süssük a sütőben egy órán keresztül
3. Vegyük ki a sütőből
4. Szolgáljuk fel
5. Tegyük be a keveréket egy sütőedénybe
6. Hagyjuk hűlni tíz percig
7. Keverjük össze a hozzávalókat

A rendezett listákat többféle számozással és alfabetikus rendszerrel is jelölhetjük — egyszóval számokkal és betűkkel. Az alapértelmezett jelölés a böngészőkben a decimális számozás, de további lehetőségek is vannak:

- Betűk
  - ASCII kisbetűk (a, b, c…)
  - ASCII nagybetűk (A, B, C…).
  - Klasszikus kis görög betűk: (?, ß, ?…)
- Számok
  - Decimális számok (1, 2, 3…)
  - Decimális számok nullával kiegészítve (01, 02, 03…)
  - Kisbetűs római számok (i, ii, iii…)
  - Nagybetűs római számok (I, II, III…)

- Tradicionális grúz számozás (an, ban, gan…)
- Tradicionális örmény számozás (mek, yerku, yerek…)

A lista stílusát ebben az esetben is CSS-sel változtathatod meg, ha szükséges.

Rendezett lista jelölése

A rendezett listákat az *
* tagek jelölik, amelyen belül az elemeket az *
* tagekkel fogjuk közre:

```
<ol>
  <li>Gyűjtsük be a szükséges hozzávalókat</li>
  <li>Keverjük össze a hozzávalókat</li>
  <li>Tegyük be a keveréket egy sütőedénybe</li>
  <li>Süssük a sütőben egy órán keresztül</li>
  <li>Vegyük ki a sütőből</li>
  <li>Hagyjuk hűlni tíz percig</li>
  <li>Szolgáljuk fel</li>
</ol>
```

Rendezett lista kezdése 1-től eltérő számmal

Lehetőség van arra is, hogy egy rendezett lista ne 1-gyel kezdődjön (vagy i-vel, I-vel, stb.). Ezt a start attribútummal érhetjük el, amelyikben egy numerikus értéket kell megadnunk (abban az esetben is, ha a CSS-ben megváltoztattuk a lista típusát alfabetikus karakterekre vagy római számokra a *list-style-type* tulajdonsággal). Ez akkor hasznos, ha van egy összefüggő listád, amelyet közben meg akarsz szakítani egy megjegyzés vagy más kapcsolódó információ beszúrásával. Például a fenti kódot így módosíthatjuk:

```
<ol>
  <li>Gyűjtsük be a szükséges hozzávalókat</li>
  <li>Keverjük össze a hozzávalókat</li>
  <li>Tegyük be a keveréket egy sütőedénybe</li>
</ol>

<p class="note">Mielőtt betesszük a hozzávalókat az edénybe,
melegítsük fel a sütőt 180 Celsius fokra/350 Farenheit fokra,
hogy készen álljunk a következő lépésre</p>

<ol start="4">
  <li>Süssük a sütőben egy órán keresztül</li>
  <li>Vegyük ki a sütőből</li>
  <li>Hagyjuk hűlni tíz percig</li>
  <li>Szolgáljuk fel</li>
</ol>
```

Ez a következő eredményt adja:

1. Gyűjtsük be a szükséges hozzávalókat
2. Keverjük össze a hozzávalókat
3. Tegyük be a keveréket egy sütőedénybe

Mielőtt betesszük a hozzávalókat az edénybe, melegítsük fel a sütőt 180 Celsius fokra/350 Farenheit fokra, hogy készen álljunk a következő lépésre

4. Süssük a sütőben egy órán keresztül
5. Vegyük ki a sütőből
6. Hagyjuk hűlni tíz percig

7. Szolgáljuk fel

Jó tudni, hogy ez az attribútum elavultként van jelölve a HTML specifikáció utolsó verziójában, ami azt jelenti, hogy szigorú doctype esetén az oldalad nem fog átmenni a validáláson. Ez különösnek tűnhet, mivel az attribútumnak amúgy van értelme, ráadásul nincs CSS megfelelője sem. Ebből is látható, hogy a HTML validálása egy hasznos és követendő cél, de nem abszolút értelemben. Ráadásul van még egy pont, amire támaszkodhatunk: a start attribútum a HTML 5 specifikációjában már nem elavult (azt tanúsítja a HTML 5 és HTML 4 különbségeiről szóló dokumentum is). Ha fel akarod használni ezt a funkcionalitást egy szigorú HTML 4 lapon, és mindenképp azt szeretnéd, hogy validáljon, akkor használd helyette a CSS Countereket.

Definíciós lista

A definíciós listák egyes elemeket kapcsolnak össze a definíciójukkal egy lista formájában. Ha például meg akarod adni a bevásárlólistán található elemek leírását, akkor ezt megteheted egy definíciós listával:

tej

: Egy fehér, folyékony tejtermék.

kenyér

: Sütött étel lisztből vagy korpából készítve.

vaj

: Egy sárgás, szilárd tejtermék.

kávé

: A kávébab termése.

A kifejezés és a definíció együtt egy definíciós csoport (vagy egy név-érték csoport). ANynyi definíciós csoportot készíthetsz, amennyit csak akarsz, de legalább egy kifejezésnek és egy definíciónak szerepelnie kell mindegyik csoportban. Nem lehet kifejezésed definíció nélkül, és definíciód sem kifejezés nélkül.

Egy kifejezéshez kapcsolhatsz több definíciót is. Például a „kávé” kifejezésnek több értelme is lehet, és ezeket mind megadhatod:

kávé

: egy ital pörkölt, őrölt kávébabból főzve

: egy csésze kávé

: a sötétbarna szín egyik árnyalata

Hasonlóan, ugyanahhoz a definícióhoz több kifejezés is tartozhat. Ez akkor hasznos, ha több kifejezésnek ugyanaz az értelme az adott kontextusban:

szénsavas üdítő

szörp szódával

kóla

: Egy édes, szénsavas ital

A definíciós listák különböznek a többi listától, mivel kifejezéseket és azok definícióit tartalmazzák egyszerű listaelemek helyett.

Éppen ezért a definíciós listákat a **<dl></dl>** tagek határolják. Ezen belül meg kell adnod legalább egy **<dt></dt>** elemet (a kifejezésnek) és egy **<dd></dd>** elemet (a definíciónak); a **<dt></dt>** elem mindig az első kell legyen a listában.

Egy egyszerű definíciós lista a kifejezésekkel és a definíciókkal így néz ki:

```
<dl>
  <dt>Kifejezés</dt>
  <dd>A kifejezés definíciója</dd>

  <dt>Kifejezés</dt>
  <dd>A kifejezés definíciója</dd>

  <dt>Kifejezés</dt>
  <dd>A kifejezés definíciója</dd>
</dl>
```

Ez a következőképpen jelenik meg:

Kifejezés

A kifejezés definíciója

Kifejezés

A kifejezés definíciója

Kifejezés

A kifejezés definíciója

A következő példában egy kifejezéshez több definíciót is megadtunk, és fordítva:

```
<dl>
  <dt>Kifejezés</dt>
  <dd>A kifejezés definíciója</dd>

  <dt>Kifejezés</dt>
  <dt>Kifejezés</dt>
  <dd>Az előző két kifejezés közös definíciója</dd>

  <dt>Kifejezés két különböző értelemmel</dt>
  <dd>A kifejezés első definíciója</dd>
  <dd>A kifejezés második definíciója</dd>
</dl>
```

Ez a következőképpen jelenik meg:

Kifejezés

A kifejezés definíciója

Kifejezés

Kifejezés

Az előző két kifejezés közös definíciója

Kifejezés két különböző értelemmel

A kifejezés első definíciója

A kifejezés második definíciója

Általában nem szoktunk több kifejezésnek egy definíciót adni, de hasznos lehet tudni róla, hogy van ilyen lehetőség is, ha valamikor mégis szükséged lenne rá.

4.2.2 Választás a listatípusok között

Ha választanod kell a különböző típusú listák között, csak tegyél fel magadnak két egyszerű kérdést:

1. Kifejezések értelmezésére van szükségem, esetleg név/érték párosokra?
    - Ha igen, akkor használj definíciós listát.
    - Ha nem, akkor ugorj a következő pontra.
2. Fontos az elemek sorrendje a listában?
    - Ha igen, használj rendezett listát.
    - Ha nem, használj rendezetlen listát.

4.2.3 A különbség a szöveg és a HTML lista között

Talán megfordult a fejedben az is, hogy végül is mi a különbség a HTML lista és egy kézzel készített pontozott vagy számozott lista között. Nos, a HTML listának több előnye is van a saját kezűleg készített listával szemben:

- Ha meg kell változtatnod a sorrendjét egy számozott listának, akkor a HTML listában egyszerűen felcseréled az elemeket. Ha a számokat kézzel írtad az elemek elé, akkor módosítanod kell a lista elemeinek számozását, akár az egész listán végigmenve — ami végsősorban eléggé unalmas dolog.
- A HTML lista használatakor egyszerűen stílusozhatod a listát. Ha csak egyszerű szöveget használsz, valószínűleg csak valamilyen bonyolult módon oldhatod csak meg az egyes elemek stílusozását.
- A HTML lista használatával a helyes szemantikus struktúrát készítheted el, nem pedig csak egy „listaszerű” megjelenést. Ennek több előnye is van, mivel a képernyő-felolvasók így tudják jelezni a látássérült embereknek, hogy most egy listát olvasnak, és nem csak egy összefüggéstelen számokat és szövegeket olvasnak nekik.

De fogalmazhatunk más módon is: a szöveg és a lista nem ugyanaz. Ha a lista helyett szöveget használsz, akkor több dolgod lesz vele, és több problémát fog okozni az olvasóidnak is. Szóval, ha a dokumentumodban szükséged van egy listára, mindig használd a megfelelő HTML listát.

4.2.4 Listák egybeágyazása

Egy listaelem tartalmazhat újabb listákat is — ezt nevezzük a listák egybeágyazásának. Nagyon hasznos például a tartalomjegyzékek esetében:

1. Első fejezet
    1. Első rész
    2. Második rész
    3. Harmadik rész
2. Második fejezet
3. Harmadik fejezet

A kulcs a beágyazott listákhoz az, hogy ne felejtsük el, hogy a belső listának egy bizonyos listaelemhez kell tartoznia. A kódban ezt úgy tehetjük meg, hogy a belső lista teljes egészében a külső lista listaelemében található. A felső lista kódja például így néz ki:

```
<ol>
  <li>Első fejezet
    <ol>
      <li>Első rész</li>
      <li>Második rész </li>
      <li>Harmadik rész </li>
    </ol>
  </li>
  <li>Második fejezet</li>
  <li>Harmadik fejezet</li>
</ol>
```

Figyeld meg, hogy a beágyazott lista a nyitó *
* tag és a hozzá kapcsolódó szöveg („Első fejezet”) után következik, és véget is ér a záró *
* tag előtt. A beágyazott listákat gyakran használják a weboldalak navigációs menüjének elkészítésére is, mivel ez egy jó módszer a weboldal struktúrájának meghatározására.

Elméletileg akárhány listát egymásba ágyazhatsz, de a gyakorlatban a túl sok lista zavaró lehet. A nagyon hosszú listák esetében hatékonyabb, ha a listát felosztod több kisebb listára, és ezeket címsorokkal jelölöd, vagy egyenesen külön lapokra teszed őket.

4.2.5 Példa lépésről lépésre

Az eddigiek jobb megértése érdekében készíteni fogunk egy példát, amelyen lépésről lépésre megyünk végig. Vegyük a következő szituációt:

Egy kis weboldalt készítesz HTML Receptsuli névvel. A főoldalon meg kell jelenítened a receptek kategorizált listáját, amelyek a különböző receptoldalakra hivatkoznak. Mindegyik receptoldal felsorolja a szükséges hozzávalókat, megjegyzéseket fűz a hozzávalókhoz és az előkészítés módjához. A három kategória a „Sütemények” (amelyben az „Egyszerű piskóta”, „Csokis süti” és „Almás teasütemény” receptek vannak); a „Kekszek” (az „ANZAC keksz”, „Dzsemes keksz” és „Teakeksz” receptekkel); valamint a „Péksütemények” (a „Szezámmagos kifli” és „Pogácsa” receptekkel). A kliens számára mindegy, hogy a kategóriák és a receptek milyen sorrendben jelennek meg; csak azt szeretné, ha a látogatók egyértelműen látnák, hogy mely elemek a kategóriák, és melyek a receptek.

Lépjünk át a website készítésének első lépésein. Most csak a kód elkészítését mutatjuk meg, és adunk egy kevés stílust a listákhoz. A stílusozást egyelőre nem tárgyaljuk részletesen, erről a sorozat egy későbbi részében lesz szó.

A főoldal kódja

Készíts egy jól megszerkesztett HTML lapot a doctype, html, head és body elemekkel, és mentsd el listapelda-fooldal.html névvel. Adj hozzá egy főcímet (h1) „HTML Receptsuli” névvel, majd egy alcímet (h2) „Receptek” névvel:

```
<h1>HTML Receptsuli</h1>
<h2>Receptek</h2>
```

A recepteket három kategóriában kell megjelenítened, és a sorrendjük nem fontos — ehhez a legjobb egy rendezetlen lista, úgyhogy adjuk is hozzá a laphoz:

```
<h2>Receptek</h2>
<ul>
  <li>Sütemények</li>
  <li>Kekszek</li>
  <li>Péksütemények</li>
</ul>
```

Az li elemek behúzása nem szükséges, de olvashatóbbá teszi a kódot.

Most pedig hozzá kell adnod a recepteket, mint alelemek, például a „Sütemények” kategória alá az „Egyszerű piskóta”, a „Csokis süti” és az „Almás teasütemény” recepteket. Ehhez készítened kell mindegyik elemhez egy beágyazott listát. Mivel a sorrend itt sem fontos, így újra rendezetlen listát használunk. A példa egyszerűségének érdekében minden recept ugyanarra az egy receptlapra fog mutatni (a HTML hivatkozásokról részletesebben majd a 18. leírásban olvashatsz):

```
<h2>Receptek</h2>
<ul>
  <li>Sütemények
  <ul>
    <li><a href="listapelda-recept.html">Egyszerű piskóta</a>
      </li>
    <li><a href="listapelda-recept.html">Csokis süti</a></li>
    <li><a href="listapelda-recept.html">Almás teasütemény</a>
      </li>
  </ul>
  </li>
  <li>Kekszek
  <ul>
    <li><a href="listapelda-recept.html">ANZAC keksz</a></li>
    <li><a href="listapelda-recept.html">Dzsemes keksz</a></li>
    <li><a href="listapelda-recept.html">Teakeksz</a></li>
  </ul>
  </li>
  <li>Péksütemények
  <ul>
    <li><a href="listapelda-recept.html">Szezámmagos kifli</a>
      </li>
    <li><a href="listapelda-recept.html">Pogácsa</a></li>
  </ul>
  </li>
</ul>
```

Adjunk hozzá stílust

A kliensnek tetszik ez az elrendezés, de azt szeretné, ha a kategóriáknál a pöttyök helyett kis nyilak lennének. Azt is szeretné, ha a kategóriák egy vonalban lennének a lap bal szélével. Ezeket úgy érheted el, ha a pontozás helyett definiálsz egy képet, majd beállítod a lista margóját és kitöltését.

Hogy a többi lista megjelenését ne befolyásold az oldalon, rendelj hozzá egy osztályt a saját listádhoz, így lehetőséged van arra, hogy csak ennek a listának a megjelenését módosítsd a stíluslapodon. A „recept-lista” osztály megfelelőnek tűnik:

```
<h2>Receptek</h2>
<ul class="recept-lista">
```

Most pedig készítened kell egy stíluslapot a szükséges szabályokkal. A stíluslapot add hozzá a dokumentumhoz a head elemben a style elem segítségével.

Most töröljük ki a fölösleges távolságokat és kitöltéseket a listában. Alapesetben a legtöbb böngésző beállít a listának egy margó (margin) és egy kitöltés (padding) értéket — úgyhogy a legjobb, ha ezeket az elején nullára állítod. Tedd be a következő részt a style tagek közé:

```
ul.recept-lista {
  margin-left: 0;
  padding-left: 0;
}
```

A következő lépésben készíts egy képet a lista elemeinek a pontozás helyett.

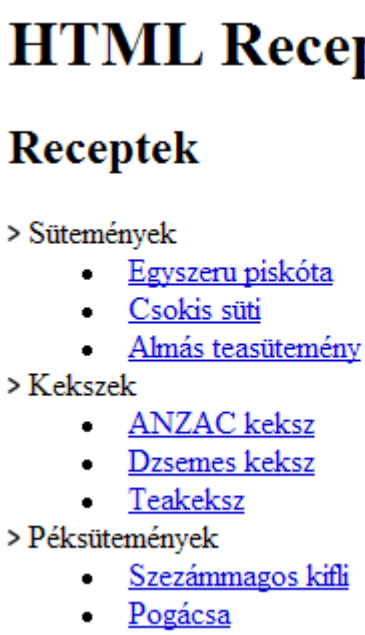
Most lecserélheted a pontokat a lista elemei mellett, és beállíthatod helyettük a saját képedet, mégpedig úgy, hogy a képet háttérképként adod meg a lista elemein, és hozzáadsz egy kevés kitöltést, hogy a szöveg ne kerüljön rá a megadott háttérképre. Ezt az alábbi CSS szabállyal adhatod meg, a style elem lezárása előtt:

```
ul.recept-lista li {
  list-style-type: none;
  background: #fff url("example-bullet.gif") 0 0.4em no-repeat;
  padding-left: 10px;
}
```

Végül vissza kell tenned a pontozást a beágyazott listákon, és módosítanod a hátterüket sima fehérre (a második beállítás specifikusabb lesz, mint az első, így felülírja a hátteres stílust). Ne felejtsd el, hogy az előbbi CSS szabályt a belső, beágyazott listák is öröklik, így mindent vissza kell állítanod. Add hozzá az alábbi CSS szabályt még a style elem lezárása elé:

```
ul.recept-lista li li {
  list-style-type: disc;
  background: #fff;
}
```

Az eredményt a 4.1. ábrán láthatod:



4.1. ábra: A befejezett főoldal, saját képpel a listaelemekhez

A recept lap

A példához az egyszerűség kedvéért csak egy receptoldalt készítünk, mégpedig a piskóta receptjét — de ha van hozzá kedved, nyugodtan készítsd el többit is ennek alapján. A kliens a piskóta receptjét egy szöveges fájlban juttatta el neked, amely így néz ki:

Egyszerű piskóta

Hozzávalók

3 tojás

100g porcukor

85g önkelő liszt

Megjegyzés a hozzávalókhoz:

Porcukor - finomra darált fehér cukor.

Önkelő liszt - előre elkészített liszt és kelesztő keverék (általában hozzáadott sóval és élesztővel).

Elkészítés

1. Melegítsük a sütőt 190°C-ra.
2. Zsírozzunk be egy 20 cm-es kerek tortaformát.
3. Egy közepes méretű tálban keverjük habosra a tojásokat és a porcukrot.
4. Keverjük bele a lisztet.
5. Öntsük bele a keveréket az előkészített tortaformába.

6. Süssük 20 percig az előmelegített sütőben, amíg a tészta teteje könnyű nyomásra vissza nem ugrik.
7. Hűtsük le a tortaformában az edényszárítón.

A recept lap kódja

Készíts egy másik helyesen formázott HTML dokumentumot, és mentsd el listapelda-recept.html névvel. Add hozzá a következő címsorokat a HTML body részéhez:

```
<h1>Egyszerű piskóta</h1>
<h2>Hozzávalók</h2>
<h2>Megjegyzés a hozzávalókhoz</h2>
<h2>Elkészítés</h2>
```

A hozzávalók listája több elemet tartalmaz, de ezeknek a sorrendje nem fontos. Ezért egy rendezetlen listába tesszük őket. Add hozzá a következő részt a megfelelő címsor alá:

```
<h2>Hozzávalók</h2>
<ul>
  <li>3 tojás</li>
  <li>100g porcukor</li>
  <li>85g önkelő liszt</li>
</ul>
```

A hozzávalókhoz tartozó megjegyzések azért vannak ott, hogy pontosan definiálják, mik is ezek a hozzávalók. Így valójában meg kell feleltetned a hozzávalót — vagyis a kifejezést — a definíciójával. Pontosan erre szolgál a definíciós lista. Add hozzá a következő kódot a HTML oldaladhoz az előbbi kód mögé:

```
<h2>Megjegyzés a hozzávalókhoz</h2>
<dl>
  <dt>Porcukor</dt>
  <dd>Finomra darált fehér cukor.</dd>
  <dt>Önkelő liszt</dt>
  <dd>Előre elkészített liszt és kelesztő keverék
      (általában hozzáadott sóval és élesztővel).</dd>
</dl>
```

Az elkészítés lépései értelemszerűen egy jól meghatározott sorrendben követik egymást, így egy rendezett listát fogunk használni. Add hozzá az alábbi kódot a HTML-hez a definíciós lista mögé:

```
<h2>Elkészítés</h2>
<ol>
  <li>Melegítsük a sütőt 190°C-ra.</li>
  <li>Zsírozzunk be egy 20 cm-es kerek tortaformát.</li>
  <li>Egy közepes méretű tálban keverjük habosra a tojásokat és
a porcukrot.</li>
  <li>Keverjük bele a lisztet.</li>
  <li>Öntsük bele a keveréket az előkészített tortaformába.</li>
  <li>Süssük 20 percig az előmelegített sütőben, amíg a tészta
teteje könnyű nyomásra vissza nem ugrik.</li>
  <li>Hűtsük le a tortaformában az edényszárítón.</li>
</ol>
```

A recept lap stílusozása

A kliensnek tetszik az eredmény, de szeretné, ha a definiált hozzávalók vastagítottak lennének a jobb olvashatóság érdekében. Ezért adjuk hozzá az alábbi kódot a HTML head részébe:

```
<style>
dt {
  font-weight: bold;
}
</style>
```

A végleges oldalt a 4.2. ábrán láthatod:

Egyszerű piskóta

Hozzávalók

- 3 tojás
- 100g porcukor
- 85g önkelő liszt

Megjegyzés a hozzávalókhoz

Porcukor
: Finomra darált fehér cukor.

Önkelő liszt
: Előre elkészített liszt és kelesztő keverék (általában hozzáadott sóval és élesztővel).

Elkészítés

1. Melegítsük a sütőt 190°C-ra.
2. Zsírozzunk be egy 20 cm-es kerek tortaformát.
3. Egy közepes méretű tálban keverjük habosra a tojásokat és a porcukrot.
4. Keverjük bele a lisztet.
5. Öntsük bele a keveréket az előkészített tortaformába.
6. Süssük 20 percig az előmelegített sütőben, amíg a tészta teteje könnyű nyomásra vissza nem ugrik.
7. Hűtsük le a tortaformában az edényszárítón.

4.2. ábra: A recept oldal a megfelelő listákkal és vastagított definíciókkal

Készen is vagyunk!

4.2.6 Tesztkérdések

Az alábbi kérdésekkel ellenőrizheted a tudásodat:

- Mi a HTML listák három típusa?
- Melyik típust mikor kell használnod? Hogyan döntöd el, hogy melyiket használod?
- Hogyan tudod a listákat egymásba ágyazni?
- Miért jobb, ha a listáid stílusozására CSS-t használsz HTML helyett?

4.3. Képek a HTML-ben

Ebben a leírásban azon dolgok egyikét mutatjuk be, amelyik a webdesignt széppé teheti — a képekről. Az anyag végére tudni fogod, hogyan adj hozzá képeket a weboldalakhoz hozzáférhető módon (vagyis hogy a látássérült emberek is képesek legyenek használni az oldaladon található információkat), hogyan használhatsz beágyazott képeket információk közlésére, valamint hogyan adhatsz háttérképeket a lapod elrendezéséhez.

4.3.1 Egy kép többet mond ezer szónál — vagy mégsem?

Nagyon csábító, hogy a weboldalaidon sok képet használj. A képekkel nagyszerűen lehet befolyásolni az oldal hangulatát, és az illusztrációkon keresztül könnyen be lehet mutatni komplex információkat is a látogatóknak.

A képek legnagyobb hátulütője a weben az, hogy nem minden webes felhasználó látja őket. Annak idején, amikor a képeket először kezdték támogatni a böngészők, nagyon sok látogató kikapcsolta a képek megjelenítését, hogy ezáltal sávszélességet spóroljon, és gyorsabban böngészhessen — az internetkapcsolatok akkoriban még nagyon lassúak voltak, és minden percért fizetni kellett, amit online töltöttél. Habár ez a mai időkben már nem jellemző, még mindig nem értünk ki az erdőből — elég, ha csak néhány példát említek:

- Azok, akik mobil eszközökön böngésznek, könnyen lehet, hogy most is kikapcsolják a képeket a kis képernyő miatt, és azért, mert gyakran a letöltött adatmennyiség után kell fizessenek.
- A látogatóid között lehetnek vakok és gyengénlátók, akik egyáltalán nem, vagy csak nagyon rosszul látják a képeidet.
- Lehetnek olyan látogatóid is, akik egy másik kultúrkörből származnak, és nem értik meg az általad használt képi jelek valódi jelentését.
- A keresőmotorok csak a szöveget indexelik — nem analizálják a képeket (egyelőre), ami azt jelenti, hogy minden olyan információ, amit a képekben tárolsz, nem lesz indexelve, és így nem fogják megtalálni a keresőben.

Ezek miatt nagyon fontos, hogy okosan válaszd meg a képeket, és csak a megfelelő helyzetekben használd őket. Még ennél is fontosabb, hogy mindig biztosíts egy tartalék lehetőséget azoknak, akik valamilyen okból kifolyólag nem látják a képeidet. Erről a témáról egy későbbi leírásban is fogunk beszélni a webes navigáció és menük témakörben, ahol szó lesz a navigációhoz hibásan használt ikonokról és képekről. Most elsősorban azt fogjuk megnézni, hogy milyen technológiákat használhatunk arra, hogy egy HTML dokumentumba képeket helyezzünk.

4.3.2 A weben található képek különböző típusai — tartalmi- és háttérképek

Alapvetően két fajta képet tehetünk a dokumentumokba: a tartalomhoz kapcsolódó képeket az img elem segítségével, valamint háttérképeket a különböző elemekhez CSS segítségével. Hogy mikor melyiket érdemes használni, az attól függ, hogy mit szeretnél:

- Ha a kép lényeges része a dokumentum tartalmának, például egy kép a szerzőről, vagy egy grafikon különböző adatokkal, akkor img elemként kell hozzáadni, egy alternatív szöveg megadásával együtt.
- Ha a kép csak dekoráció, akkor használd a CSS háttérképeit. Ennek az oka, hogy az ilyen képeknek nem kell adni alternatív szöveget (mi haszna egy vak számára annak, hogy „lekerekített, csillogó zöld sarok”?), ráadásul sokkal több lehetőséged van a képek stílusozására CSS-ben, mint HTML-ben.

4.3.3 Az img elem és az attribútumai

Egy kép hozzáadása a HTML dokumentumhoz az img elem segítségével rendkívül egyszerű. Az alábbi HTML dokumentum (inlinekeppelda.html a csomagolt fájlban) megjeleníti a balkonlatkep.jpg fotót a böngészőben (feltételezve, hogy a kép ugyanott található, mint a HTML fájl).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Példa egy inline képre</title>
</head>

<body>
  <img src="balkonlatkep.jpg">
</body>
</html>
```

Ha ezt a kódot megnyitod a böngészőben, akkor a 4.3. ábrán látható eredményt fogod kapni.

4.3. ábra: A kép, ahogy a böngészőben megjelenik

Alternatív szöveg megadása az alt attribútummal

Ez a kód szépen megjeleníti a képünket, viszont a HTML érvénytelen lesz, mivel az img elemben meg kell adni az alt attribútumot is. Ez az attribútum tartalmazza azt a szöveget, ami a kép helyén jelenik meg, ha a kép valamilyen okból nem elérhető. Ennek több oka is lehet, például nem található a kép a megadott helyen, nem lehet betölteni, vagy a kliens eszköz (ami általában a böngésző) nem támogatja a képeket. Továbbá, a látássérült emberek általában valamilyen kisegítő technológiát használnak a böngészéshez, ami felolvassa nekik a lapokat; ezek a technológiák pedig az img elem esetén éppen az alt attribútum szövegét olvassák fel a felhasználónak. Éppen ezért nagyon fontos, hogy jó alternatív szöveggel írjuk le a kép tartalmát, és ezt az alt attribútumba tegyük.

A weben sok olyan szöveget találhatsz, amelyben az „alt tagekről” beszélnek. Ez valójában hibás, mivel nincs ilyen nevű tag (vagy elem). Ez az img elem egy attribútuma, és rendkívül fontos szerepe van mind a hozzáférhetőségben, mind a keresőoptimalizálásban (SEO).

Ahhoz, hogy a képünk mindenki számára érthető legyen, hozzá kell adnunk egy alternatív szöveget, ebben az esetben például ezt: „A látkép a balkonomról, ahonnan látszik egy sor ház, néhány fa és egy kastély” (inlinekeppeldaalt.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Példa egy inline képre</title>
</head>

<body>
  <img src="balkonlatkep.jpg" alt="A látkép a balkonomról,
ahonnan látszik egy sor ház, néhány fa és egy kastély">
</body>
</html>
```

Az alt attribútum azt a szöveget tartalmazza, amely akkor jelenik meg, ha a kép nem elérhető. Az alt attribútumban található információ nem szabad megjelenjen akkor, ha a képet sikerült betölteni és megjeleníteni; ennek ellenére az Internet Explorer ezt másképp gondolja, és egy eszköztippben megjeleníti a szöveget, ha az egeret a kép fölé viszed. Ez valójában egy hiba, ami rengeteg embert arra késztetett, hogy további információkat fűzzön a képhez az alt attribútum segítségével. Ha ezt szeretnéd elérni, akkor használd helyette a title attribútumot, amelyet mindjárt be is mutatunk.

Hasznos információk megadása a *title* attribútummal

A legtöbb böngésző az img elem title attribútumának értékét egy eszköztippben jeleníti meg, ha az egeret a kép fölé viszed (lásd a 2. ábrán). Ennek segítségével a látogató többet is megtudhat az adott képről, de nem feltételezheted azt sem, hogy minden látogatódnak van egere. A title attribútum nagyon hasznos lehet, de ez a módszer nem biztonságos kritikus információk megadása esetén. Nagyon hasznos viszont például akkor, ha a kép hangulatáról akarunk írni, vagy hogy mit jelent ez a kép az adott kontextusban (inlinekeptitle.html)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Példa egy inline képre</title>
</head>

<body>
  <img src="balkonlatkep.jpg" alt="A látkép a balkonomról,
ahonnan látszik egy sor ház, néhány fa és egy kastély"
title="Ezt látom, ha kinézek az ablakomon; a kastély miatt
költöztem ide elsősorban.">
</body>
</html>
```

Ha ezt a kódot betöltöd a böngésződbe, és a kép fölé állsz az egérrel, akkor a 2. ábrán látható eredményt fogod kapni:

4.4. ábra: A title attribútum értéke megjelenik az eszköztippben

longdesc használata komplex képeknél alternatív leírás biztosítására

Ha a kép nagyon összetett, mint például egy grafikon, akkor adhatsz hozzá egy sokkal hosszabb leírást is a longdesc attribútum segítségével, így a képernyő-felolvasó szoftver

használó látogatók, vagy akik képek nélkül böngésznek továbbra is pontosan elérhetik azt az információt, ami egyébként csak a képen látható.

Ez az attribútum egy olyan URL-t tartalmaz, amely a képben található információkat tartalmazó dokumentumra mutat. Például ha a grafikon egy adathalmazt jelenít meg, akkor a longdesc segítségével hivatkozhatsz arra az adattáblára, amely alapján a grafikon készült (inlinekeplongdesc.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Példa egy inline képre longdesc leírással</title>
</head>

<body>
  <img src="chart.png" width="450" height="150" alt="A grafikon
a 15 évesek gyümölcsfogyasztását mutatja meg. A legtöbb gyerek
ananászt eszik, majd körtét" longdesc="gyumolcsfogyasztas.html">
</body>
</html>
```

A gyumolcsfogyasztas.html egy nagyon egyszerű adathalmazt tartalmaz, amely ugyanezt az információt jeleníti meg:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Gyümölcsfogyasztás</title>
</head>

<body>
<table summary="Gyümölcsfogyasztás a 15 évesek körében, 2007
március">
  <caption>Gyümölcsfogyasztás</caption>
  <thead>
    <tr><th scope="col">Gyümölcs</th><th scope="col">Mennyiség</
th></tr>
  </thead>

  <tbody>
    <tr><td>Alma</td><td>10</td></tr>
    <tr><td>Narancs</td><td>58</td></tr>
    <tr><td>Ananász</td><td>95</td></tr>
    <tr><td>Banán</td><td>30</td></tr>
    <tr><td>Mazsola</td><td>8</td></tr>
    <tr><td>Körte</td><td>63</td></tr>
  </tbody>
</table>

<p><a href="inlinekeplongdesc.html">Vissza a cikkre</a></p>
</body>
</html>
```

Az adatok kétféle megjelenítését egymás mellett a 4.5. ábrán láthatod:

4.5. ábra: Egy összetett képhez a longdesc attribútum segítségével kapcsolhatsz egy részletes leírást

Annak, hogy egy képhez hozzá van rendelve egy részletes leíró dokumentum, nincs semmilyen vizuális jelölése. A kisegítő technológiák viszont tudatni fogják a felhasználóikkal, hogy elérhető egy alternatív leírás is a képhez.

Gyorsabb képbetöltés a kép méreteinek megadásával

Amikor a kliens eszköz (vagyis a böngésző) talál egy img elemet a HTML-ben, akkor elkezdi letölteni azt a képet, amelyre az src attribútum mutat. Normális esetben nem ismeri a kép méreteit, ezért egyszerűen csak bedobja a szövegeket, majd amikor a kép letöltése befejeződött, akkor félretolja őket és megjeleníti a képet a maga helyén. Ez lelassítja a lap betöltését, ráadásul zavaró lehet a látogató számára is. Ha ezt el szeretnéd kerülni, akkor előre megadhatod a kliens eszköz számára a kép méreteit a width és height attribútumokkal, így már a kép betöltése előtt fenn tudja tartani számára a megfelelő méretű helyet (inlinekepmeretekkel.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Példa egy inline képre</title>
</head>
```

```
<body>
  <img src="balkonlatkep.jpg" alt="A látkép a balkonomról,
ahonnan látszik egy sor ház, néhány fa és egy kastély"
width="400" height="186">
</body>
</html>
```

Ez előre kitölti a kép helyét egészen addig, amíg be nem töltődik, és meg nem jelenik a valódi kép, így elkerülheted az oldal idétlen átrendezését. A kép valódi méretét is megváltoztathatod ezeknek az attribútumoknak a segítségével (próbáld ki a fenti példában, hogy megfelezed az értékeket, majd töltsd be újra a lapot). Ez általában nem működik megfelelően, mivel a kép átméretezésének minősége nem egyforma minden böngészőben. Különösen káros az, ha a nagy képekből ezzel a módszerrel készítesz bélyegképeket, mivel a bélyegképeknek éppen az lenne a lényege, hogy kisebb fizikai fájlmérettel rendelkeznek a kis képméret mellett; senki sem akar letölteni egy 300 KB méretű apró képet, amikor az akár 5 KB is lehetne.

4.3.4 Ennyit az inline képekről

Még van jó néhány attribútum, amit a képekhez használhatsz, de a legtöbb ezek közül már elavult, mivel a kép elrendezését és elhelyezkedését adják meg. Ez nem a HTML feladata — erre találták ki a CSS-t. Egyelőre elég annyit mondani, hogy van még egy fontos dolog, amit érdemes megjegyeznek, mégpedig hogy a képek — alapesetben — inline, azaz soron belüli elemek. Ez azt jelenti, hogy beteheted őket a szövegben a szavak közé anélkül, hogy új sort kezdenének. Ez nagyon jó abban az esetben, ha kis ikonokat vagy hangulatjeleket akarsz betenni a szövegedbe, de idegesítő lehet akkor, ha az elrendezést képekkel és szövegekkel akarod megoldani. A CSS segítségével felülírhatod a képeknek ezt a viselkedését, és beállíthatod, hogy a képek blokk szintű elemként viselkedjenek (vagyis minden esetben új sort kezdenek, ha hozzáadod őket a dokumentumhoz).

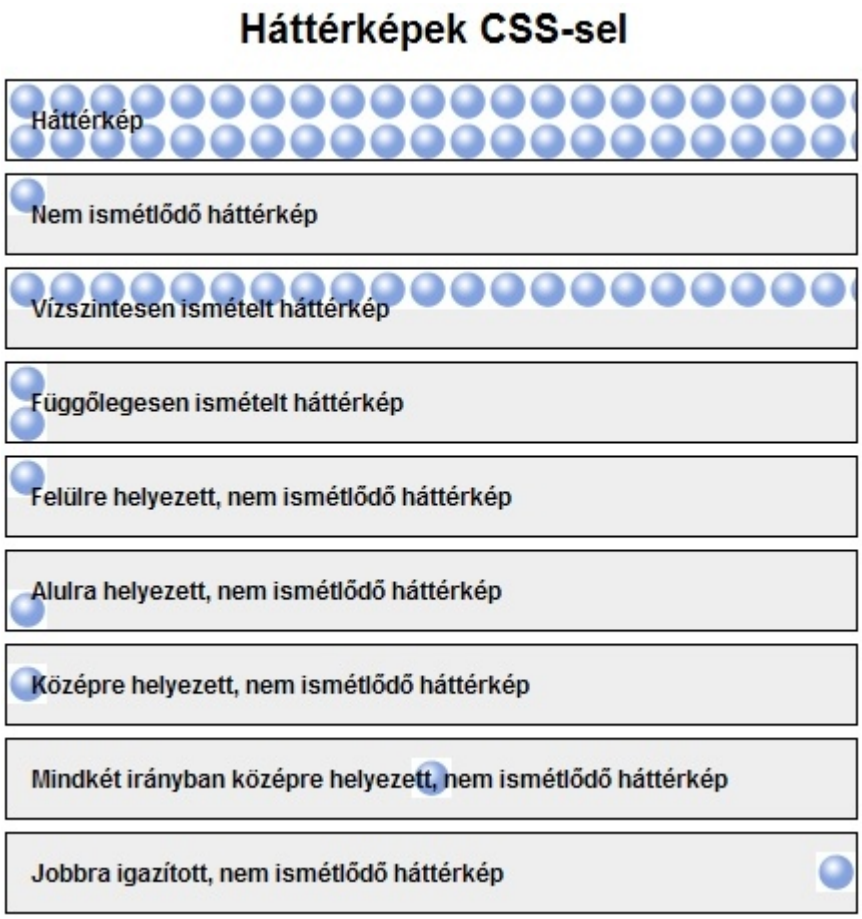
4.3.5 Háttérképek CSS-sel

Alighanem nagy biztonsággal kijelenthetjük, hogy a webfejlesztés sokkal vidámabb lett azt követően, hogy a böngészők elkezdték támogatni a CSS-t. Addig a lap különböző elemeinek elrendezéséhez a HTML táblázatait kellett összetákolni, az üres helyeken kitöltő elemeket ( ), a margókhoz pedig kitöltő GIF-eket (1x1 pixeles átlátszó GIF képeket) kellett használni. Ezek helyett végre lehetőség nyílt a valódi kitöltés (padding), margó (margin), méret és pozícionálás használatára a CSS-sel, míg a HTML-t meghagyhattuk a tartalom felépítéséhez.

A CSS egyben azt is jelenti, hogy a háttérképeket rendkívül sokoldalú módon használhatod fel — úgy pozícionálhatod őket a szöveg alá vagy köré, ahogy éppen szeretnéd, de egymás mellé is helyezheted ismételten a képeket a megfelelő háttér előállításához. Most csak érintőlegesen fogunk beszélni a CSS képekkel kapcsolatos lehetőségeiről, mivel egy későbbi leírásban sokkal részletesebben is lesz még szó a CSS háttérképeiről.

Hogyan adható meg a háttér CSS-ben?

CSS-ben egy háttérkép megadása nagyon egyszerű. Mielőtt megnéznéd az alábbi CSS kódot, nézd meg a 4.6. ábrát, hogy legyen valamilyen fogalmad a CSS háttérképek különböző lehetőségeiről.



4.6. ábra: CSS háttérképek

A különböző dobozok valójában stílusozott h2 címsor elemek, egy kevés kitöltéssel és kerettel, amiket CSS-ben rendeltünk hozzájuk, hogy elég helye legyen a háttérképnek. Ha megnézed a HTML fájlt, látni fogod, hogy mindegyik h2 elemnek egy egyedi id azonosítója van, így mindegyikhez más CSS szabályt tudunk rendelni. Az első példához tartozó CSS a következő:

```
background-image:url(ball.gif);
background-color:#eee;
```

A képet a *background-image* szelektorral adhatod hozzá a háttérhez, és a zárójelben adhatod meg, hogy milyen képet töltsön be. Ha a kép valamilyen okból nem elérhető, akkor második lehetőségként érdemes megadnod egy háttérszínt is a *background-color* szelektorral, a szín kódjával egyetemben (ami lehet egy hexadecimális, nevesített vagy RGB érték). Ebben az esetben világosszürkét választottunk.

Alapesetben a háttérképek mind vízszintesen, mind függőlegesen ismétlődnek, amíg ki nem töltik a teljes elemet. Az ismétlődés te is módosíthatod a background-repeat szelektor használatával:

- Ha egyáltalán nem akarsz ismétlődést: *background-repeat:no-repeat;*
- Ha csak vízszintes ismétlődést akarsz: *background-repeat:repeat-x;*
- Ha csak függőleges ismétlődést akarsz: *background-repeat:repeat-y;*

Alapesetben a háttérkép (ha nem ismétlődik) az elem bal felső sarkába kerül. Ezt is módosíthatod a background-position szelektorral, így oda teheted a háttérképet, ahová akarod. A legegyszerűbb választható értékek a top (fel), center (középre) és bottom (le) a

függőleges, valamint left (balra), center (középre) és right (jobbra) a vízszintes elhelyezésre. Ha például a képet a jobb alsó sarokba akarod helyezni, akkor a background-position:bottom right; formát kell használnod; ha pedig függőlegesen középre, míg vízszintesen jobbra akarod helyezni, akkor a background-position:center right; forma lesz a megfelelő.

Az ismétlődés és a pozícionálás befolyásolásával, valamint egy jó háttérkép használatával olyan lenyűgöző megjelenéseket tudsz készíteni, amelyekre a CSS előtt nem volt lehetőség. Ráadásul a háttérképeket egy külön CSS fájlban definiálhatod, így később nagyon egyszerűen módosíthatod a teljes website megjelenését néhány sor kód átírásával. Erről később is lesz még szó a 30. leírásban.

4.3.6 Tesztkérdések

Miért fontos, hogy jó szöveget adjunk a képhez az alt attribútumban? Valóban szükség van erre?

Van egy 1280x786 méretű képed, amit egy 40x30 méretű bélyegképként szeretnél megjeleníteni. Megteheted ezt HTML-ben? Miért nem okos dolog ezt tenni?

Mit csinál a longdesc attribútum, és hogyan jelenítik meg ezt a böngészők?

Mire jók a valign és az align attribútumok? Miért nem beszéltünk ezekről?

Hová kerülnek alapesetben a CSS háttérképek az elemen belül, és hogyan ismétlődnek?

4.4. HTML hivatkozások — építsük fel a webet!

Ebben a leírásban mindent meg fogsz tudni a web történetének egyik legnagyobb dobásáról — a hivatkozásokról. Egy dokumentum olvasója a hivatkozást követve átugorhat egy másik dokumentumra, és egyik kiszolgálóról átléphet egy másik kiszolgálóra úgy, hogy közben nem kell újra kapcsolódnia minden alkalommal. Mióta feltalálták, már nagyon sok minden történt, de egy dolog változatlan maradt: a hivatkozások most is különösen fontos részei a webes életnek, és a használatukkal jelentősen megkönnyítheted, de akár meg is nehezítheted a látogatóid dolgát.

Miután végigolvasod ezt a leírást, tudni fogod, hogyan készíthetsz olyan hivatkozásokat, amelyek jól érthetőek, és minden környezetben működnek. Azt is meg fogod ismerni, hogyan befolyásolják a hivatkozások az oldalad népszerűségét a keresőkben, és kapsz néhány tippet a hivatkozások elnevezéséhez is.

4.4.1 Mik azok a hivatkozások?

A hivatkozások a weblap részei (melyeket a legtöbb esetben HTML-ben hoznak létre, de nem mindig), és valamilyen másik erőforrásra mutatnak — más HTML dokumentumokra, szöveges fájlokra, PDF fájlokra, stb. Vannak olyan hivatkozások, amelyeket a böngésző automatikusan követni szokott, ilyenek például a link elem hivatkozásai (ezekkel már találkozhattál néhány korábbi leírásban, ezt használtuk például a CSS fájlok importálására a HTML dokumentumban), míg más hivatkozásokat a felhasználó aktiválhat tetszés szerint. Ezeket kapcsoknak nevezzük, és az a elem segítségével adhatod hozzá a dokumentumhoz.

4.4.2 Egy hivatkozás felépítése

Bármilyen soron belüli (inline) elemet átalakíthatsz kapcsolt hivatkozássá, ha köré teszed az a elemet. Például az alábbi HTML dokumentumban a Yahoo Developer Network szöveg hivatkozássá alakul (linkpelda.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-
8">
<title>Hivatkozás példa</title>
<link rel="stylesheet" href="linkexamplestyles.css">
</head>

<body>
  <h1>Hivatkozás a YDN-re</h1>
  <p><a href="http://developer.yahoo.com">Yahoo Developer
Network</a></p>
</body>
</html>
```

Azok a látogatók, akik aktiválják ezt a hivatkozást (rákattintva az egérrel, kiválasztva a billentyűzettel vagy hanggal) el fogják hagyni az aktuális oldalt, és átkerülnek a YDN oldalára. A hivatkozással sok mindent lehet csinálni, de a stílusozásáról csak később fogunk részletesebben is beszélni.

A hivatkozáshoz több attribútumot is megadhatsz:

- *href* — az erőforrás, amire mutat (ez egy külső fájl vagy egy azonosító).
- *id* — az azonosító, ha ez a kapocs egy cél, és nem egy hivatkozás.
- *title* — további információk a külső erőforrásról.

Először is vegyük sorra a legfontosabb attribútumokat, majd nézzük meg, hogyan könnyítheted meg a látogatóid dolgát.

4.4.3 Hivatkozás vagy cél? Az *id* és *href* attribútumok

Az a elemnek többféle szerepe is lehet a dokumentumban attól függően, hogy milyen attribútumot állítottunk be rajta. A leggyakoribb, amit közülük használni fogsz, az a href attribútum, ez adja meg, hogy egy hivatkozás milyen erőforrásra mutat. Ez az attribútum több különböző értéket is tartalmazhat:

- Egy URL-t az aktuális mappában (sugo.html), vagy egy másik mappában, az aktuális mappához viszonyított útvonallal megadva (például "../../sugo/sugo.html" — a 2 pont itt a hierarchiában eggyel feljebb található mappát jelenti), vagy pedig a kiszolgálón található abszolút útvonallal (például "/sugo/sugo.html" — itt a kezdő perjel azt jelenti, hogy az útvonal a kiszolgáló gyökérkönyvtárából indul).
- Egy URL-t egy teljesen más kiszolgálóhoz (például "http://wait-till-i.com" vagy "ftp://ftp.opera.com/" vagy "http://developer.yahoo.com/yui").
- Egy szakaszazonosítót vagy egy hivatkozásnevet, kettőskereszt után megadva (például "#menu"). Ez a dokumentumon belül mutat egy területre.

- Egy vegyes, URL-ből és szövegrész azonosítóból álló szöveget — így direkt tudsz hivatkozni egy másik dokumentumban egy megjelölt szakaszra, az URL után álló szakaszazonosítóval (például "http://developer.yahoo.com/yui/#cheatsheets").

Bármelyik megoldás ezek közül egy hivatkozást fog készíteni, amelyik valahová máshová mutat. Másrészről az id attribútum csak egy kapcsot hoz létre a lapon — erre tudnak aztán más hivatkozások mutatni. Ez talán első hallásra bonyolultnak tűnik, mivel mindkét esetben ugyanazt az elemet használjuk (a). Hogy könnyen megjegyezhesd, gondolj rájuk így: az id attribútum csak egy hivatkozás jelölőt készít, amelyet aztán felhasználhatsz arra, hogy a dokumentum különböző szakaszaira hivatkozz vele. Az alábbi HTML-ben az összes felsorolt hivatkozástípusra találsz egy példát (linkpeldak.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-
8">
<title>Különböző hivatkozás példák</title>
<link rel="stylesheet" href="linkexamplestyles.css">
</head>

<body>
  <h1>Különböző hivatkozás példák</h1>

  <h2>Példa a lapon belüli navigációra szakasz azonosítókkal,
hivatkozásokkal és kapcsokkal</h2>
  <div id="nav">
    <ul id="toc">
      <li><a href="#sec1">Első szakasz</a></li>
      <li><a href="#sec2">Második szakasz</a></li>
      <li><a href="#sec3">Harmadik szakasz</a></li>
      <li><a href="#sec4">Negyedik szakasz</a></li>
      <li><a href="#sec5">Ötödik szakasz</a></li>
    </ul>
  </div>

  <div id="content">
    <div>
      <h2><a id="sec1">Szakasz #1</a></h2>
      <p><a href="#toc">Vissza a menühöz</a></p>
    </div>

    <div>
      <h2><a id="sec2">Szakasz #2</a></h2>
      <p><a href="#toc">Vissza a menühöz</a></p>
    </div>

    <div>
      <h2><a id="sec3">Szakasz #3</a></h2>
      <p><a href="#toc">Vissza a menühöz</a></p>
    </div>

    <div>
      <h2><a id="sec4">Szakasz #4</a></h2>
      <p><a href="#toc">Vissza a menühöz</a></p>
    </div>
```

```
    <div>
      <h2><a id="sec5">Szakasz #5</a></h2>
      <p><a href="#toc">Vissza a menühöz</a></p>
    </div>
  </div>

  <h2>Néhány más hivatkozás példa</h2>
  <ul>
    <li><a href="http://developer.yahoo.com">Yahoo Developer
Network</a></li>
    <li><a href="http://dev.opera.com/articles/view/the-
freelancing-business-part-1-pricing/#marketing">Önmarketing
tippek</a></li>
    <li><a href="ftp://get.opera.com/pub/opera/win/">Az Opera
különböző verzióinak letöltése</a></li>
    <li><a
href="http://farm1.static.flickr.com/56/188791635_0b8bdd808d.jpg
?v=0">Képek a könyvemről</a></li>
  </ul>
</body>
</html>
```

Nyisd meg ezt a fájlt a böngésződben, és próbáld ki benne a hivatkozásokat. Ha az első listából aktiválod a linkeket, akkor azt fogod észrevenni, hogy ezek a dokumentum megfelelő szakaszaira fognak ugrani. Ez azért van, mert a kapcsolódó szakasz azonosítóra hivatkoznak — a lista első eleme például a href attribútummal hivatkozik a #sec1 azonosítójú szakaszra, ami az első h2 elemben található hivatkozás id attribútumának felel meg. Ennyit kell tenned ahhoz, hogy összeköss két kapcsot egy dokumentumon belül — csak add meg a href attribútumban azt az értéket, amelyik a hivatkozott rész id attribútumában található, és tegyél elé egy kettőskeresztet. Figyeld meg azt is, hogy a böngésződ címsorában megjelenített URL ilyenkor tartalmazza a szakaszazonosítót is, így a látogatók elmenthetik vagy elküldhetik emailben a direkt erre a szakaszra mutató hivatkozást, amellyel egyből a dokumentumnak a megadott pontjára ugorhatnak.

Ha viszont a „Vissza a menühöz” linkeket aktiválod, akkor is működik a funkcionalitás, bár a célnál nincs megadva egy azonosítóval rendelkező hivatkozás. Hogyan lehetséges ez? Úgy, hogy a szakaszazonosító valójában bármilyen elem lehet, amelyiknek meg van adva az id attribútuma. Ismétlésként:

- A hivatkozásoknak meg lehet adni egy szakasz azonosítót a href attribútumban — az azonosító ilyenkor mindig kettőskereszttel (#) kezdődik.
- Aktiváláskor a hivatkozás átugrik bármilyen HTML elemre, amelyiknek az id attribútuma a megadott értéket tartalmazza. Egy lapon belül az id attribútumoknak egyedieknek kell lenniük.
- Az id elemek elnevezésére vonatkozik néhány megkötés. A legfontosabb, hogy egy alfanumerikus karakterrel kell kezdődnie, és nem lehet benne üres hely.

Ezzel lefedtük a menüt és a példában szereplő különböző szakaszokat, de mi van a többi hivatkozással? Ha kipróbálod őket, látni fogod, hogy mindegyik más célra mutat — az egyik egy másik oldalra visz, a másik megjelenít egy képet, a harmadik egy másik weboldal egy bizonyos pontjára visz át (a megfelelő id megadásával). Ha mindegyik működött nálad, az nagyszerű, de mi van akkor, ha a böngésződ nem képes egyik-másik linkkel megbirkózni?

4.4.4 Ne hagyj kétséget afelől, hogy mire hivatkozol

Soha ne felejtsd el az egyik legfontosabb dolgot a hivatkozásokkal kapcsolatban, mégpedig hogy alapvető részét képezik a felhasználóval való kapcsolatodnak. Ha felkínálsz nekik egy hivatkozást, akkor megbíznak abban, hogy nyugodtan követhetik, és ott hasznos, releváns információt fognak találni. Ha a hivatkozásaid nem működnek, mert a hivatkozott tartalom nem elérhető, vagy olyan formában található meg, amit a látogató nem tud megnyitni, akkor visszaélsz ezzel a bizalommal és elveszíted a hiteledet a látogatók szemében. Ne hagyd, hogy ez megtörténjen.

Extra információk a *title* attribútummal

Mint sok más HTML elemhez, az a elemhez is hozzá lehet adni egy title attribútumot, ha további információkat akarsz megadni. A böngészők ezt a szöveget egy ún. tooltipben jelenítik meg, ha a látogató a hivatkozás fölé viszi az egérkurzort. Ez a megjegyzés megmondhatja számukra, hogy mire vonatkozik ez a hivatkozás. Például adhatsz egy rövid bevezetést a hivatkozott tartalomhoz:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-
8">
<title>Extra információk hozzáadása a title
attribútummal</title>
<link rel="stylesheet" href="linkexamplestyles.css">
</head>

<body>
  <h1>Extra információk hozzáadása a title attribútummal</h1>
  <ul>
    <li>További információt a <a title="A Yahoo Developer
Network egy központi oldal azokhoz az eszközökhöz, amelyeket a
Yahoo biztosít, mint például a Yahoo User Interface Library
(YUI) és a Design Patterns repository"
href="http://developer.yahoo.com">Yahoo Developer Network</a>
oldalán találhatsz.</li>
  </ul>
  </body>
</html>
```

Ennek ellenére nem várhatod el azt sem a látogatóidtól, hogy elég figyelmük és türelmük legyen ehhez, így nem támaszkodhatsz teljes mértékben csak erre a megoldásra. Könnyen lehet, hogy a látássérült felhasználók, akik a lapot egyáltalán nem is látják, nem fognak hozzáférni ehhez az információhoz. Bár a képernyőfelolvasók képesek kiolvasni a *title* attribútum tartalmát, alapesetben ezt nem olvassák fel a felhasználóknak — éppen ezért soha ne adj meg fontos információt egy hivatkozásról a title attribútumban. Fontos információk a következők lehetnek:

- Hivatkozás nem HTML erőforrásokra, mint például PDF fájlok, képek, videók, hangfájlok vagy letöltések.
- Hivatkozás egy másik kiszolgálóra, ami elhagyja az aktuális oldalt (külső kontra belső linkek).

- Hivatkozás egy olyan dokumentumra, amelyik egy új lapon vagy egy felugróban fog megnyílni.

Hivatkozás nem HTML erőforrásokra — ne hagyd, hogy találgassanak

Nagyon bosszantó tud lenni, ha rákattintasz egy hivatkozásra, és a böngésződ nem tudja, hogy a hivatkozáson található tartalommal mit kezdjen. Mégis nagyon gyakran előfordul az, hogy a weboldalak mindenféle figyelmeztetés nélkül hivatkoznak képekre, PDF dokumentumokra és videókra anélkül, hogy erről a felhasználót értesítenék. A videók különösen gyakran le tudják fagyasztani a böngészőket. Az ilyen erőforrások ráadásul sokszor elég nagyméretűek (biztos láttál már 20 megás PDF-et), ami azt jelenti, hogy a felhasználó esetleg le szeretné tölteni ahelyett, hogy megnyitná a böngészőben, és tovább növelné annak memóriafogyasztását, de az is lehet, hogy egyáltalán nem akar megnyitni nagyméretű fájlokat.

A webes termékek sikerének egyik legnagyobb titka abban rejlik, hogy ne hagyjuk az embereket bizonytalanságban az általuk végezhető műveletekkel kapcsolatban, hanem tisztán és nyíltan leírjuk, hogy a különböző műveleteknek milyen hatásai lesznek. Hogy elkerülhesd a bosszúságot, a hivatkozások esetében mindössze annyit kell tenned, hogy megmondod a látogatónak, milyen erőforrásra mutat ez a hivatkozás. Íme néhány példa:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-
8">
<title>Hivatkozás nem HTML erőforrásokra</title>
<link rel="stylesheet" href="linkexamplestyles.css">
</head>

<body>
  <h1>Hivatkozás nem HTML erőforrásokra</h1>
  <ul>
    <li>További információt a <a
href="http://developer.yahoo.com">Yahoo Developer Network site
(külső)</a> oldalán találhat.</li>
    <li>Töltsd le a <a href="http://www.wait-till-
i.com/stuff/JavaScript-DOM-Cheatsheet.pdf">Dom tippek és trükkök
(PDF, 85KB)</a> dokumentumot</li>
    <li>Válaszd ki és <a
href="ftp://get.opera.com/pub/opera/win/">töltsd le az Opera
különböző verzióit az FTP oldalukról (külső)</a></li>
    <li>Nézd meg a <a
href="http://farm1.static.flickr.com/56/188791635_0b8bdd808d.jpg
?v=0">képet a könyvemről (JPG, 200KB)</a></li>
  </ul>
  </body>
</html>
```

Külső kontra belső hivatkozások

A webes vállalkozások egyik legnagyobb félelme az, hogy a látogatóik egyszer csak fogják magukat, elhagyják a cég weboldalát, és lelépnek a hipertérbe. Emiatt gyakran tiltják meg külső hivatkozások használatát (kivéve, ha a külső hivatkozás célpontja hajlandó eleget fizetni az átirányított forgalom után). Erre a döntési hibára még később visszaté-

rünk; most csak arról fogunk beszélni, hogy mit tesznek az emberek azért, hogy a látogatók ne hagyják el az oldalt, és ez hogyan befolyásolja az oldal sikerét.

4.4.5 Keretek és felugrók — mondj nemet rájuk

A látogatók elvesztésének a félelme és a más oldalakra való hivatkozás kényszerűsége olyan torz fejlesztéseket hozott a webfejlesztésben, amelyek évekig mérgezték az oldalak használhatóságát. Ezek a keretek (frame-ek) és a felugrók.

A HTML keretek használata azt jelenti, hogy a böngészőben megjelenített lapot több különálló dokumentumra osztod fel. Ennek az előnye az, hogy a dokumentum látszólag változatlan marad akkor is, ha a különböző részeit több kiszolgálóról töltöd is be, akár a sajátodról, akár más, külső szerverekről. A használhatóság ezen a ponton véget is ér — a keretek borzalmas felhasználói élményt nyújtanak, és rendkívül károsak:

- A keresőmotorok nem tudják a teljes oldal leindexelni, viszont a találatoknál megjeleníthetik a lap egyik részét, amely így, kontextus nélkül akár teljesen értelmetlen is lehet.
- A látogatók nem tudnak könyvjelzőt készíteni a laphoz — amikor legközelebb megnyitják az elmentett könyvjelzőt, akkor a keretrendszer eredeti állapotát fogják látni, és nem azt az állapotot, amiben a mentést végezték.
- Azoknak a látogatóknak, akik kisegítő technológiákra szorulnak, igencsak nehéz napjuk lesz, ha navigálni akarnak a keretek között.
- A külső oldalak sokszor nem szeretik, ha egy ilyen keretben próbálják megjeleníteni őket, és ún. „framebreaker” szkripteket használnak arra, hogy lecseréljék az URL-t a valódi címre az ilyen esetekben. Erre azért van szükség, hogy meggátolják egyes adathalász oldalak trükkjét, amelyek ehhez hasonló módon próbálják rávenni az embereket arra, hogy megadják a webbankos jelszavukat vagy a hitelkártyájuk adatait.

A keretkészleten belüli hivatkozások a target attribútumot használják fel arra, hogy megnevezzék a helyes keretet. Minden keret a keretkészleten belül kap egy bizonyos nevet, és a hivatkozást aktiválásakor a href attribútumban megadott dokumentum a megadott keretben fog megjelenni. Ha keretkészlet nem elérhető (például a látogató a dokumentumot egy keresőben találta meg), akkor mindegyik hivatkozás egy új lapot fog nyitni.

Az új lap nyitása egy másik gyakori megoldás a külső oldalakra való hivatkozásra — vagy egy szkriptes felugró ablakkal, vagy a target attribútum _blank értékével. Az a tény, hogy manapság már minden modern böngésző le tudja tiltani a felugró ablakokat, elég jól mutatja, hogy mennyire építhetsz erre a technikára. Nem igazán.

Röviden: ne használd a target attribútumot, amikor hivatkozásokat hozol létre, csak abban az esetben, ha pontosan tudod, mit csinálsz. A módszer különben is már rég elavult — ma már a böngészők tudnak füleket nyitni, így a látogatók a külső hivatkozásokat megnyithatják egy másik fülön a háttérben, hogy később olvassák el azokat, és közben ott maradhatnak az oldaladon. Bizonyos esetekben különbséget tehetsz valamilyen jelöléssel a külső és a belső hivatkozások között, de mindig hagyd a döntést a felhasználóra, hogy mit szeretne tenni velük.

4.4.6 A bejövő és kimenő hivatkozások haszna

Több előnye is van annak, ha külső oldalakra hivatkozol, akár még a konkurenciára is.

- Megbízhatóvá tesz a látogatók előtt — azt sugallja, hogy biztos vagy a saját tartalmaid minőségében, és nem bújsz el a kihívóid elől.
- Lehetőséget ad arra, hogy teljes szolgáltatást nyújts — hivatkozhatsz olyan tartalmakra, cikkekre vagy akár termékekre más oldalakról, amelyeket te nem tudsz lefedni, de a látogatóidat esetleg érdekelhetik, ha mélyebben akarnak a témával foglalkozni.
- Ha egy jobb vagy egy eltérő megoldást dolgoztál ki valamire, akkor hivatkozhatsz a régi megoldásra referenciaként.

A bejövő hivatkozások hasznosságát (amikor külső oldalak hivatkoznak az oldaladra) kevesen vitatják. Minél több érvényes és minőségi oldal hivatkozik rád releváns kulcsszavakat használva, annál nagyobb besorolást kapsz a webes keresőkben, mint például a Google. Viszont mielőtt ez megtörténne, bizonyítanod kell, hogy te sem félsz másokra hivatkozni az oldaladról.

A releváns kulcsszavak egy másik nagyon fontos témához irányítanak a jó hivatkozások készítésével kapcsolatban, mégpedig hogy hogyan nevezzük el őket.

4.4.7 Hivatkozások elnevezése

Erről már részben beszéltünk annál a résznél, ahol a nem HTML erőforrásokra mutató hivatkozásokról volt szó, de azért nem árt emlékeztetni magunkat, hogy egy hivatkozás nem csak egyszerűen a lap szövegezésének a része, hanem egy interaktív elem a dokumentumban.

Egyes kisegítő megoldások a teljes dokumentum helyett először csak a hivatkozások listáját ajánlják fel a látogatóak, hogy gyorsan átfuthasson rajtuk, ami azt jelenti, hogy érdemes figyelned arra, hogy a hivatkozásaid szövege önmagában is értelmes legyen, a saját kontextusán kívül is. Ezt könnyen ellenőrizheted Operában, ha megnyitod a weblapot, és kiválasztod az Eszközök > Hivatkozások pontot a menüben, vagy megnyomod a Ctrl + Shift + L gombokat. Egy új fülön látni fogod a dokumentum összes hivatkozását, valamint azt is, hogy melyik hová mutat.

A fentiek alapján nem csak arra érdemes figyelned, hogy a hivatkozások szövege értelmes legyen, hanem arra is, hogy ne legyenek olyan linkek, amelyeknek ugyanaz a szövege, de különböző címekre mutatnak. A klasszikus hiba a „klikk ide” típusú linkek, például a következő szövegkörnyezetben: „A program utolsó változatának letöltéséhez kattints ide”. Sokkal jobb, ha olyan szöveget választasz a hivatkozásnak, amelyik megmagyarázza, hogy az mire mutat — ebben az esetben például „Töltsd le és próbáld ki az alkalmazás utolsó változatát”.

Ugyanez vonatkozik a „tovább” linkekre is. Ezeket főleg a híroldalakon találod meg, ahol van egy címsor és egy bevezető szöveg, majd egy „tovább” vagy „bővebben” link a végén. Erre a problémára megoldás lehet, ha készítesz egy „tovább” képet, és egyedi alternatív szöveget adsz hozzá, vagy készítesz egy span elemet a hivatkozáson belül, amit aztán CSS-sel elrejtesz. Ezekről a trükkökről bővebben is fogunk még beszélni a sorozat egy későbbi részében, a menük és a navigáció keretében.

4.4.8 Hivatkozások stílusozása

Bár még nem érkeztünk el a sorozatban a CSS-hez, de ezen a ponton érdemes megemlíteni, hogy nagyon fontos a hivatkozásaid megjelenése is, és ehhez a hivatkozásoknak több különböző állapotát kell figyelembe venned. A linkek lehetséges állapotai (amelyek

megfelelnek a CSS pszeudo-szelektoroknak — ne ijedj meg, nem bonyolult) a következők:

- *link* — ez az alapértelmezett állapot — ezzel határozhatod meg, hogyan nézzen ki egy hivatkozás a dokumentum különböző részeiben. Alapesetben a még nem látogatott hivatkozások kékkel jelennek meg.
- *visited* — azoknak a hivatkozásoknak a stílusa, amelyeket már korábban meglátogattál (és valószínűleg még benne is vannak a böngésző gyorsítótárában). Alapesetben a látogatott hivatkozások lilával jelennek meg.
- *hover* — a hivatkozás stílusa, amikor az egérkurzor fölötte van.
- *active* — a hivatkozás stílusa, amikor aktiválva van, például amíg kattintás után felépül a kapcsolat a hivatkozott oldalra; olyankor is ebben az állapotban van az utolsó aktivált link, amikor a böngészőben visszalépsz egy korábbi oldalra.

4.4.9 Tesztkérdések

- Mi a hiba a következő hivatkozásban: *töltse le a legutolsó jelentést*?
- Mire szolgál a hivatkozásoknál a *target* attribútum, és hogyan lehet ezt jól felhasználni?
- Hogyan készíthetsz olyan hivatkozást, amelyik aktiváláskor a lap alsóbb részére visz el? Mire kell vigyáznod az ilyen esetekben?

4.5. HTML táblázatok

„Ó jaj!” — hogyan tudom a webes szabványokkal ezt a rengeteg adatot megjeleníteni? A lelki szemeid előtt talán már meg is jelentek a többszörösen egymásba ágyazott elemek, amelyek szépen sorokba és cellákba szabályozzák az adathalmazt… de nem kell aggódnod, van megoldás — a táblázatok segítenek!

A webdesignban a táblázatokat nagyon jól lehet használni arra, hogy az adatokat rendezett módon jelenítsd meg. Más szóval, a táblázatokra, diagramokra és a többi hasonló grafikonra gondolhatsz úgy is, hogy ezek segítenek az információk összehasonlításában, kontrasztba állítják a különböző adatrészeket. Gyakran láthatsz ilyeneket weblapokon, például az elnökválasztás szavazatainak összehasonlításakor, sport statisztikákról, árak összehasonlításakor, méretek megjelenítésekor és még sok másféle adat esetén.

Az internet őskorában, még mielőtt a CSS elterjedt volna, mint egy HTML-től különálló prezentációs réteg, a táblázatokat gyakran használták a weblapok elrendezésének beállítására — oszlopokat, dobozokat hoztak így létre, és ezzel helyezték el a weblap különböző részeit az oldalon. Ez a lehető legrosszabb módszer az elrendezés megvalósítására; a táblázatos elrendezéssel kusza, zavaros lapokat kapunk egy halom egymásba ágyazott táblázattal — a végén csak hatalmas fájlok maradnak, amelyeket nehéz karbantartani és még nehezebb módosítani. Manapság még mindig nagyon sok oldalt találhatsz a neten, amelyek így épülnek fel, de a biztonság kedvéért te már a táblázatokat csak arra használd, amire készültek — vagyis adatok megjelenítésére —, és az elrendezés kialakításához használd inkább a CSS-t.

4.5.1 A lehető legegyszerűbb táblázat

Először is építsük fel a táblázathoz szükséges szemantikus HTML kódot — a példában most Észak-Amerika vulkánkitöréseit fogjuk megnézni. Nagyon szeretem a vulkánokat, és gyerekkoromban sikerült is meggyőznöm az édesanyámat, hogy vigyen el ezekhez a vulkánokhoz, amikor meglátogattuk a nagymamát. Nagyon reménykedtem benne, hogy a vakáció alatt valamelyik ki fog törni, de sajnos hiába. Lássuk akkor az első táblázatunkat:

```
<table>
    <tr>
        <td>Vulkán neve</td>
        <td>Hely</td>
        <td>Utolsó nagyobb kitörés</td>
        <td>Kitörés típusa</td>
    </tr>
    <tr>
        <td>Mt. Lassen</td>
        <td>Kalifornia</td>
        <td>1914-17</td>
        <td>Explozív kitörés</td>
    </tr>
    <tr>
        <td>Mt. Hood</td>
        <td>Oregon</td>
        <td>1790</td>
        <td>Piroklaszt-torlóár és lávafolyás</td>
    </tr>
    <tr>
        <td>Mt .St. Helens</td>
        <td>Washington</td>
        <td>1980</td>
        <td>Explozív kitörés</td>
    </tr>
</table>
```

Ez a kód a következőképpen jelenik meg:

Vulkán neve	Hely	Utolsó nagyobb kitörés	Kitörés típusa
Mt. Lassen	Kalifornia	1914-17	Explozív kitörés
Mt. Hood	Oregon	1790	Piroklaszt-torlóár és lávafolyás
Mt .St. Helens	Washington	1980	Explozív kitörés

Lássuk, milyen részekből áll a fenti kódban látható HTML jelölés:

- *<table></table>*: A table tag mondja meg a böngészőnek, hogy a benne található tartalmat táblázatos formában szeretnéd elrendezni.
- *<tr></tr>*: A *tr* elemmel hozhatsz létre egy sort a táblázatban. Ezáltal a böngésző tudni fogja, hogy minden tartalmat a *<tr>* és *</tr>* tagek között vízszintesen rendezzen el, a táblázat egy sorában.

- *<td></td>*: A td elemmel határozhatsz meg egy cellát a táblázatban, vagy más önálló tartalmat a soron belül. Figyelj arra, hogy mindig csak annyi *td* elemet használj a cellákhoz, amennyire az adatoknak szüksége van. Ne használj üres cellákat azért, hogy kitöltsd a helyet, vagy távolabb tedd egymástól a cellákat — ilyen esetekben használhatod a CSS-t, amellyel könnyen készíthetsz kitöltéseket a táblázatban. Ez nem csak azért jobb módszer, mert szétválasztja a megjelenést az adatoktól, hanem a táblázat struktúrája is érthetőbb lesz például a gyengénlátók számára, akik képernyő-felolvasóval férnek hozzá a táblázatod tartalmához.

Az alap elemeket a következőképpen ágyazhatod egymásba:

```
<table>
    <tr>
        <td>content</td>
        <td>content</td>
        <td>content</td>
    </tr>
</table>
```

Ha más sorrendben használod őket, akkor az a böngésző számára olyan, mint egy hajcsomó: megpróbálhatja kigubancolni a kódot, de az eredmény nem garantált, sőt az sem biztos, hogy egyáltalán megjelenik a táblázat.

4.5.2 Adjunk hozzá több funkciót

Most, hogy már megvan az alap táblázatunk, adjunk hozzá néhány komplexebb funkciót is — először is adunk neki egy címet és az oszlopoknak egy fejlécet, amelyek javítják a táblázat szemantikáját, és egyúttal megkönnyíti a képernyő-felolvasókat használók életét is. A módosított táblázat így néz ki:

```
<table>
    <caption>A legutóbbi nagyobb vulkánkitörések Észak-
Amerikában</caption>
    <tr>
        <th>Vulkán neve</th>
        <th>Hely</th>
        <th>Utolsó nagyobb kitörés</th>
        <th>Kitörés típusa</th>
    </tr>
    ...
```

Ez a kód a böngészőben így jelenik meg:

Vulkán neve	Hely	Utolsó nagyobb kitörés	Kitörés típusa
Mt. Lassen	Kalifornia	1914-17	Explozív kitörés
Mt. Hood	Oregon	1790	Piroklaszt-torlóár és lávafolyás
Mt .St. Helens	Washington	1980	Explozív kitörés

A következő új elemeket használtuk ebben:

- *<caption></caption>*: A *caption* elemmel adhatsz egy címet a táblázat adatainak. A legtöbb böngésző alapesetben a címet középre helyezi, és a szélessége akkora lesz, mint a táblázaté, kivéve, ha ezt átállítod CSS-ben.
- *<th></th>*: A *th* elem jelöli a táblázat oszlopainak a fejlécét. Ez azért hasznos, mert szemantikusan is jelöli a tartalom funkcióját, valamint segít abban, hogy a böngészők és a különböző eszközök pontosabban jelenítsék meg a tartalmat. A fenti példa a th elem legegyszerűbb használatát mutatja be.

4.5.3 Alakítsuk tovább a táblázatot

A táblázat strukturálásában utolsó lépésként megadom a táblázat fejléc- és törzsszakaszait, hozzáadunk egy láblécet, valamint megadjuk a sorok és oszlopok hatókörét. Hozzáadunk még egy summary attribútumot is, amelyben összefoglaljuk a táblázat tartalmát. A végleges jelölés így a következőképpen néz ki:

```
<table summary="összefoglaló a legnagyobb vulkánkitörésekről
Észak-Amerikában">
    <caption>A legutóbbi nagyobb vulkánkitörések Észak-
Amerikában</caption>
    <thead>
        <tr>
            <th scope="col">Vulkán neve</th>
            <th scope="col">Hely</th>
            <th scope="col">Utolsó nagyobb kitörés</th>
            <th scope="col">Kitörés típusa</th>
        </tr>
    </thead>
    <tfoot>
        <tr>
            <td colspan="4">Készítette Ms. Jen 2008-ban</td>
        </tr>
    </tfoot>
    <tbody>
        <tr>
            <th scope="row">Mt. Lassen</th>
            <td>Kalifornia</td>
            <td>1914-17</td>
            <td>Explozív kitörés</td>
        </tr>
        <tr>
            <th scope="row">Mt. Hood</th>
            <td>Oregon</td>
            <td>1790</td>
            <td>Piroklaszt-torlóár és lávafolyás</td>
        </tr>
        <tr>
            <th scope="row">Mt .St. Helens</th>
            <td>Washington</td>
            <td>1980</td>
            <td>Explozív kitörés</td>
        </tr>
    </tbody>
</table>
```

Ez a kód a böngészőben így jelenik meg:

Vulkán neve	Hely	Utolsó nagyobb kitörés	Kitörés típusa
Mt. Lassen	Kalifornia	1914-17	Explozív kitörés
Mt. Hood	Oregon	1790	Piroklaszt-torlóár és lávafolyás
Mt .St. Helens	Washington	1980	Explozív kitörés
Készítette Ms. Jen 2008-ban			

Most a következő új elemeket és attribútumokat használtuk fel:

A *thead*, *tbody* és *tfoot* elemek: Ezek határozzák meg a táblázat fejlécét, törzsét, valamint a láblécét. Hacsak nem olyan komplex táblázatot készítesz, amelyiknek sok oszlopa és sora van, ezeknek az elemeknek a használata már túlzásnak számít. A komplex táblázatokban viszont a használatukkal strukturáltabbá teheted a kódot, és megkönnyíted a böngésző és más eszközök dolgát is a táblázat értelmezéséhez.

A *colspan* és *rowspan* attribútumok: A *colspan* attribútum olyan cellát hoz létre, amelyik több oszlopot is áthidal. A fenti esetben a lábléc egyetlen cellájára azt szerettem volna, ha kitölti a teljes szélességét a táblázatnak, ezért azt adtam meg, hogy mind a négy oszlopot foglalja magába. Hasonlóan adhatod meg egy cellának a *rowspan* attribútumot, amellyel így megadhatod, hogy hány sort vonjon össze a cella, például *<td rowspan="3">*.

A *summary* attribútum: Ezt az attribútumot arra használhatod, hogy megadj egy összefoglalót a táblázat tartalmáról, főként a képernyő-felolvasók számára (ugyanis a táblázat megjelenítésében ez nem látható). A régebbi W3C ajánlások, a WCAG 1.0 és a HTML 4.0 azt írja, hogy a summary attribútumot a fent bemutatott módon használhatod. A specifikációk új tervezetei viszont már egyáltalán nem említik a summary attribútumot. Mivel még nincs eldöntve a summary attribútum használata, ezért mi a Webes szabványok sorozat keretein belül úgy döntöttünk, hogy a használata továbbra is biztonságosnak számít. Végül is nem ronthat el semmit, és hozzáférhetőségi szempontból előnyös.

A *scope* attribútum: Biztosan észrevetted a *th* tagekben a *scope* attribútumot, valamint azt, hogy ezúttal a vulkánok neveit is fejlécként definiáltam, a sorokon belül! Ez abszolút megengedett, de én nem inkább erőltetem. A *scope* attribútumot a *th* elemben használjuk arra, hogy a képernyő-felolvasók tudják, hogy a *th* tartalma egy oszlopra vagy egy sorra vonatkozik-e. A *scope* attribútumot csak a *th* elemekben használhatjuk.

4.5.4 Segít a CSS: legyen szebb táblázatunk

A fent bemutatott elemek és attribútumok már elegendőek arra, hogy egy jó adattáblát készíthess. Most, hogy a HTML struktúra már megvan, néhány egyszerű CSS tulajdonsággal sokat javíthatunk a táblázat megjelenésén:

```
body {
  background: #ffffff;
  margin: 0;
  padding: 20px;
  line-height: 1.4em;
  font-family: tahoma, arial, sans-serif;
  font-size: 62.5%;
}

table {
  width: 80%;
  margin: 0;
  background: #FFFFFF;
  border: 1px solid #333333;
  border-collapse: collapse;
}

td, th {
  border-bottom: 1px solid #333333;
  padding: 6px 16px;
  text-align: left;
}

th {
  background: #EEEEEE;
}

caption {
  background: #E0E0E0;
  margin: 0;
  border: 1px solid #333333;
  border-bottom: none;
  padding: 6px 16px;
  font-weight: bold;
}
```

Ha ezt alkalmazzuk az előző táblázatunkon, akkor a táblázat a 4.7. ábrához hasonlóan fog megjelenni.

A legutóbbi nagyobb vulkánkitörések Észak-Amerikában			
Vulkán neve	**Hely**	**Utolsó nagyobb kitörés**	**Kitörés típusa**
Mt. Lassen	Kalifornia	1914-17	Explozív kitörés
Mt. Hood	Oregon	1790	Piroklaszt-torlóár és lávafolyás
Mt .St. Helens	Washington	1980	Explozív kitörés
Készítette Ms. Jen 2008-ban			

4.7. ábra: A táblázat most már sokkal jobban néz ki

Ó, igen... jobban néz ki. A táblázatodat úgy jelenítheted meg, ahogyan csak akarod, de a fenti példa jó kiindulási alapnak szolgálhat. A táblázatok stílusozásáról CSS-sel egy későbbi leírásban többet is megtudhatsz, jelenleg elég lesz az is, ha megnézzük, hogy a CSS melyik része mire szolgál:

body: A fenti példában hozzáadtam a CSS-hez néhány tulajdonságot. Beállítottam a margót (ebben az esetben nullára), a kitöltést, hogy legyen egy kis hely, a háttérszínt (fehérre), a betűméretet és a betűcsaládot, valamint a sormagasságot a levegősebb sorok kedvéért. A fenti példakódot innen töltheted le — változtasd meg nyugodtan a CSS tulajdonságokat, és próbáld ki, mi hogyan működik.

table: Beállítottam a kereteket a CSS border tulajdonságával. Ahhoz, hogy ez jól működjön, be kellett állítanom a *border-collapse* tulajdonságot is a collapse értékre, hogy összevonjam a táblázat kereteit, és hogy később a *border-bottom* tulajdonság a teljes sorra vonatkozzon, és ne szakadjon meg minden cellánál. A szélességet 80%-ra állítottam (ez azt jelenti, hogy a táblázat a rendelkezésre álló szélesség 80%-át tölti ki; ha változtatjuk a böngésző szélességét, akkor a táblázat szélessége is változni fog).

th és *td*: A fenti példa CSS-ben először balra igazítottam a szöveget, de te állíthatod középre is, sőt ha a *th*, *td* és *tr* elemekhez osztályneveket rendelsz, akkor a CSS-ben külön állíthatod a sorok, oszlopok, fejlécek stílusát. Mind a *th*, mind a *td* elemekhez adtam egy kevés kitöltést, hogy szélesebbek, és ezáltal olvashatóbbak legyenek a sorok. A fenti *th* szelektornál beállítottam egy eltérő színt is, hogy elválasszam a címeket a táblázat többi részétől.

caption: Ha a *caption* szelektornak nem változtatod meg a tulajdonságait CSS-ben, akkor nem kap keretet, és a háttere ugyanolyan színű lesz, mint a lap többi részének a háttere, annak ellenére, hogy a HTML jelölés a *caption* elemhez már a *table* elemen belül van. Ezért a fenti példában adtam a *caption* elemnek egy keretet (a keret alsó része hiányzik, mivel azt a táblázat kerete már biztosítja), egy saját háttérszínt, valamint félkövér megjelenést, hogy elválasszam a táblázat fejlécétől.

4.5.5 Tesztkérdések

- Készíts egy egyszerű táblázatot, amely csak 3 fő elemet tartalmaz: table, tr és td. Mentsd el, és nézd meg egy böngészőben.
- Adj hozzá a táblázatodhoz még néhány elemet: egy címet, egy fejlécet és egy láblécet. Hogyan változik a megjelenés a böngészőben?
- Mit tehetsz azért, hogy a táblázat jobban hozzáférhető legyen a képernyő-felolvasókban és a mobil eszközökön?
- Készíts a táblázathoz egy CSS fájlt. Próbáld meg beállítani a CSS-ben a keretek, a kitöltés, a cellák stílusát a táblázathoz úgy, hogy a HTML jelölést nem változtatod meg. Add meg a háttérszínt is, és állítsd be a betűk stílusát.

4.6. HTML űrlapok — az alapok

Mindenki látott már űrlapot. Minden használt már űrlapot. De kódoltál már ilyet valaha?

Az online világban űrlapnak nevezünk minden olyan részt a weblapokon, ahová információt tudsz begépelni, például amikor szöveget és számokat írsz egy szövegmezőbe, amikor bejelölsz egy mezőt, amellyel ki tudsz választani egy pontot, vagy kiválasztasz egy lehetőséget egy listából. Az űrlap tartalmát ezután egy gombra kattintva küldheted el a weblapnak.

A weben nagyon sok helyen találkozhatsz űrlapokkal, például amikor meg kell adnod a felhasználóneved és a jelszavad egy belépéshez, ha hozzászólsz egy blogon, kitöltöd a profilodat egy közösségi oldalon, vagy amikor megadod a fizetési adatokat egy vásárlás során.

Űrlapot könnyű készíteni, de mi a helyzet a webes szabványoknak megfelelő űrlapokkal? Mostanra már remélhetőleg sikerült meggyőznünk arról a tanfolyam korábbi részeiben, hogy a követendő út mindig a szabványos megoldás kell legyen. Szerencsére a hozzáfér-

hető, szabványos kód megírása egyáltalán nem igényel nagyobb ráfordítást, mint egy akármilyen űrlap összedobása.

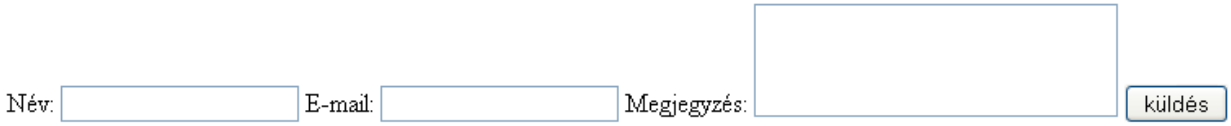
Szóval kezdjük egy alapvető és egyszerű űrlap készítésével, amelyet már használni lehet, és ebből készítsünk egy összetettebb űrlapot. Ebben a leírásban bemutatjuk az alapokat, amelyek segítségével már képes leszel egy elegáns, hozzáférhető HTML-alapú űrlapot készíteni.

4.6.1 Első lépés: Az alap kód

Kezdjük egy valóban egyszerű hozzászólás űrlappal, amellyel lehetővé teszed a látogatóidnak, hogy megjegyzést fűzzenek a tartalomhoz, visszajelzést adjanak a cikkedre, vagy kapcsolatba lépjenek veled anélkül, hogy tudnák az e-mail címed. A kód a következőképpen néz ki:

```
<form>
  Név: <input type="text" name="nev" id="nev" value="" />
  E-mail: <input type="text" name="email" id="email" value="" />
  Megjegyzés: <textarea name="megjegyzes" id="megjegyzes"
cols="25" rows="3"></textarea>
  <input type="submit" value="küldés" />
</form>
```

Ha ezt beírod egy HTML dokumentumba, majd megnyitod a böngészőben, akkor a kód úgy jelenik meg, ahogy az a 4.8. ábrán látható:



4.8. ábra: Az első, egyszerű űrlap példa

Ha megnézed a kódot, akkor az elején láthatsz egy nyitó *<form>* taget, a végén egy záró *</form>* taget, és a kettő között több elemet. Ezek között van két beviteli mező, ahol a látogató megadhatja a nevét és az e-mail címét, valamint egy szövegmező, amiben egy megjegyzést küldhet az oldal tulajdonosának.

Lássuk, mit is találunk itt:

form

<form></form>: Ez a két tag nélkülözhetetlen egy űrlap készítéséhez — ezek nélkül nem tudsz webes űrlapot létrehozni. Minden webes űrlap a *<form>* taggel kezdődik és a *</form>* taggel ér véget.

A <form> tagnek van néhány attribútuma is, amelyeket a következő lépésben fogunk bemutatni. Fontos tudni azt is, hogy nem ágyazhatsz egymásba több különböző űrlapot.

input

<input> (vagy XHTML doctype esetén *<input />*): Ez a tag adja meg azt a területet, ahol információt írhatsz be. A fenti példában az input tagek definiálják a beviteli mezőket, ahová a látogatók beírhatják a nevüket és az e-mail címüket.

Minden *input* tagnek kell legyen egy *type* attribútuma, amely megadja a mező típusát: a lehetséges értékek *text* (szöveg), *button* (gomb), *checkbox* (jelölőnégyzet), *file* (fájl), *hid-*

den (rejtett), *image* (kép), *password* (jelszó), *radio* (választógomb), *reset* (visszaállítás) és *submit* (küldés).

Minden <*input*> tagnek kell legyen egy neve (kivéve néhány speciális esetben, amikor a *value* attribútum mindig ugyanarra van állítva, mint a *type* attribútum, például a *tpye="submit"* vagy *"reset"* esetében), amelyet te állíthatsz be kedved szerint. A *name* attribútum adja meg az űrlap elküldésekor az adatokat fogadó oldalnak (ami lehet egy adatbázis, vagy egy szerver-oldali szkripten keresztül küldött e-mail az adminisztrátornak), hogy mi a neve az *input* mezőben megadott információnak. Az űrlap elküldésekor a szkriptek általában a *name* attribútumot használják fel arra, hogy az adatot elhelyezzék egy adatbázisban, vagy olvasható formában elküldjék egy személynek.

Éppen ezért, ha az input elem célja az, hogy a látogató megadhassa benne a nevét, akkor a *name* attribútum ennek megfelelően lehet *name="nev"* vagy *name="csaladnev"*. Ha az *input* elem egy e-mail cím megadására szolgál, akkor a *name* attribútum legyen *name="email"*. Hogy megkönnyítsd a saját és az űrlapot feldolgozó személy dolgát, mindig szemantikusan nevezd el az *input* elemeket.

A szemantikus alatt itt azt értem, hogy a szerint nevezd el, hogy mi az elem funkciója, amint azt a fenti példában láthatod. Ha az input egy e-mail címet vár, akkor nevezd el ennek megfelelően *name="email"* formában. Ha egy utcacímet kérsz a látogatótól, akkor nevezd el *name="utcanev"* formában. Minél pontosabban nevezed el őket, annál könnyebb dolgod lesz később egy esetleges frissítés vagy módosítás során, ráadásul ezzel megkönnyíted a fogadó adatbázis vagy személy dolgát is, hogy könnyebben megértse a kapott űrlapot. Gondolkodj egyszerűen és törekedj a pontos elnevezésre.

value

Minden <*input*> tagnek kell legyen egy *value* (érték) attribútuma. Ezt állíthatod üresre — *value=""* —, ami azt jelzi a feldolgozó szkriptnek, hogy egyszerűen állítsa be, amit a felhasználó beírt a mezőbe. A jelölőnégyzetek, rádiógombok, rejtett-, küldés- és más típusú attribútumok esetén itt adhatod meg azt az értéket, amelyet alapesetben a mezőnek adni akarsz. Más esetekben, például küldés- és rejtett mezők esetében megadhatod azt az értéket, amely az elküldött érték lesz. Például *value="yes"* az igen gomb, *value="submit"* a küldés gomb, *value="reset"* a visszaállítás gomb, vagy *value="http://www.opera.com"* a rejtett átirányítás esetében.

Néhány példa a value attribútum használatára:

Lássunk egy üres értéket, amikor a felhasználó adja meg majd az attribútum értékét:

- A kód így néz ki: <input type="text" name="keresztnev" id="keresztnev" value="" />
- A felhasználó ezt írja be: Katalin
- Az űrlap elküldésekor a keresztnev mező értéke „Katalin” lesz.

Egy előre beállított érték:

- A kód így néz ki: *<input type="checkbox" name="levelezo-lista" id="levelezo-lista" value="no" />*
- A felhasználó bejelöli a jelölőnégyzeten, hogy csatlakozni szeretne a levelezőlistához.
- A „levelezo-lista” értéke az űrlap elküldésekor *yes* lesz.

textarea

A két <input> elem után találhatsz valami mást — a *textarea* elemet.

A *textarea* egy szép, új, továbbfejlesztett szövegmezőt bocsát a rendelkezésedre a szövegek megadásához. Nem csak egy egyszerű, egysoros szövegmezőt, mint amit a barátja, az input biztosít. A *textarea* elem több soros bevitelt tesz lehetővé, sőt azt is megadhatod, hogy hány sorban adhasd meg a szövegedet a beviteli mezőben. Figyeld meg a *cols* és *rows* attribútumokat — ezeket minden *textarea* elemben meg kell adnod, mivel ezekkel definiálhatod, hogy hány oszlopa és hány sora legyen a beviteli mezőnek. Az értékek karakterszámra vonatkoznak.

submit

Végül, de nem utolsósorban van egy speciális input elem is egy *value="submit"* attribútummal. Ahelyett, hogy ez egy egysoros beviteli mezőt készítene, az elem egy küldés gombot hoz létre, amelyre ha rákattint a felhasználó, akkor az elküldi az űrlapot arra a címre, amelyet az űrlap létrehozásakor megadtunk (jelen esetben ezt még egyáltalán nem adtuk meg, így az űrlap elküldésekor nem történik semmi).

4.6.2 Második lépés: Struktúra és funkció

Szóval, eddig már kipróbáltad a fenti példa űrlapot, kitöltötted és megnyomtad a küldés gombot — de miért nem történt semmi, és miért néz ki ennyire rondán, miért van az egész egy sorban? A válasz az, hogy még nem strukturáltuk, és nem adtuk meg azt a helyet, ahová az űrlap az adatokat a kitöltés után elküldhetné.

Most térjünk vissza a tervezőasztalhoz:

```
<form id="contact-form" action="script.php" method="post">
    <input type="hidden" name="redirect"
value="http://www.opera.com" />
    <ul>
        <li>
            <label for="nev">Név:</label>
            <input type="text" name="nev" id="nev" value="" />
        </li>
        <li>
            <label for="email">E-mail:</label>
            <input type="text" name="email" id="email"
value="" />
        </li>
        <li>
            <label for="megjegyzes">Megjegyzés:</label>
            <textarea name="megjegyzes" id="megjegyzes"
cols="25" rows="3"></textarea>
        </li>
        <li>
            <input type="submit" value="küldés" />
            <input type="reset" value="visszaállítás" />
        </li>
    </ul>
</form>
```

Ez az űrlap a 4.9. ábrán látható módon jelenik meg a böngészőben:

4.9. ábra: A második űrlap példa — már jobban néz ki, de még nem tökéletes

Néhány módosítást tettem az eredeti, egyszerű űrlapoz képest. Nézzük meg ezeket egyenként:

id

Néhány új attribútum került be a <*form*> tag mellé. Hozzáadtam egy *id* attribútumot, nem csak a szemantikai okokból, hogy az űrlapnak legyen egy neve, hanem azért is, hogy legyen egy egyedi azonosítója, amellyel könnyebben lehet stílust rendelni hozzá CSS-ből, vagy módosítani rajta valamit JavaScriptben. Minden ilyen azonosítóból csak egy darab lehet egy adott névvel a lapon belül; jelen esetben az űrlap neve *contact-form* lett.

method

Fény, kamera, próba! Amikor az első űrlapon megnyomtad a küldés gombot, és nem történt semmi, az azért volt, mert nem adtunk meg hozzá semmilyen műveleted vagy metódust. A *method* attribútum szabja meg, hogy hogyan lesznek elküldve az adatok a feldolgozó szkriptnek. A két leggyakoribb metódus a „GET” és a „POST”. A „GET” metódus az adatokat a lap URL-jében küldi át (néha láthatsz ilyen hosszú URL-eket, mint például

```
http://www.pelda.hu/page.php?data1=ertek1&data2=ertek2...;
```

Ezek mind olyan adatok, amelyek a „GET” metódussal továbbítódnak). Ha nincs konkrét indokod a „GET” metódus használatára, és érzékeny információkat akarsz továbbítani, akkor inkább ne használd ezt, mivel az URL-ből bárki láthatja ezeket az értékeket. A „POST” metódus az adatokat az űrlap szkriptjén keresztül küldi, vagy egy e-mailben az oldal adminisztrátorának, vagy az adatokat egy később hozzáférhető adatbázisba mentve, nem pedig az oldal URL-jében, mint a „GET”. A „POST” éppen ezért jóval biztonságosabb, így általában ez a jobb lehetőség.

Ha számodra nagyon fontos az űrlap adatainak biztonsága, például hitelkártya számokat küldessz át egy kereskedő oldalról, akkor neked érdemes a https protokollt használnod, amely támogatja az SSL-t (Secure Socket Layert). Alapvetően ez csak annyit jelent, hogy az adatok a https protokollon keresztül lesznek elküldve, nem pedig a *http* protokollon. Legközelebb, amikor fizetsz valamiért a neten vagy online bankolsz, nézd meg az oldal URL-jét — valószínűleg a címe a *https://* jelöléssel fog kezdődni, nem a megszokott *http://* jelöléssel. A *https* kapcsolat valamivel lassabb, mint a http, viszont az adatok titkosítva vannak, így ha valaki lehallgatja az adatfolyamot, képtelen lesz bármilyen értelmes információt kinyerni belőle. Lépj kapcsolatba a tárhely szolgáltatóddal, hogy tudnak-e neked https-et és SSL-t biztosítani.

action

Az *action* attribútum adja meg, hogy melyik szkript fájl számára legyenek elküldve az adatok feldolgozásra. Sok tárhely szolgáltató biztosít egy alap e-mail küldő vagy más hasonló szkriptet (nézd meg a tárhely leírását), amelyeket a szervereikhez állítottak be. Másrészről használhatsz olyan szerver-oldali szkripteket is, amelyeket te vagy valaki más hozott létre az űrlap feldolgozására. Sok esetben ilyen célokra PHP, Perl vagy Ruby szkripteket használnak az űrlap adatainak feldolgozásához — például küldhetsz egy e-mailt, amely az űrlap adatait tartalmazza, vagy beírhatod az adatokat egy adatbázisba, későbbi felhasználás céljából.

A szerver-oldali szkriptek írásával később fogunk foglalkozni.

hidden

A második sor, amit az új űrlaphoz hozzáadtunk, egy „rejtett” (*hidden*) beviteli mező — ami egy átirányítás. Mi van?

Abból a célból, hogy szétválasszuk a jelölés struktúráját a megjelenéstől és a működéstől, érdemes úgy megadni az űrlapot feldolgozó szkriptet, hogy az átirányítsa a felhasználót az űrlap elküldésekor. Valószínűleg nem akarod ilyenkor magukra hagyni a felhasználóidat bámulni az űrlapot, és azon gondolkodni, hogy most mégis mihez kezdjenek; gondolom te is egyetértesz, hogy jobb a felhasználót átirányítani egy köszönő oldalra, ahol egyúttal biztosíthatsz neki néhány „következő” linket is, miután elküldte az űrlapot. Ez a sor pontosan ezt teszi: miután az űrlapot elküldték, a felhasználót átirányítja az Opera főoldalára.

Rendezetlen lista

Az űrlap megjelenésének javításához az összes űrlapelemet egy rendezetlen listába tettem, így felhasználhatom ezt a jelölést, hogy világosan szétválasszam őket, majd CSS-ben megadjam a megjelenésüket.

Néhányan talán ellenkeznek, hogy az űrlap elemeit nem egy rendezetlen listában, hanem egy definíciós listában kellene megadni. Mások azt mondhatják, hogy egyáltalán ne tegyük bele az elemeket semmilyen listába, hanem használjuk CSS-ben a <*label*> és <*input*> tageket. Én nem akarok erről senkivel vitatkozni, hanem rád hagyom, hogy eldöntsd, szerinted melyik megoldás helyesebb szemantikailag. Ehhez az egyszerű példához én rendezetlen listát használtam.

label

Végezetül, a második űrlapon már elneveztem az űrlap elemeit. Mind az érhetőség, mind a széleskörű hozzáférhetőség szempontjából nagyon jó ötlet, hogy neveket adj az űrlap elemeinek — a *label* elem segítségével — mivel ezek hozzákapcsolódnak a megfelelő *textarea* és *input* elemekhez, az egyedi azonosítókon keresztül (az *id* attribútumban), amelyeknek az értéke megegyezik a *label* elem *for* attribútumával. Ez nem csak azért jó, mert vizuálisan jelöli a különböző űrlapmezők funkcióját a képernyőn, hanem szemantikailag is értelmet ad a mezőknek. Például így egy képernyő-felolvasót használó látogató is pontosan tudni fogja, hogy melyik űrlap elem mit takar. Az *id* értékeket pedig felhasználhatod a CSS-ben a különböző mezők stílusozására is.

Talán elgondolkodtál azon, hogy miért adtunk meg az űrlap elemekben egyszerre egy id és egy name attribútumot is. A válasz az, hogy a name attribútum nélküli input elemek nem lesznek elküldve a kiszolgálóra, így azokra mindenképp szükség van. Az id attribú-

tumokra pedig azért van szükség, hogy összekapcsoljuk az űrlap elemeket a megfelelő label elemekkel. Ezért mindkettőt használnunk kell.

A második űrlap már valamivel szebben jelent meg, de még mindig egyszerű, mint egy bot. Ideje hozzáadni néhány varázsbitet, hogy igazi stílust kapjon.

4.6.3 Harmadik lépés: Szemantika, stílus és még egy kis struktúra

Most pedig befejezzük, amit a cikk elején elkezdtünk, a példa űrlapunk végleges változatával:

```
<form id="contact-form" action="script.php" method="post">
  <fieldset>
    <legend>Kapcsolat:</legend>
    <ul>
      <li>
        <label for="nev">Név:</label>
        <input type="text" name="nev" id="nev" value="" />
      </li>

      <li>
        <label for="email">Email:</label>
        <input type="text" name="email" id="email" value="" />
      </li>

      <li>
        <label for="megjegyzes">Megjegyzés:</label>
        <textarea name="megjegyzes" id="megjegyzes" cols="25"
rows="3"></textarea>
      </li>

      <li>
        <label for="levelezo-lista">Feliratkozik a
levelezőlistánkra is?</label>
        <input type="checkbox" checked="checked" id="levelezo-
lista" value="Igen, feliratkozom!" />
      </li>

      <li>
        <input type="submit" value="küldés" />
        <input type="reset" value="visszaállítás" />
      </li>
    </ul>
  </fieldset>
</form>
```

Ha ezt megnyitod a böngészőben, akkor a 4.10. ábrán látható képet kapod.

4.10. ábra: A harmadik, végleges példa űrlap, teljes pompájában

A két nagyobb elem, amit most hozzáadtunk az űrlaphoz, az a *fieldset* és a *legend*. Egyik elemet sem kötelező használni, de nagyon hasznosnak a komplex űrlapok és prezentációk esetében.

A *fieldset* elem segítségével szemantikus egységekbe rendezheted az űrlapot. Egy bonyolultabb űrlapon például használhatsz különböző *fieldset-eket* a címekhez, a számlázási adatokhoz, a felhasználói szokásokhoz, és így tovább. A *legend* elemmel pedig elnevezheted a különböző *fieldset* részeket.

Ezen kívül most CSS-t is adtunk az űrlaphoz, hogy stílust kapcsoljunk a jelöléshez. Két dolgot tettem meg a CSS-ben: hozzáadtam egy kis margót a *label* és *input* dobozokhoz, valamint kitöröltem a lista pontjait. A külső stíluslapon ez a kód szerepel:

```
#contact-form fieldset {width:40%;}
#contact-form li {margin:10px; list-style: none;}
#contact-form input  {margin-left:45px; text-align: left;}
#contact-form textarea {margin-left:10px; text-align: left;}
```

Mit is csinál ez pontosan? Az első sor a *fieldset* keretét állítja be, hogy ne foglalja el a teljes lapszélességet; azt is beállíthatod, hogy egyáltalán ne legyen kerete a *{border: none;}* tulajdonsággal. A második sor egy 10 pixeles margót állít be az li elemeknek, hogy egy kis helyet teremtsen a listaelemek között. A harmadik és a negyedik sor egy bal margót állít be az *input* és *textarea* elemeknek, hogy ne folyjanak össze a nevekkel, és egymás alá kerüljenek.

4.11. ábra: Űrlap stílusokkal kiegészítve

A harmadik űrlaphoz hozzáadtam egy jelölőnégyzetet is, hogy megmutassam, hogyan használhatod az input elem attribútumait az egyszerű és a többsoros szövegmezőn túl. A checkbox és a radio button attribútum értékek használatával lehetőséged van egyszerű kérdések feltevésére, és egy sor lehetséges válasz megadására is az űrlapon belül (a jelölőnégyzetekkel egyszerre több választ is ki lehet jelölni, a választógombokkal csak egyet). A választógombok különösen hasznosak például közvélemény kutatáskor.

A select elem — amelyről nem volt szó a leírásban — használható egy több elemet tartalmazó lenyíló menü létrehozására (például az országok listájával).

4.6.4 Tesztkérdések

Itt az ideje, hogy elkészítsd a saját űrlapodat.

- Készíts egy egyszerű kapcsolat űrlapot, amelyben bekéred a felhasználó nevét, e-mail címét, valamint egy hozzászólást.
- Adj hozzá egy jelölőnégyzetet (checkbox), amellyel megkérdezed a felhasználót, hogy fel akar-e iratkozni a levelező listádra.
- Készíts CSS-t az űrlap stílusozására: állítsd be az űrlap szélességét, igazítsd a neveket balra, adj meg egy háttérszínt, stb.

Pluszponért: minél többet játszol az űrlapelemekkel és a különböző CSS tulajdonságokkal, annál többet tanulsz meg abból, hogy mit lehet tenni egy egyszerű űrlappal.

További pluszpontokért: ha szeretnél ismeretlen területeket is felfedezni, akkor keress egy szkriptet vagy válassz egyet azok közül, amiket a tárhely szolgáltatód biztosít, és küldd el magadnak az űrlapot. Ha a szkriptes résszel nem tudnál megbirkózni, akkor ideje összebarátkoznod egy programozóval.

4.7. Kevéssé ismert szemantikus elemek

Ebben a leírásban néhány kevéssé ismert és ritkán használt szemantikus elemet fogok bemutatni a HTML-ből. Megnézzük, hogyan lehet programozási kódot megjelölni, számítógép kimenetet jelölni, idézeteket és rövidítéseket megadni, a dokumentumok válto-

zásait megjelölni, stb. A végén még bemutatok néhány új szemantikus jelölést a HTML 5 vázlatából.

4.7.1 Kapcsolat információk kiemelése

Az address elem valószínűleg az egyik legrosszabb hírű és leginkább félreértett elem a HTML-ben. Első ránézésre azt mondanánk, hogy az „address” elnevezés alapján ez az elem leginkább postai- vagy e-mail címeket foghat közre. Ez azonban csak részben igaz.

Az address valódi jelentése az lenne, hogy kapcsolati információkat biztosítson a lap, vagy a lap nagyobb részének szerzőjéről, esetleg szerzőiről. Ez vonatkozhat névre, e-mail címre, postai címre vagy egy kapcsolati lapra való hivatkozásra is. Például:

```
<address>
  <span>Mark Norman Francis</span>,
  <span class="tel">1-800-555-4865</span>
</address>
```

A következő példában az *address* elemet a láblécben adjuk meg, és egyszerűen hivatkozunk benne egy másik lapra az oldalról. A bővebb kapcsolati információkat érdemes így egy külön lapon megadni, hogy ne ismételjük vég nélkül az oldal minden lapján.

```
<p class="footer">© Copyright 2008</p>
<address>
<a href="/contact/">Mark Norman Francis</a>
</address>
```

Természetesen, ha az oldalnak több szerzője is van, akkor ezt a formát használhatjuk ugyanígy a többi szerzőre is, különböző kapcsolati lapokra hivatkozva.

Hibás viszont az address elem használata minden más cím megadására, mint például:

```
<p> A cégünk címe: </p>
  <address>
    Opera Software ASA,
    Waldemar Thranes gate 98,
    NO-0175 OSLO,
    NORWAY
  </address>
```

(Persze ha az Opera felelősséget vállal a leírásban foglaltakért, akkor ez is helyes lehet, annak ellenére, hogy én írtam a cikket, és nem az Opera.)

Általános címekhez használhatod az ún. „microformat” elemeket annak jelölésére, hogy egy paragrafus egy címet tartalmaz. A microformatokról bővebben a dev.opera.com leírásaiban olvashatsz.

4.7.2 Programozási nyelvek és kódok

A code elemet arra használhatjuk, hogy számítógépes kódokat vagy programozási nyelvek kódjait jelöljük vele, mint például PHP, JavaScript, CSS, XML és társaik. Bekezdésen belül egy rövid kódrészletet egyszerűen tedd a code tagek közzé, mint itt:

```
<p>Nem ajánlott az eseménykezelőket, mint például
az <code>onclick</code>, közvetlenül a HTML-ben használni.</p>
```

Nagyobb, többsoros kódrészletek bemutatására előformázott blokkokat érdemes használni, amint azt a Szöveges részek megjelölése HTML-ben leírásban már megmutattuk.

Bár nincs megszabott módszer arra, hogy hogyan jelölheted meg a *code* elemben használt programozási nyelvet vagy kódot, de erre használhatod a *class* attribútumot. Gyakori módszer (sőt a HTML 5 vázlatban is említik), hogy először beírod a *language-* előtagot, majd utána a használt nyelvet. Így a fenti példa a következőképpen módosul:

```
<p>Nem ajánlott az eseménykezelőket, mint például
az <code class="language-javascript">onclick</code>,
közvetlenül a HTML-ben használni.</p>
```

Egyes programozási nyelveknek olyan nevei vannak, amelyeket nem lehet egyszerűen használni a class attribútumban. Ilyen például a C# (C-Sharp). A helyes módja a nyelv leírásának az lenne, hogy *class='language-c\#'*, de ez zavaró és könnyen elírható. Ezért inkább azt javaslom, hogy az osztálynevekben csak betűket és kötőjelet használjunk, és ilyen esetekben írjuk ki az ejtett formát. Ebben az esetben például használjuk inkább a *class='language-csharp'* elnevezést.

4.7.3 Számítógépes kimenet és bemenet megjelenítése

A *samp* és *kbd* elemekkel jelölhetjük meg egy számítógépes program kimenetét és bemenetét. Például:

```
<p>Annak a megállapítására, hogy egy számítógép csatlakozva
van-e az internethez, gyakori módszer a <code>ping</code> nevű
számítógépes program használata, amellyel ellenőrízhetjük, hogy
egy másik gép elérhető és működik.</p>

<pre><samp class="prompt">kittaghy%</samp> <kbd>ping -o
google.com</kbd>
  <samp>PING google.com (64.233.187.99): 56 data bytes

  64 bytes from 64.233.187.99: icmp_seq=0 ttl=242 time=108.995
ms

  --- google.com ping statistics ---
  1 packets transmitted, 1 packets received, 0% packet loss
  round-trip min/avg/max/stddev = 108.995/108.995/108.995/0.000
ms
  </samp></pre>
```

A *samp* elemmel egy számítógépes program kimenetét jelölhetjük. Amint a fenti példában is látható, a különböző típusú kimeneteket megjelölhetjük a *class* attribútum használatával. Arra azonban nincs széles körben elterjedt gyakorlat, hogy ezeket a típusokat hogyan nevezzük el.

A *kbd* elemmel az olyan bemeneteket jelölhetjük, amelyekben a felhasználó kapcsolatba lép a számítógéppel. Bár ez hagyományosan a billentyűzet (erre utal a „*kbd*” rövidítés is), más típusú bemenetekre is használhatjuk, például hangparancsok esetében.

4.7.4 Változók

A *var* elemet használhatjuk változók jelölésére szöveges környezetben. Ez lehet egy matematikai változó egy algebrai kifejezésben, vagy egy programkódban található változó. Például:

```
<p>Az <var>x</var> értéke 4 ebben az egyenletben:
 3<var>x</var>+2=14.</p>
```

```
<pre><code class="language-perl">
my <var>$udvozlet</var> = "Helló világ!";
</code></pre>
```

4.7.5 Idézetek

A *cite* elemet arra használjuk, hogy megjelöljük vele a közrefogott tartalom forrását — egy személy, egy könyv vagy más publikáció idézésekor, vagy általánosan egy másik forrásra való hivatkozáskor. Ilyen esetekben a forrást a *cite* elemmel jelölhetjük. Például:

```
<p>A <q>Mindent a legegyszerűbben, de nem egyszerűbben kell
csinálni</q>
mondást gyakran tulajdonítják <cite>Albert Einsteinnek</cite>,
pedig valójában csak egy körülírása egy olyan idézetnek,
amelyet jóval nehezebb megérteni.</p>
```

4.7.6 Rövidítések

Az *abbr* és az *acronym* elemeket olyan esetekben használhatjuk, amikor egy rövidítést kell jelölnünk, lehetőséget teremtve ezáltal a rövidítés feloldására anélkül, hogy megtörnénk a dokumentum formáját.

A szöveg rövidített alakja kerül az *abbr* elembe, míg a hosszabb, teljes vátlozata a *title* attribútumba. Például:

```
<p>A <abbr title="Hypertext Markup Language">HTML</abbr>
dokumentumokhoz <abbr title="Cascading Style Sheets">CSS</abbr>
használatával rendelhetünk stílust.</p>
```

Az *acronym* elemmel egy betűszót jelölhetünk, ami ugyancsak egy rövidítési forma, azzal a különbséggel, hogy a végeredmény olyan, mintha egy külön szó lenne, és úgy is kell ejteni. Ilyen például a gyes, ami a "gyermekágyi segély" rövidítése. Bár a HTML 4.01 specifikáció megengedi mind az *abbr*, mind az *acronym* elemeket is, mégis egy kis problémába ütközünk, ha helyesen akarjuk jelölni ezeket a szavakat.

Az Internet Explorer (a 7-es verzió előtt, de a 7-es sem húzza alá a rövidítéseket szaggatott vonallal, mint a többi böngésző) nem ismeri fel az *abbr* elemet, viszont felismeri az *acronym* elemet. Sajnos az *acronym* a rövidítéseknek csak egy részhalmaza, ezért helytelen például egy "HTML" rövidítést *acronym* elemmel jelölni.

A HTML 5 specifikáció vázlatában ráadásul az egyértelműbb szabvány érdekében el is hagyták az *acronym* elemet, mivel minden így jelölt szöveg egyben egy érvényes rövidítés is.

A legjobb, amint tehetsz, hogy kerülöd az *acronym* használatát, és csak az *abbr* elemet használod a kódodban. Ha vizuális stílust akarsz rendelni az *abbr* elemhez, akkor tedd bele a tartalmát még egy *span* elembe, és ehhez rendeld a stílust az *abbr* helyett, hogy minden böngésző ugyanúgy jelenítse meg az elemet. Erről később részletesebben is fogunk beszélni a Szöveg stílusozása CSS-sel leírásban.

4.7.7 Példányok definiálása

Ugyancsak zavaros a *dfn* elem helyes használata is, amit a HTML specifikáció úgy értelmez, hogy "a közrezárt kifejezés példánydefiníciója". Ez eléggé közel áll a *dt* elem értelméhez, amit a definíciós listákban használhatunk.

A különbség az, hogy a *dfn* elem nem kell része legyen egy kifejezéseket és értelmezéseket definiáló listának, hanem egyszerűen bele lehet fűzni a normál szövegfolyamba. Lássunk egy példát a *dfn* elem használatára:

```
<p><dfn>HTML</dfn>: A HTML a "HyperText Markup Language"
rövidítése. Ezt a nyelvet használják a webes
dokumentumok tartalmának leírására.</p>
```

Megjelenik benne a HTML kifejezés, majd közvetlenül utána következik a kifejezés értelmezése, így ideális hely a *dfn* elem számára. Minden kifejezésre csak egyszer használjuk a lapon belül, az első előfordulásnál, de a kifejezéseket amúgy is csak egyszer szoktuk értelmezni egy lapon, úgyhogy ez nem okozhat nagy gondot.

Ez mind szép és jó, de egy ilyen izolált példa nem túl praktikus. A dfn elemet akkor ajánlott használni, ha egy kifejezés többször is előfordul a lapon. Például A HTML alapjai leírásban több mint 50-szer fordul elő a HTML kifejezés. Az <abbr title="HyperText Markup Language>HTML</abbr> kód használata minden egyes alkalommal csak fölösleges sávszélesség-pazarlás, vizuálisan zavaró és valószínűleg igen fárasztó a képernyőfelolvasókat használóknak is, akik minden egyes előforduláskor meghallgathatják, hogy minek is a rövidítése a HTML. Ehelyett megadhatjuk ezt a kódot a HTML definiálásának a helyén:

```
<p>A <dfn><abbr>HTML</abbr></dfn> („HyperText Markup Language”,
azaz hiperszöveg-leíró nyelv) a webes dokumentumok
leírására szolgáló nyelv.</p>
```

Később, a HTML rövidítés használatakor egyszerűen jelölhetjük így: *<abbr>HTML</abbr>*. A kliens eszköz így biztosíthat egy módot a felhasználónak arra, hogy a rövidítés definícióját megmutassa. Sajnos jelenleg nincs olyan kliens eszköz, amelyik biztosítana erre valamilyen módot, beleértve a képernyő-felolvasókat is. Éppen ezért jobb módszer, ha ezt az információt a *title* attribútumban is megadod:

```
<p>A <dfn><abbr title="HyperText Markup
Language">HTML</abbr></dfn>
(„HyperText Markup Language”, azaz hiperszöveg-leíró nyelv)
a webes dokumentumok leírására szolgáló nyelv.</p>
```

Most viszont sajnos megdupláztuk a HTML rövidítés kifejtését, ami problémát okozhat a képernyő-felolvasóknak. Ha viszont kihagyjuk a vizuális megjelenítését, akkor a dokumentum kevésbé lesz hasznos a látóknak, akik a látogatók jóval nagyobb részét teszik ki az esetek túlnyomó részében.

Szerintem ez egy elfogadható megoldás, ha csak egy-két kifejezés definícióját kell megadnunk (az olyan lapokon, amelyeken sok definíciót kell megadni, jobb lehet egy külön lapot létrehozni a definícióknak, ahol a sokkal egyértelműbb definíciós lista is használható). De ha nagyon zavar a fenti megoldás, még használhatod ezt is:

```
<p>A <abbr title="HyperText Markup Language">HTML</abbr>
(<dfn>HyperText Markup Language</dfn>, azaz hiperszöveg-leíró
nyelv)
a webes dokumentumok leírására szolgáló nyelv.</p>
```

Ebben az esetben a kliens eszköznek még mindig kellene biztosítania egy módszert arra, hogy az értelmezést összekapcsolja a különböző példányokkal. Jelenleg viszont egyetlen böngésző sem tesz semmi hasznosat a dfn elemmel, így szinte csak CSS stílusozásra használhatjuk. A fenti megoldás a legjobb, amit eddig sikerült kitalálnunk.

Ez egy elég szerencsétlen eset, mivel a specifikáció anélkül készült el, hogy világos útmutatást adna arra, hogyan kellene használni pontosan az elemet, és valószínűleg nem a va-

lódi használatát vették figyelembe az elemnek, máskülönben lenne valamilyen módszer a kifejezések definíciójának és leírásának az összekapcsolására. A HTML 5 specifikáció már részletesebben foglalkozik azzal, hogy hogyan kell használni a dfn elemet, de ez még csak vázlat, és nem alkalmas a mindennapi használatra.

4.7.8 Felső- és alsó index

Ha egy szövegrészt felső- vagy alsó indexként akarsz megjelölni (vagyis a többi szövegrészhez viszonyítva kissé feljebb vagy lejjebb elhelyezni), akkor erre használhatod a sup és a sub elemeket.

Egyes nyelvekben szükség van ezekre a rövidítések helyes használatához, valamint rövid matematikai formulák leírására is használható (bonyolultabb formulákhoz már a MathML használatát javasoljuk, amely pontosan a komplex és bonyolult matematikai formulák leírására készült).

```
<p>A víz kémiai vegyjele a H<sub>2</sub>O, vagy más néven
a hidrogén-hidroxid.</p>
<p>Albert Einstein híres képlete az energia megmaradására
az E=mc<sup>2</sup> — az energia egyenlő
a tömeg és a sebesség négyzetének a szorzata.</p>
```

4.7.9 Sortörések

Amiatt, ahogy a HTML a fehér karaktereket kezeli, nem tudjuk a sorok tördelését úgy befolyásolni, hogy egyszerűen megnyomjuk az Entert a szöveg beírásakor (például ha egy bekezdésben megadunk egy postai címet, és azt szeretnénk, ha a cím minden része külön sorban jelenne meg).

A szövegbe a *br* elemmel tehetünk be egy sortörést. Viszont csak abban az esetben használjuk, amikor valóban szükség van a sortörésre, és soha ne használjuk helykitöltésre, például arra, hogy megnöveljük a távolságot két bekezdés között — ezt sokkal könnyebben és szebben megtehetjük a CSS-sel.

Például a korábban megadott Opera címet így írhatjuk le:

```
<p> A cégünk címe: </p>
  <address>
    Opera Software ASA,<br>Waldemar Thranes gate 98,<br>
    NO-0175 OSLO,<br>NORWAY
  </address>
```

Természetesen, ha XHTML-t használsz a HTML helyett, akkor az elemet le kell zárni, valahogy így:

4.7.10 Vízszintes vonalak

Egy vízszintes vonalat a HTML-ben a *hr* elemmel készíthetünk. Ez behelyez a dokumentumba egy vonalat, amely meghatározás szerint arra szolgál, hogy elválassza egymástól a dokumentum különböző szakaszait.

Sokan nem értenek egyet azzal, hogy ez valóban egy szemantikus elem, és nem pusztán vizuális, prezentációs célokat szolgál. Ennek az előzményei a nyomtatott irodalomba nyúlnak vissza, ahol egy fejezeten belül (amely egy könyv szakaszainak felel meg) egy vízszintes vonalat tettek az olyan jelenetek közé, amelyek más időben és/vagy más he-

lyen játszódtak. A verseskötetekben ugyancsak sokszor használtak ilyet a nagyobb szakaszok elválasztására.

Egyik használati mód sem ad okot egy új fejléc elem bevezetésére, amely egy elfogadott mód a dokumentum szakaszhatárainak a jelölésére.

A hr elemnek nincs semmilyen különleges attribútuma, és CSS-ben stílusozható, ha az alap megjelenése nem megfelelő.

Hasonlóan a sortöréshez, ha XHTML-t használsz HTML helyett, akkor az elemet le kell zárnod: <hr />

4.7.11 Dokumentumok változása (beillesztés és törlés)

Ha egy dokumentum az első megjelenés után megváltozott, akkor a későbbi változásokat megjelölheted, hogy a visszatérő látogatók vagy az automatikus folyamatok láthassák, mi és mikor változott.

Az új szövegeket (beszúrásokat) az *ins* elemmel jelölheted meg. Az eltávolított (törölt) szövegeket a *del* elemmel tudod megjelölni. Ha a törlés és a beszúrás ugyanazon a helyen történt a dokumentumban, akkor a javasolt módszer szerint előre kerül a törölt szöveg, majd ezt követi a beszúrás.

Mindkét elemnek van még két attribútuma, amelyekkel megadhatod a szerkesztés részleteit.

Ha a módosítás oka valahol megtalálható a lapon vagy a weben, akkor hivatkozhatsz erre a dokumentumra vagy szakaszra a *cite* attribútum segítségével. Ez egészen pontosan arra vonatkozik, hogy „ez a változás ebből az okból történt”.

Ezen kívül megadhatod azt az időpontot is, amikor a változás történt, mégpedig a *datetime* attribútum segítségével. Az érték egy ISO szabvány szerinti időpont kell legyen, amelyik a következő formátumot használja: “ÉÉÉÉ-HH-NN ÓÓ:PP:MM ±ÓÓ:PP” (további információt erről a Wikipédián találsz).

Egy példa mindkét attribútum használatára:

```
<p>Csak a felmerült problémákat kellene megoldanunk.
Amint <cite><del datetime="2008-03-25 18:26:55 Z"
  cite="/changes.html#revision-4">Donald Knuth</del><ins>
  datetime="2008-03-25 18:26:55 Z"
  cite="/changes.html#revision-4">C. A. R. Hoare</ins></cite>
  mondta: <q>az elhamarkodott optimalizálás minden
  gonosz gyökere</q>.</p>
```

4.7.12 Néhány jövőbeli HTML elem

Amint azt már többször is megemlítettük ebben és más leírásokban is, már hozzáférhető a HTML 5-ös verziójának a vázlata. Ez lesz a legradikálisabb frissítés a HTML-hez a fennállása óta. Ahelyett, hogy azon gondolkodtak volna, hogy mi lenne jó az embereknek, elemezték a jelenleg használt HTML dokumentumokat az interneten, így jó esély van arra, hogy az időközben széleskörűen elterjedt szemantikus jelöléseket beveszik a szabványba.

A következő néhány új HTML elem például jelentősen javítja majd a dokumentumok kódolását és használatát:

- *header* — a lap fejlécét tartalmazza; szokás szerint a logót és a címet, esetleg egy rövid bemutatkozó részt vagy egy globális navigációs struktúrát, mint például belépés, kilépés, profil hivatkozások.
- *footer* — a lap láblécét tartalmazza, amely szokás szerint további hivatkozásokat tartalmaz a lapra; copyright és más hivatalos információkat.
- *nav* — a lap fontosabb navigációs hivatkozásait tartalmazza.
- *article* — a lapnak azt a részét tartalmazza, amelyik a fő területen van, kivéve a lap olyan elemeit, mint a navigáció, a fejléc vagy a lácléc.
- *aside* — az oldalsáv információit tartalmazza a lap egy adott területén, de arra is jó lehet, hogy szavazásokat vagy megjegyzéseket helyezz el a fő tartalomban.

Természetesen több is van, ezeket a HTML 5 specifikációjában találod meg.

4.8. Általános tárolók — a div és a span elemek

Ebben a leírásban be fogom mutatni, hogyan és mikor használatod a HTML-ben azt a két elemet, amelyek nem a tartalom leírására szolgálnak. A div és a span elemek ugyanis semmilyen jelentést nem rendelnek ahhoz a tartalomhoz, amit közrefognak, hanem egy általános módszert biztosítanak ahhoz, hogy a tartalmat olyan saját struktúrába vagy csoportokba rendezd, amelyre nincs más, odavaló HTML elem. Az ilyen csoportokat aztán CSS-ből stílusozhatod és JavaScripttel módosíthatod is. Habár a div önmagában nem tartalmaz szemantikus jelentést a dokumentumban, mégis tekinthetjük úgy, mint egy strukturált felosztást elősegítő jelölést, a megfelelő szemantikus osztály- vagy azonosító névvel együtt.

Ezeket a tageket csak „végső esetben” használjuk, amikor már semmilyen más HTML elemet nem tudunk felhasználni, mivel ezek az elemek már semmilyen plusz információt nem hordoznak például a kisegítő technológiák, keresőmotorok számára.

4.8.1 Szemantikailag semleges

A HTML elemek legnagyobb része azért van, hogy leírjon valamilyen tartalmat, például képeket, listákat, címsorokat, vagy hogy segítsen a dokumentum felépítésében, a fejléc (*head*), a törzs (*body*), a külső hivatkozások (*link*) vagy a metaadatok (*meta*) definiálásában. A HTML specifikáció a következőt írja:

A div és a span elemek, ellentétben az id és a class attribútumokkal, egy sokkal általánosabb mechanizmus nyújtanak a dokumentumok strukturálására.

Ezekre az elemekre úgy tekinthetünk, mint a HTML állványaira. Lehetőséget adnak arra, hogy csoportosítsd a tartalmat, hogy extra információkat definiálhass a tartalmak körül, amelyeket aztán kapaszkodóként szolgálnak a JavaScript és CSS használatakor. Viszont nem adnak semmilyen új szemantikai jelentést a dokumentumhoz, sem a jelölésen belül, sem kívül.

4.8.2 Inline kontra blokk

Ahogy már korábban tanultuk, a blokk szintű elemek olyan elemek, amelyek segítenek a dokumentum strukturálásában (3.1.6. fejezet). A *div* elem — ami a division, vagyis a felosztás rövidítése — egy blokk szintű általános tároló elem. Normál esetben arra használhatjuk, hogy más blokk szintű elemeket fogjon közre, egy csoportba rendezve azokat (a

következő szakaszban bővebben is fogunk erről beszélni). Arra is használható, hogy összegyűjtsön egy adag inline elemet és/vagy szöveget, amelyek egyébként logikailag nem tartoznak egy közös blokk szintű elem alá, de ezt csak végső esetben használjuk.

A *span* az inline, vagyis szövegen belüli általános tároló elem. Segíthet abban, hogy a dokumentum struktúráját felépítsük, de általában arra használjuk, hogy összefogjon más inline elemeket és/vagy szövegeket, kivéve a blokk szintű elemeket.

A kettő közötti különbség első ránézésre eléggé elenyészőnek tűnhet. Ami alapján elsősorban különbséget tehetünk közöttük, az a tartalom típusa, valamint hogy hogyan jelenne meg, ha stílusozás nélkül írnánk le őket. A *div* elemet egy csoport blokk szintű elem köré szoktuk tenni — például összefoglalhatunk egy címsort és egy hivatkozáslistát, hogy készítsünk belőlük egy navigációs menüt. A *span* ezzel szemben inkább a csoport inline elemet és (elsősorban) egyszerű szöveges részeket fog közre. A kulcsszó mindkét esetben a „csoport”: ha egy *div* csak egyetlen blokk szintű elemet fog közre, vagy a *span* egyetlen inline elemet, akkor fölösleges a használatuk. Nézd meg például, hogy használjuk a *div* és a *span* elemeket az alábbi egyszerű kódban:

```
<body>
  <div id="foTartalom">
    <h1>A lap címe</h1>
    <p>Ez az első bekezdés a példa lapunkon.</p>
    <img src="pelda.gif"
        alt="Ez csak egy példa kép, semmi különleges">
    <p>Ez a második bekezdés a példa lapunkon. Nagyon hasonló
    az elsőhöz, csak itt van egy
    <span id="specialisFigyelmeztetes">speciális
        figyelmeztetés, amit meg akarunk színezni és a kicsit
        megnövelni CSS-sel</span>.
    Ez nem a normál kiemelés, inkább stílusozás, szóval az
    em és a strong nem a legmegfelelőbbek.</p>
  </div>
</body>
```

Mostantól a *div* és *span* elemekben található tartalmat elérheted az id attribútumaik segítségével, és speciális stílusozást és pozicionálást adhatsz nekik CSS-ben.

4.8.3 Példa a tartalom felosztására

Ha megnézed az interneten található lapok forrását, találhatsz egy csokor olyan *div* elemet, amelyeknek mind hasonló neveik vannak a *class* vagy *id* attribútumaikban, például header (fejléc), footer (lábléc), content (tartalom), sidebar (oldalsáv), és így tovább.

A *class* és *id* attribútumaid nevei legyenek szemantikusak, vagyis elsősorban a közrefogott tartalom jelentésére, funkciójára vonatkozzon, ne pedig a vizuális megjelenésére. Például az *oldalsav* és a *figyelmeztetes* jó osztálynevek, míg a *pirosbalsav* és a *kekvillogoszoveg* már nem. Mi van akkor, ha az oldalsávod színét egy későbbi időpontban át szeretnéd állítani pirosról kékre, vagy áttennéd balról jobbra? Mi van akkor, ha a figyelmeztetéseket mostantól zöld aláhúzottként akarod megjeleníteni a villogó kék helyett?

Ezek a felosztások jól átláthatóvá teszik a struktúrát a lap készítésekor, de még fontosabb, hogy amikor később megnézzük a HTML kódot, nagy segítséget nyújtanak abban, hogy melyik rész mire szolgál. Egy jól felosztott lap önmagát magyarázza.

Ahhoz, hogy ez érthetőbb legyen, nézzük meg egy valódi weboldal felosztását — mégpedig a dev.opera.com oldalét. Ne felejtsd el, hogy az alábbi példakód egyáltalán nem tartalmaz hasznos tartalmat, leszámítva azt a néhány elemet, amelyik szükséges az oldal

struktúrájának bemutatásához. Csak arra koncentráltunk, hogy felépítsük az adott oldal struktúráját, *div* elemeket felhasználva. A kódban olvasd el figyelmesen a HTML kommenteket — ezekben magyarázom meg az oldal struktúráját. Miközben végigolvasod a kódot, nyisd meg a dev.opera.com oldalt egy másik fülön vagy ablakban, így azonnal láthatod a felépített struktúraelemeket élesben is.

```
<body>
<!-- Legelőször van egy wrapper div, ami közrefogja a teljes
lapot, ezen keresztül pontosabban befolyásolhatjuk a teljes
lapot →
  <div id="wrap">
  <!-- Ez a rendezetlen lista tartalmazza a hivatkozásokat az
Opera különböző oldalaira, amelyeket a lap legfelső részében
láthatsz →
    <ul id="sitenav" class="hidemobile">
      …
    </ul>

      ...

    <!-- Ez egy keresőmező - a kereső, amit a lap jobb felső
részében láthatsz →
    <form action="/search/" method="get" id="search">
      <div>
        …
      </div>
    </form>

    <!-- Ez a rendezetlen lista tartalmazza a fő navigációs
menüjét az oldalnak - a vízszintes menüsávot, amelyet a főcím
képe alatt láthatsz →
    <ul id="menu">
      …
    </ul>

    <!-- Ezek az egymásbaágyazott div-ek adják meg a belépődoboz
struktúráját, ahol megadhatod a felhasználóvedet és a
jelszavadat a belépéshez az oldalra. Ezt csak akkor látod, ha
nem vagy belépve →
    <div id="loginbox">
      <div id="login">
        …
      </div>
    </div>

    <!-- Ezekbe az egymásbaágyazott div-ekbe kerül a lap valódi
tartalma - a cikkek rövid tartalma, amelyek a lap fő tartalmát
adják →
    <div id="content2">
      <div id="main">
        …
        <div class="major">
          …
        </div>
```

```
        <div class="major">
          …
        </div>
        …
      </div>
    </div>

    <!-- Ez a div tartalmazza a lap oldalsávját - a cikkek
kategóriáit, az utolsó hozzászólásokat, stb →
    <div id="side">
      …
    </div>

    <!-- Ez a div tartalmazza a láblécet, ahol láthatod a
copyright megjegyzést, valamint mindenféle hivatkozásokat →
    <div id="footer">
      …
    </div>
  <!-- A lap vége - itt zárjuk be az első wrapper div-et -->
  </div>
</body>
```

4.8.4 Extra információk

Néhány tartalom olyan extra információkat tartalmaz, amelyeket a különböző kliens eszközök fel tudnak használni. Az ilyen információkat valamilyen attribútummal lehet leírni. A *span* elem gyakran a legjobb megoldás az ilyen információk befűzésére, amint azt az alábbiakban bemutatjuk.

Egy jó példa az az eset, amikor valamilyen más nyelvű szövegrész jelenik meg a dokumentumban. Például:

```
<p><q>Plus ça change, plus c'est la même chose</q> — mondta.</p>
```

Bár a fő dokumentum nyelve magyar, az idézet valójában francia. Ezt a *lang* attribútum használatával adhatjuk meg:

```
<p><q lang='fr'>Plus ça change, plus c'est la même chose</q> —
mondta.</p>
```

Ebben az esetben könnyen meg tudtuk jelölni az eltérő nyelvet, mivel az egy idézetben jelent meg, így a *q* elem éppen megfelelt az idegen nyelvű tartalom közrezárására. Vannak viszont olyan esetek, amikor nem találunk egyetlen olyan elemet sem, amelyik megfelelő lenne, így kénytelenek vagyunk használni a *div* vagy a *span* elemeket. Például:

```
<p>A képernyő-felolvasó a francia chat (macska) szót mindaddig
hibásan fogja felolvasni, amíg helyesen meg nem jelöljük a szó
nyelvét.</p>
```

Ebben a példában a *chat* szó megjelenésekor valahogyan jelezni kellene a dokumentumban, hogy a képernyő-felolvasó más nyelvet használjon a kiolvasására. Ebben az esetben egy *span* elemmel a szó körül nagyon egyszerűen érhetjük el ezt, mivel semmilyen más HTML elemet nem tudunk használni helyette. Mivel csak egyetlen szóról van szó egy hosszabb szövegen belül, így inline, szövegen belüli elemre van szükség. A példát így lehetne átírni helyesen:

```
<p>A képernyő-felolvasó a francia <span lang='fr'>chat</span>
(macska) szót mindaddig hibásan fogja felolvasni, amíg helyesen
meg nem jelöljük a szót.</p>
```

Ugyanezt a technikát használják a microformatok esetében is, az adattípusok egyforma jelölésére a weblapokon belül. A microformatokról sokkal többet is megtudhatsz a dev.opera.com haladó HTML cikkeiből.

4.8.5 Kapaszkodók a JavaScriptekhez és CSS-hez

Már beszéltünk arról, hogyan használhatod a *div* és a *span* elemeken az *id* és a *class* attribútumokat arra, hogy ezeken keresztül CSS-ből közvetlenül stílusozhasd és pozicionálhasd a dokumentum bizonyos részeit. Pontosan ugyanígy kell eljárnod ahhoz, hogy JavaScripttel is elérhesd ezeket a részeket.

Ha egy bizonyos elemet megkeresni vagy módosítani szeretnél JavaScripttel, akkor az általános módszer szerint rendelj az elemhez egy *id* azonosítót, majd használd a *getElementById* függvényt a lekéréséhez. A JavaScriptről a tanfolyam későbbi részeiben lesz szó bővebben.

4.8.6 div-itis

Érdemes megemlíteni még egy gyakran előforduló problémát, amelyre „div-itis” névvel szoktak hivatkozni a webfejlesztők.

Bár nagyon egyszerű a *div* és a *span* elemek stílusozása, ezt a módszert lehetőség szerint kerülni kell. A legtöbb esetben a stílusozást vagy a JavaScript funkcionalitást a már létező elemekhez is hozzá lehet rendelni a dokumentumban. Az általános tárolók használata csak végső megoldásként merülhet fel — a legjobb, ha a weblapot megpróbáljuk csak a szemantikus elemek felhasználásával megírni, és a tárolókat csak akkor alkalmazzuk, amikor már nincs más lehetőség.

4.8.7 Helytelen szemantika

Ebben a részben néhány olyan HTML jelölésekkel kapcsolatos gyakori félreértést szeretnék bemutatni, amelyeket érdemes elkerülni, ha csak lehetséges.

Általános „bekezdések”

Néha csábító lehet egyszerűen egy *p* elembe tenni egy szöveget (bekezdésnek jelölve), de ez nem mindig helyes. Ahogy már korábban is említettem a szöveges részek megjelölése leírásban:

Ha csak néhány szóról van szó, vagy nem egy teljes mondatról, akkor ezeket nem kell feltétlenül paragrafusként megjelölnöd.

Egy div vagy span elem (a környezet függvényében) sokkal inkább megfelelő az olyan szöveges tartalmak szétválasztására, amelyek között nincs más HTML elemmel leírható kapcsolat.

Prezentációs elemek

Az interneten fellelhető tanácsok között van egy, amelyik egyértelműen káros, mégpedig a rövid, megjelenítő elemek, mint a *b* vagy az *i* használata általános tárolóként a *span* elem helyett. Általában két indokot hoznak fel a módszer mellett:

- Ezek az elemek három byte-tal rövidebbek, így sávszélességet lehet spórolni a HTML és a CSS esetében is.

- Az elemeket csak vizuális megjelenítésre használjuk, így a a „prezentációs” elemek használata ilyen esetekben megengedhető.

Az első pont az igaz, de a megspórolt mennyiség túlnyomórészt jelentéktelen (kivéve ha irdatlan mennyiségű vizuális effektust használsz), különösen a mai modern tömörítő módszerek mellett, amelyeket a webes kiszolgálók alkalmaznak a dokumentumokon a böngészőhöz való átküldés előtt. Ezek sokkal rövidebbé teszik a dokumentumokat, mint bármilyen kézileg alkalmazott trükk.

A második pont arra utal, hogy nem értették meg pontosan a prezentációs elemek jelentését a HTML-lel kapcsolatban. A prezentációs elemek azt jelölik, hogy a tartalmuk hogyan néz ki (tehát egyszerűen annyit, hogy „ez a szöveg félkövér”). Nem jelentenek viszont kapaszkodókat a bennük található szövegrészek stílusozásához.

Ha egy rövid szövegrészt egy bekezdésen belül stílusozni vagy JavaScripttel módosítani kell, és nincs hozzá illő szemantikus elem, ami közrefoghatná, akkor az egyetlen helyes elem, amit használhatunk, az a *span*.

4.8.8 Tesztkérdések

- Mi a különbség a *div* és a *span* között?
- Nevezd meg a fenti elemek 2 felhasználási módját a weblapokon.
- Nézd meg az egyik kedvenc weboldalad forráskódját. Figyeld meg a felépítését. Sok *div* és *span* elemet használ hozzá? Látsz helytelen felhasználási módokat rajtuk? Hogyan lehetne jobban megoldani?

4.9. Több lap létrehozása navigációs menüvel

http://dev.opera.com/articles/view/23-creating-multiple-pages-with-navigat/

4.10. A HTML validálása

http://dev.opera.com/articles/view/24-validating-your-html/