# 1. FEJEZET

# Hogyan írjunk XHTML és CSS kódot?

Az XHTML és a CSS két teljesen különböző állatfajta, pontosabban weblapkészítésre vonatkozó előírás. Mindkettőnek jól megkülönböztethető kinézete és célja van. Ha együtt használjuk őket, hasznos, információgazdag és szemet gyönyörködtető weblapokat hozhatunk létre.

Ha másképpen szedett példákat látunk itt vagy egy későbbi fejezetben, pontosan kell azok szövegét beírnunk, amikor végrehajtjuk a könyv gyakorlatait. A nyelv rendszerét leíró szabályok neve *nyelvtan*, idegen szóval *szintaxis*. Ebben a fejezetben az XHTML és a CSS nyelvtanát fogjuk megismerni. Megtanuljuk, hogy mi végre vannak az egyes előírások, hogyan néznek ki, és hogyan lehet egy HTML, illetve egy CSS lapot megírni. Ez a fejezet ismerteti mind az XHTML, mind a CSS alapszabályait is, például hogy mikor kell szóközt használni, mikor pontosvesszőt, vagy mikor zárójelet.

## A webhelyek felépítése

Most röviden összefoglalom, mi kerül egy webhelyre. Ez sokat segíthet azoknak, akik soha nem készítettek még webhelyet, hogy megértsék a HTML/XHTML és a CSS szerepét, és azt, hogy miként működnek ezek együtt a képernyőn megjelenő látvány kialakításában. (Aki használta már a „vizuális eszközök” valamelyikét, például a Dreamweaver vagy a FrontPage programot, az kicsit már előrébb tart a tanulás útján, de ez a kis visszatekintés nekik is jobb rálátást ad arra, hogy miről is lesz majd szó a könyvben.)

Ha bolyongtunk már a Világhálón, tudjuk, hogy a weblapokon lehet szöveg, lehetnek képek, hivatkozások, hangok, filmek és mozgóképek. Talán azzal is tisztában vagyunk, hogy egyes weblapok tartalmazhatnak különféle programnyelveken (JavaScript, PHP,

ASP stb.) írt parancsfájlokat, amelyek segítségével a weblap látogatója is beleszólhat, mi történjen a következőkben, illetve amelyek egy adatbázishoz kapcsolják a weblapot, vagy összegyűjtik az űrlapokon beküldött adatokat.

Az enyv, ami összetartja ezeket a darabokat, és segít abban, hogy a weblap tartalma a böngészőprogramok – ilyen például az Internet Explorer vagy a Netscape Navigator – ablakában olvasható vagy használható módon jelenjen meg, a HTML vagy az XHTML. A böngészőprogram a Világhálóra nyíló ablak, az XHTML pedig az a nyelv, amely tudatja a böngészővel, hogy hogyan kell formázni a weblapot alkotó összetevőket.

A CSS úgy lép be a képbe, hogy formázási stílusokkal tudja módosítani a weblap formázható elemeit. A stílus vonatkozhat színre, elhelyezésre, képekre, betűkre, térközökre, de az XHTML által megformázott, az egész weblap alapját képező részeket és összetevőket nem módosítja.

Az Internet a világ minden részéről egymáshoz kapcsolódó számítógép-hálózatok óriási halmaza. A Világháló (World Wide Web, röviden WWW) az Internet része ugyan, de nem azonos magával az Internettel. Az Internetnek a Világhálón kívül számos más része is van, ilyen például az e-mail (elektronikus levelezés).

## Mi az XHTML és a HTML?

A Hypertext Markup Language (HTML, hiperszöveges leírónyelv) egy programozási előírás, amely azt szabályozza, hogy miként kell megírni a weblapokat ahhoz, hogy a számítógépek megértsék és megfelelő módon jelenítsék meg azokat. Az XHTML az Extensible Hypertext Markup Language (bővíthető hiperszöveges leírónyelv) kifejezés rövidítése, egy olyan előírás, amelyet a HTML-ből fejlesztettek ki. Azt perceken belül meg fogjuk látni, mit jelent az „extensible” (bővíthető) szó, de a HTML és az XHTML szerepének megértéséhez meg kell előbb értenünk a HTML névben szereplő három kifejezést.

A *hiperszöveg* (hypertext) egyszerűen csak olyan szöveg, amely a hipertérben – az Interneten – található. Ez egyszerű (formázatlan) szöveg, amely a weblap tartalmát hordozza, valamint programozási információk, amelyek a weblap megjelenítéséhez és más weblapokhoz kapcsolásához szükségesek. A hiperszöveget egy *leírónyelv* (markup language) segítségével lehet formázni. Ez egy olyan szabványos jel- és kódkészlet, amelyet minden böngésző képes értelmezni.

A Világhálóra vonatkozó különféle szabványi előírások – köztük az XHTML – létrehozásával és közzétételével foglalkozó szervezet a World Wide Web Consortium, rövidítve W3C, amely a `www.w3.org` címen érhető el. Érdemes még tudni arról, hogy rajtuk kívül létezik a Web Standards Project, egy olyan önszerveződő egyesülés, amely az internetes szabványok elfogadtatásáért küzd, és a `www.webstandards.org` címen érhető el.

A *leírókód* (markup) két dolgot árul el a szövegről vagy egyéb weblaptartalomról: először is azonosítja, hogy milyen *szerkezeti felépítést* igényel a tartalom. Ha a weblapot úgy képzeljük el, mint csupán szavak halmazát, akkor a HTML az a leírókód vagy keret, amely megadja, hogy bizonyos szavak címsorok, listák vagy bekezdések legyenek. Az, ahogyan a szöveget kódokkal látjuk el, a weblapot jelentéssel bíró információkból álló részekké – például címsorokká, alcímekké és idézetekké – szervezi.

A leírókód meghatározza ezenkívül ezeknek az elemeknek a *megjelenítését* is, például a címsorokhoz és alcímekhez használt különböző betűtípusokat. Amikor első ízben kifejlesztették, a HTML volt a képernyőn való vizuális megjelenítés meghatározásának egyetlen eszköze. Amikor a Világháló elindult, a *Hypertext Transfer Protocol* (*HTTP*) protokoll által átvitt egyetlen információtípus a szöveg volt. Amikor képessé tették képek, hangok és egyéb adattípusok átvitelére, megjelenítést meghatározó leírókódok kerültek a HTML-be, hogy segítsék az újfajta adattípusok formázását. Néhány évnyi meglepően gyors növekedés után az eredetileg az egyéni elemek kódolására használt HTML kezdett túlméretezetté válni. Nyilvánvalóvá vált, hogy pusztán a leírókódot használni a megjelenítés szabályozására nem elég hatékony módszer arra, hogy meg lehessen határozni, hogyan nézzenek ki az egyes webhelyeken lévő szövegek és képek, ezért a webes közösség kifejlesztette a rangsorolt stíluslapokat (Cascading Style Sheets, CSS), mint a megjelenítés kezelésének egy jobb módszerét.

## Mi a különbség az XHTML és a HTML között?

Az XHTML igazából HTML – ez a W3C által ajánlott legújabb HTML-szabvány. Az XHTML-t éppen azért választottam a könyvben használt kódok alapjául, mert ez a legújabb ajánlott HTML-változat. Amikor az XHTML-t tanuljuk, a HTML nyelvet tanuljuk. Egyfajta „egyet fizet, kettőt kap” kedvezmény ez. Van persze néhány alapvető eltérés az XHTML és a HTML írása között, ezekre majd a könyv megfelelő részein rá fogok mutatni.

Az XHTML több, mint a HTML, mivel bővíthető. Az XHTML az *Extensible Markup Language* (*XML, bővíthető leírónyelv*) nyelvtani szabályait követi. A bővíthető leírónyelvek olyan modulokkal bővíthetők, amelyek képesek olyan feladatok végrehajtására, amilyen például a matematikai számítások elvégzése vagy a képek megrajzolása. Az XHTML-ben írt weblapok könnyedén együtt tudnak működni az XML nyelvvel.

## Mi a CSS?

A CSS egy rövidítés, az angol Cascading Style Sheets (rangsorolt stíluslapok) kifejezés rövidítése, ami nem más, mint egy másik programozási előírás. A CSS *stílus* néven ismert szabályok segítségével határozza meg a vizuális megjelenítést. A stílusok többféleképpen egyesülnek a weblapok tartalmával. Ebben a könyvben azokkal a stílusokkal foglalko-

zunk majd, amelyeket magába a weblapba ágyazunk be, valamint azokkal, amelyeket a weblaphoz csatolunk, illetve amelyeket a program beolvas a weblapba. Meg fogjuk tanulni, hogyan kell stílusokat írni, illetve beolvastatni, csatolni és beágyazni az általunk készített weblapokba.

A HTML-ben a stílusok beleírhatók a HTML lap szövegébe is (*szövegközi stílus*).

A CSS más módon is beépíthető a weblapokba. Néha nincs ráhatásunk ezekre a szabályokra. A böngészők lehetővé teszik a felhasználóknak, hogy saját kedvükre létrehozzanak bizonyos CSS stílusokat (más néven felhasználói stílusokat). A felhasználói beállítások felülbírálhatják az általunk írt stílusokat. Sőt mi több, minden böngészőben vannak beépített stílusok is. A beépített stílusokat általában felül lehet bírálni a CSS stílusokkal. A böngésző beépített stílusainak neve: *alapértelmezett megjelenítési stílusok*. A megtanulandók része az is, hogy mit várhatunk a böngészőtől alapértelmezésben. Erre azért van szükség, hogy tudjuk, milyen új CSS stílusokat kell létrehoznunk az alapértelmezett értékek felülbírálatához.

# Ismerkedés az XHTML nyelvtanával

Az XHTML-nyelvtan építőkockái a *címkék* (angolul *tag*), amelyeket az *elemek* kódolására lehet használni. A címke olyan kód, amely megadja az elem nevét is. Például a bekezdések formázására a `p` címkét használjuk, amelyet ezért „bekezdéscímke” vagy „p címke” néven emlegetnek. Amikor a szöveg `p` címkével van kódolva, az egy utasítás a böngészőnek, hogy az adott elemet bekezdésként jelenítse meg.

Az XHTML elemeinek, például a bekezdéseknek lehetnek jellemzői (attribútumai), és azokhoz tartozhatnak hozzárendelt értékek. Mielőtt azonban megismerkednénk a jellemzőkkel és az értékekkel, merüljünk egy kicsit mélyebbre a címkék világában.

## Kezdő- és zárócímkék

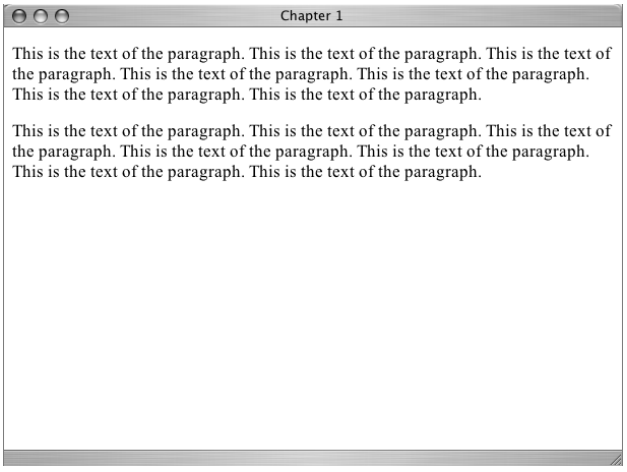
A kezdőcímkéket és zárócímkéket elemek megadására használják. Íme egy bekezdéselem a címkével együtt:

```
<p>Ez a bekezdés szövege.</p>
```

A bekezdést egy `p` címke nyitja. A címkék csúcsos zárójelek közt (< és >) vannak. A `<p>` leírókód azt tudatja a böngészővel, hogy itt kezdődik a bekezdés.

A bekezdés végét a záró `</p>` címke jelzi. Figyeljük meg, hogy a zárócímke ugyanaz, mint a kezdőcímke, csak kiegészül egy törtvonallal (`/`), ami a címke elejére kerül. A címkék olyanok, mint egy ki- és bekapcsoló gomb: itt bekapcsolják a bekezdést,

amott pedig ki. Ha beírunk még egy pár mondatot a bekezdésbe, majd felveszünk egy második, az elsővel azonos bekezdést is, ezek nagyjából úgy jelennek meg a böngészőben, ahogy az az 1.1. ábrán látható.




**1.1. ábra**
*Két bekezdéselem*

Figyeljük meg, hogy a böngésző egy üres sort hagy a két bekezdés között, és hogy nincs szövegbehúzás. Ez az ábra az *alapértelmezett bekezdést* mutatja. Az alapértelmezett megjelenítést az adja, hogy a böngésző beépített értelmezőrendszere szerint az elemnek hogyan kellene megjelennie. Az egyik mód arra, hogy megváltoztassuk, hogyan értelmezzen a böngésző egy elemet alapértelmezés szerint, ha további utasításokat veszünk fel *jellemzők* és *értékek* formájában, amelyekkel finomíthatjuk az elem megjelenítését.

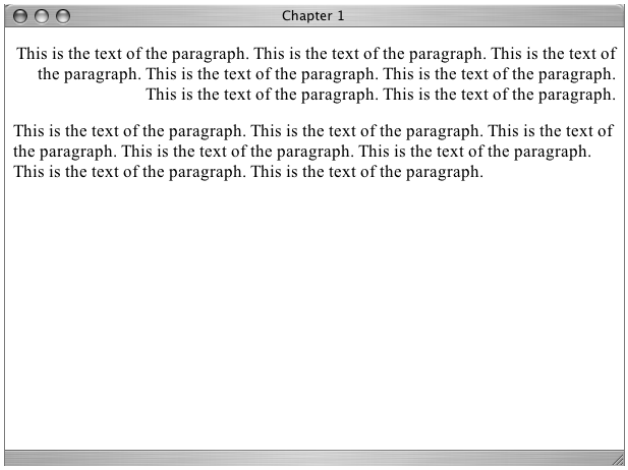
A jellemző az elemre vonatkozó információ. A bekezdést meghatározó jellemzőkre jó példa az igazítás. A bekezdések szövege lehet igazítva balra, jobbra, középre, vagy lehet sorkizárt. Amint az 1.1. ábrán látható, a böngésző alapértelmezett szövegigazítási módja a balra igazítás. Az XHTML-ben az igazítás típusát a választott érték adja meg. A pontos jellemző az `align` (igazítás), ennek értékei lehetnek a `left` (balra), a `right` (jobbra), a `center` (középre) és a `justify` (sorkizárt).

A jellemzőt a kezdőcímke részeként kell megadni. A jellemző nevét egy egyenlőségjel (`=`) követi, majd az érték idézőjelben (`"`).

Íme egy kódolt bekezdéselem jellemzőnévvel és -értékkel:

```
<p align="right">This is the text of the paragraph.</p>
```

Ha az 1.1. ábra első bekezdéseleméhez hozzáadjuk az `align="right"` jellemzőt és értéket, láthatjuk, hogy az igazítás olyanná változik, mint ami az 1.2. ábrán látható.




**1.2. ábra**
*Az első bekezdést az align="right" jellemzővel formáztuk*

Ebből a példából két fontos dolgot kell megjegyezni. Az első, hogy a címke és a jellemzőnév (`align`) között szóköz van. A jellemzőt egy egyenlőségjel követi, majd a jellemző értéke (`"right"`) következik, idézőjelben, de szóközök nélkül. Az XHTML nyelvben a jellemző–érték párok mindig ezt a formát követik. Figyeljük meg a zárócímkét is. Ez a bekezdés és a bekezdés jellemzői által keltett hatás lezárását végzi. Ehhez csak egy törtvonalat (`/`) és utána újra a `p` címkét kell használnunk. A bekezdés végén annak minden jellemzője és a jellemzők minden értéke érvényét veszti.

Az XHTML és a HTML közötti egyik különbség az, hogy az XHTML-ben mindig kötelező zárócímkét használni, míg a HTML-ben nem.

## Üres elemek

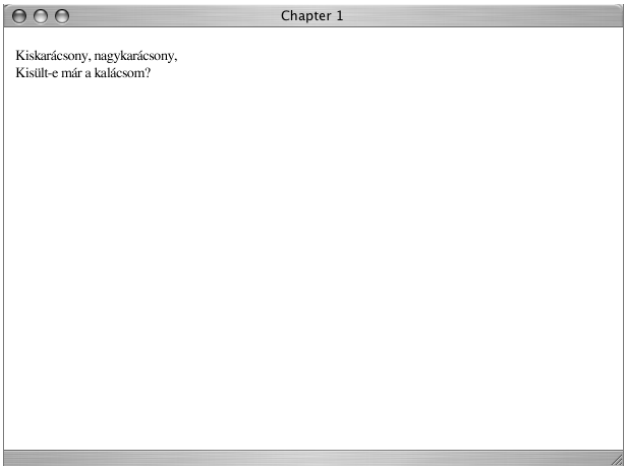
Mielőtt az üres elemeket ismertetném, hadd határozzam meg gyorsan, mik azok a nem üres elemek. Azokat az elemeket tekintik nem üres elemeknek, amelyek tartalmaznak valamilyen szöveget, mint például az előző példákban lévő bekezdések.

Néha olyan elemet is elhelyezünk a weblapokon, amely nem tartalmaz szöveget. Az ilyen elemek nem igényelnek zárócímkét – ezeket nevezik *üres elemeknek*. Ilyen elem például a kép vagy a sortörés. Mielőtt azonban kitárgyalnánk az XHTML-nek az üres elemekkel szemben támasztott követelményeit, íme egy példa HTML nyelven:

```
<p>Kiskarácsony, nagykarácsony,<br>
Kisült-e már a kalácsom?</p>
```

A `<br>` HTML címkét a bekezdés formázása során a sortörés beszúrására használják. A sortöréshez nem kell kezdő- és zárócímke, egyszerűen csak *jelen kell lennie*. A sortörés hatására új sor kezdődik, üres sor beszúrása nélkül, eltérően attól, amit alapértelmezés szerint akkor kapunk, ha a második sort új bekezdéselemben helyezzük el.

Ha az első sor végén sortöréscímkét helyezünk el, a bekezdés az 1.3. ábrán látható módon jelenik meg a böngészőkben.




**1.3. ábra**
*Sortörés*

Az XHTML-ben azonban az XML (Extensible Markup Language, *bővíthető leírónyelv*) nyelvtanát kell használni a HTML kód megírására. Az XML egyik követelménye, hogy minden elemet le kell zárni, akár üres, akár nem. Most kérdezhetné valaki, hogy hogyan lehet egy elemet „lezárni", ha nincs is hozzá zárócímke. A megoldás az XHTML esetében az, hogy magához az üres címkéhez adjuk hozzá a zárást jelképező törtvonalat. Az XHTML kódban az üres címke az alábbi példában látható módon néz ki.

```
<p>Kiskarácsony, nagykarácsony,<br />
Kisült-e már a kalácsom?</p>
```

Jegyezzük meg, hogy itt szóköz van a `br` címke és a törtvonal (`/`) között. A jellemzőket és értékeket tartalmazó üres címkéket is le lehet zárni ily módon.

A záró törtvonal előtti szóköz használata az XHTML nyelvben *nem kötelező*, azaz a `<br/>` kód is helyesnek számít. Ha viszont használunk szóközt, a régebbi böngészők is képesek lesznek helyesen megjeleníteni a dokumentumot, ezért én használni fogok szóközt a példákban. (Régebbi böngészőknek most azokat tekintem, amelyeknek a verziószáma 5-nél kisebb, például a Netscape 4.7.)

Egy másik üres elem a képekhez használt `img` címke. Vizsgáljuk meg az alábbi példát:

```
<img src="fenykep.gif" />
```

Ez az üres elem egy képet helyez el a weblapon. A kép forrását az `img` címke jellemzőjeként adjuk meg. A címke végén lévő szóköz és törtvonal az üres elemnek az XHTML nyelvben kötelező lezárását jelenti.

Az XHTML-ben nincs sok üres elem. Ilyenek még – a fentieken kívül – a vízszintes vonal kódja, a `link` elem és a `meta` elemek. Legtöbbször kezdő- és zárócímkét is használnunk kell a szöveg kódolásához. Még amikor HTML kódot írunk is, ahol a zárócímkék használata nem minden esetben kötelező, a gyakorlatban nem árt, ha használjuk a zárócímkéket is, ahol csak lehetséges.

### XHTML: különleges követelmények

Mint korábban már említettem, az XHTML nyelv az XML nyelv nyelvtani szabályait követi. Azon a követelményen kívül, hogy minden elemet le kell zárni, van még néhány olyan előírás az XHTML írására vonatkozóan, ami eltér a HTML nyelv követelményeitől:

- Különleges DOCTYPE meghatározásokra van szükség, amelyeket a 3. fejezetből ismerhetünk meg.
- Minden elemet, jellemzőt és értéket kisbetűvel kell beírni.
- Minden értéket idézőjelben kell megadni. Az idézőjel lehet egyes (angol) vagy kettős (normál), de egységesen ugyanazt a típust kell használni végig. Én ebben a könyvben következetesen kettős idézőjeleket (`"`) használok.
- Minden jellemzőnek egyértelműen kell megadni az értékét.

Bár ezeket a szabályokat HTML kód írásakor nem kötelező betartani, a HTML-ben is kiválóan használhatók. Az egyetlen olyan XHTML-szabály, ami a HTML-ben érvénytelen eredményt ad, az üres elemek törtvonallal történő lezárása. Amennyiben úgy döntenénk, hogy az XHTML helyett HTML-t fogunk használni, csak ezt a kis módosítást kell elvégeznünk a kódon, és máris HTML lapot írtunk.

A könyvből megtanulhatjuk a legfontosabb XHTML kódokat és jellemzőket, de amikor önállóan kezdünk dolgozni, hasznosnak találjuk majd, ha lesz egy teljes körű ismertetőnk a nyelvről. Ezt megtalálhatjuk például a *HTML Complete* (Sybex, 2003) című könyvben, amely hasznos információkat tartalmaz, többek közt egy teljes CSS-parancsismertetőt is.

## Ismerkedés a CSS nyelvtanával

A *rangsorolt stíluslapokat* (*Cascading Style Sheets*, *CSS*) arra használják, hogy megjelenítési tulajdonságokkal lássák el a leírókódon belüli elemeket. A CSS segítségével beállíthatjuk a színeket, a betűtípusokat, a háttereket, a szegélyeket, a margókat, sőt még az elemek helyét is a weblapon.

A CSS stíluslap lehet közvetlenül az XHTML dokumentumban, vagy önálló fájlban kapcsolódhat hozzá. A 2. fejezetben mindkét változatot megismerhetjük, de a gyakoribb eset az, amikor a CSS-t hozzákapcsolják az XHTML laphoz. Az egyik dokumentum tartal-

mazza az XHTML lapot, amelyet megtanulunk majd megtervezni úgy, hogy tiszta, logikus felépítése legyen címsorokkal, bekezdésekkel, hivatkozásokkal és képekkel, amelyek az ötleteink megjelenítéséhez szükségesek, és egy másik fájl tartalmazza a stíluslapot, amely azt adja meg, hogy milyen színűek legyenek a képernyőn megjelenő betűk, mi legyen kihangsúlyozva stb. Így egyszerűen módosíthatjuk a weblapok megjelenését, elég a stíluslapot megváltoztatni anélkül, hogy a tartalomhoz egyáltalán hozzányúlnánk.

Az, hogy egy webhely teljes megjelenését megváltoztathatjuk pusztán a stíluslapot módosítva, hatalmas rugalmasságot jelent. Rengeteg időt takaríthatunk meg a karbantartás és a frissítés terén, mivel a stílusok egy olyan fájlban vannak, amely elkülönül a tartalomtól. Egy stíluslaphoz bármennyi weblapot hozzákapcsolhatunk, így percek alatt végrehajthatjuk a megjelenésbeni módosításokat az összes érintett weblapon. Ha a böngésző letöltötte a stíluslapot, egy különleges mappába menti, amelynek *gyorsítótár* (angol nevén *cache*, kiejtve: *kes*) a neve. Amikor a következő alkalommal is olyan weblapot töltünk le, amelyik ugyanazt a stíluslapot használja, nem kell várni, amíg a stíluslap letöltődik, mert a böngésző már a gyorsítótárában tárolja azt, így minden olyan weblap, amelyik ugyanazt a stíluslapot használja, nagyon gyorsan letöltődik, időt és sávszélességet takarítva meg ezáltal.

**Ha ötleteket keresünk, hogy miként lehet ugyanazt a tartalmat különféleképpen formázni CSS stíluslapokkal, keressük fel a `www.csszengarden.com` webhelyet.**

A stílusok és stíluslapok kinézete nagymértékben eltér az XHTML lapokétól, és eltérőek a stílusok írásához használandó nyelvtani szabályok is.

## Kijelölők és meghatározások

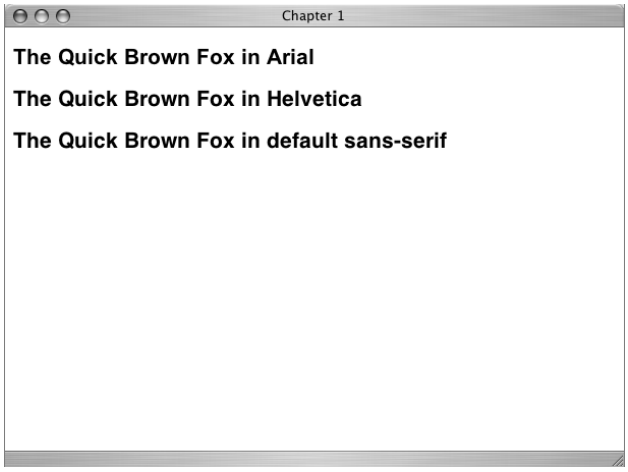
A stílusokat *kijelölők* (selector) és *meghatározások* (declaration) segítségével lehet megadni. A kijelölők, mint a nevük is mutatja, kijelölnek valamit – azaz kijelölik, hogy a stílus az XHTML lap mely elemeire vonatkozik. A legegyszerűbb kijelölő az elemkijelölő: a `p` kijelölő kijelöli például a weblapon lévő összes bekezdéselemet.

Minden kijelölőhöz írnunk kell egy meghatározáskészletet, ami azt adja meg, hogy a kijelölt elem hogyan jelenjen meg a képernyőn. A kijelölő és a meghatározások együtt alkotják a stílust. Íme egy meghatározáskészlet a `p` kijelölőhöz:

```
p {
font-family: Arial, Helvetica, sans-serif;
font-size: small;
color: blue;
}
```

Vizsgáljuk meg a fenti példát részletenként. Azt már tudjuk, hogy a `p` a kijelölő. Minden, ami a két kapcsos zárójel (`{}`) között van, a meghatározásblokk része, ami esetünkben három különböző meghatározást tartalmaz.

A meghatározások összetevői: egy *tulajdonság* (property), amelyet egy kettőspont követ, majd egy szóköz, és legvégül áll az *érték* (value). Az érték után egy pontosvessző kerül. Amint az ezen az egyszerű példán is látható, a CSS stíluslapok tulajdonságai hasonlók az XHTML lapok jellemzőihez. Mindkettő a formázandó elem egy jellemzőjét azonosítja. A példában az első megadott tulajdonság a bekezdések szövegéhez használandó betűcsalád. Első választásként az Arial betűtípust adtam meg, ha az megtalálható a számítógépen. Ha nem található meg, akkor megteszi a Helvetica, ha az sincs a számítógépen, akkor pedig a rendszer alapértelmezettként használt talp nélküli (sans-serif) betűtípusával jelenik meg a bekezdések szövege. Az 1.4. ábrán mindhárom betűtípust megtekinthetjük.

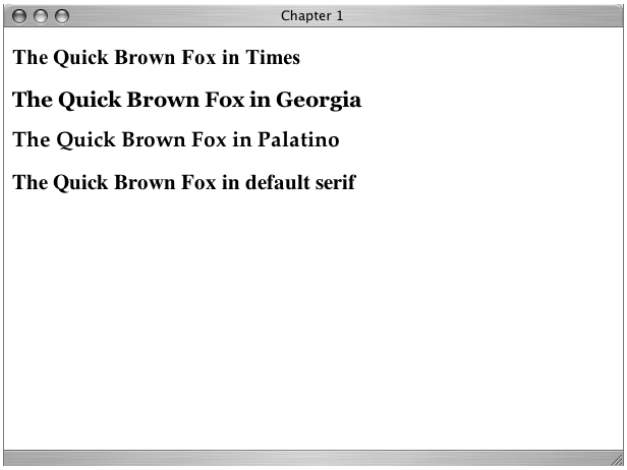



**1.4. ábra**
*Talp nélküli betűcsaládok*

A *betűcsalád* a nyomdászok által használt, kicsit szakmai ízű kifejezés arra, amit mi, hétköznapi emberek általában *betűtípus* vagy *betűkészlet* néven szoktunk emlegetni. Ha szigorúan vesszük, akkor egy betűtípuson belül minden méret- vagy vastagságváltozat külön betűkészlet, és ezeknek a változatoknak az összessége a betűcsalád.

A gyakorlatban érdemes több betűcsaládot is megadni a meghatározásban, mert nem ugyanazzal a betűtípuskészlettel van ellátva minden számítógéprendszer. Amint a fenti példában is látható, a betűtípusokat elsőbbségi sorrendben kell megadni (azaz legelsőként azt, amelyikkel leginkább szeretnénk megjeleníteni a szöveget).

Ha nem adunk meg betűcsaládot, a böngészők általában a Times betűtípust használják alapértelmezettként. A talpas betűtípusokra az 1.5. ábrán láthatunk néhány példát.




**1.5. ábra**
*Elterjedt talpas betűcsaládok*

A betűkészletekkel és betűcsaládokkal bővebben a 4. fejezet foglalkozik. A tipográfiáról és a betűkészletekről sok könyv készült már, ajánlom például a Robin Williams által írt *The Non-Designer's Design Book* című művet. A `http://counterspace.motivo.com/` címen a CounterSpace cég jó alapismereteket nyújt a témához.

Az előző stílus második meghatározása a `font-size: small`. A betűméretek különféle beállításairól a 4. és 5. fejezetben olvashatunk, de biztos vagyok benne, hogy mindenki kitalálta, hogy ez a meghatározás az összes `p` elem (azaz bekezdés) betűméretét kicsire (`small`) állítja. Az utolsó meghatározás a betűk színét állítja kékre (`blue`).

Ha a felhasználó nem módosítja a böngésző alapbeállításait, az alapértelmezett betűméret a legtöbb böngészőben a közepes (`medium`).

Ennek a stílusnak a hatására az adott weblapon lévő összes bekezdés a normál méretűnél kicsit kisebb, kék színű, az Arial betűcsaládba tartozó betűvel jelenik meg a képernyőn.

A könyvben lévő példákban minden meghatározást külön sorba írtam, a záró kapcsos zárójelet pedig szintén külön sorba. Így a stílus könnyebben olvasható, de nem szükséges pontosan így formázni a stílust a megírásakor, lehet írni akár így is:

```
p {font-family: Arial, Helvetica, sans-serif; font-size: small;
   ➥ color: blue;}
```

Ha azonban több meghatározást is írunk egy sorba, okvetlenül tegyünk szóközt a pontosvesszők után!

Egyes stílusokat megírhatunk egyfajta „gyorsírással” is. A betűtípus minden tulajdonságát megadhatjuk például ilyen gyorsírásos módszerrel, a `font-style` (betűstílus), `font-variant` (betűváltozat), `font-weight` (betűvastagság), `font-size` (betűméret), `line-height` (vonalmagasság) és `font-family` (betűcsalád) tulajdonságot is. Ez lehetőséget ad arra, hogy két betűtípus-meghatározást egyesítsünk egyetlen gyorsírásos meghatározássá az alábbi módon:

```
p {
font: small Helvetica, Arial, sans-serif;
color: blue;
}
```

A szín nem számít a betű tulajdonságának. A szín tulajdonság az előtérszínt jelenti, azaz azt, hogy a weblap előterében lévő szöveg a megadott színű lesz. Megkülönböztetik viszont a színt a háttérszíntől, ami – mint már biztosan mindenki rájött – a teljes elem hátterének színét állítja be.

## Pontos kijelölők

Gyakran egyértelműbben kell megadni az XHTML-ben egy elem stílusát, mint ami az első példában látható. A CSS lehetővé teszi pontosabb kijelölők használatát, mint amit az általános elemkijelölők le tudnak írni. Mivel a kijelölők adják meg, hogy a dokumentum mely elemére lesznek hatással a stílusok, az olyan elemkijelölők, mint a fenti példában szereplő `p` kijelölő a weblap összes azonos elemére (a példában az összes bekezdésre) hatással lesznek. Vannak azonban olyan esetek, amikor azt szeretnénk, hogy adott (a CSS szóhasználata szerint *specific*, azaz pontosan megadott) bekezdésekre más szabályok legyenek érvényesek, mint a bekezdésekre általában. Az olyan kijelölők két fajtája, amelyekkel ez megoldható, az *azonosítókijelölő* (ID selector), illetve az *osztálykijelölő* (class selector). Ez a két kijelölő lehetővé teszi, hogy olyan stílusokat írjunk, amelyek csak bizonyos környezetben előforduló elemekre lesznek érvényesek. Például ahelyett, hogy a weblapon lévő összes bekezdést egyetlen stílussal formáznánk, formázhatunk csak egy bizonyos osztályba tartozó bekezdéseket vagy egy bizonyos azonosítójú bekezdést.

**A könyvben szereplő kijelölőtípusok a következők: elemkijelölők, osztálykijelölők, azonosítókijelölők, környezetfüggő kijelölők, álosztály-kijelölők és csoportkijelölők.**

### *Azonosítókijelölők*

Egy azonosítót egy XHTML lapon csak egyszer lehet használni. Általában azoknak a tartalomrészeknek az azonosítására szolgálnak, amelyeket egy szerkezeti egységként (például fejléc, lábléc, tartalomblokk vagy menü) szeretnénk valamilyen stílussal formázni. Ezzel a fogalommal a könyv csaknem minden fejezetében dolgozni fogunk, de most egyszerűen csak azt fogjuk megvizsgálni, hogyan néznek ki az azonosítókijelölők a stíluslapon.

Az azonosítókijelölőket ez a jel előzi meg: `#`. Ennek a jelnek az igazi neve „oktotorp”, de általában kettőskeresztnek nevezik, a könyvben mi is annak fogjuk. A stíluslapon egy azonosítóstílus így néz ki:

```
#lablec {
font-size: x-small;
}
```

**Az XHTML nyelvben az `id` jellemzőt csupa kisbetűvel kell írni.**

Ez a stílus a weblap `lablec` (lábléc) névvel azonosított részén (vagy szakaszában) mindent nagyon kis méretűre állít. Figyeljük meg, hogy a kettőskereszt és az azonosító neve között nincs szóköz.

Most tegyük fel, hogy nem akarunk a láblécben mindent ennyire kis méretűre állítani, csak a bekezdést. Ezt egy olyan *környezetfüggő kijelölő* segítségével valósíthatjuk meg, ami csak a weblap `lablec` névvel azonosított szakaszában lévő bekezdésekre lesz hatással:

```
#lablec p {
font-size: x-small;
}
```

Figyeljük meg, hogy a `#lablec` és a `p` között van szóköz. Ez a betűméretérték a weblapon lévő többi bekezdésre nem lesz hatással, csak azokra, amelyek a láblécben (pontosabban a `lablec` bekezdésben) helyezkednek el. Az azonosítókijelölő ilyen használata, amikor az elemkijelölő követi, nagyon pontosan csak azt a bekezdést jelöli ki, amelyik a weblap adott részén helyezkedik el.

A `#lablec p` kijelölő környezetfüggő (vagy *származtatott*, descendant) kijelölő. Az XHTML lapba névvel ellátott azonosítóval lehet környezetfüggő kijelölőket beilleszteni, ezek az azonosítók aztán lehetővé teszik a finoman megrajzolt CSS kijelölők használatát. A következő fejezetekben származtatott kijelölőkkel hozunk majd létre számos CSS stílust.

Az azonosítókijelölők nevét mi találhatjuk ki, nem kell kapcsolatban lenniük semmilyen XHTML címkével. Érdemes rövid és beszédes nevet adni, amely jellemzi az azonosítóval jelölt szakasz szerepét a teljes tartalomban. A `#lablec` azonosítókijelölő jó példa minderre. Ha készítünk egy stíluslapot, a `#lablec` azonosítókijelölőről akkor is tudni fogjuk, mire vonatkozik, ha csak hónapokkal később térünk vissza a stíluslaphoz, mert akkor kell azt megtekintenünk és módosítanunk.

### *Osztálykijelölők*

Az osztálykijelölőket annyi példányban használhatjuk egy XHTML lapon, ahányban csak akarjuk. Az osztálykijelölők előtt egy pont (`.`) áll. Akárcsak az azonosító, az osztály (`class`) nevét is mi magunk adhatjuk meg. Ha olyan stílust szeretnénk létrehozni, amelyik kihangsúlyoz egyes kifejezéseket a weblapon, hozzunk létre egy osztályt, és adjuk neki a `kifejezes` nevet.

```
.kifejezes {
background-color: silver;
}
```

A stílus ezüstszürke hátteret hoz létre azon szavak és kifejezések mögött, amelyeket úgy azonosítottunk, hogy azok a `kifejezes` osztály részei. Figyeljük meg, hogy a felvezető pont és az osztálynév között nincs szóköz.

A CSS népszerűségének egyik oka, hogy egy weblap teljes kinézete szinte pillanatok alatt megváltoztatható a segítségével. Érdemes olyan osztálynevet választani, amelyik az osztály célját fejezi ki, nem pedig a pillanatnyi formázási feladatot, például színnevet, ami később esetleg megváltozhat. A `.kifejezes` osztálynév ezért jobb választás, mint az `.ezust`, mivel a `kifejezes` név akkor sem válik „elavultá", ha a színt netán megváltoztatnánk.

A weblap egy adott részének formázásához használhatjuk a kijelölők, azonosítók és osztályok kombinációit is. Az alábbi kijelölő, illetve a vele meghatározott stílus csak azokra a `kifejezes` osztályba tartozó bekezdésekre lesz hatással, amelyek a weblap `lablec` azonosítójú szakaszában helyezkednek el.

```
#lablec p.kifejezes {
background-color: gray;
}
```

Figyeljük meg, hogy amikor stílusmeghatározást írunk egy XHTML lap egy olyan eleméhez – például a `p` elemhez –, amely egy adott osztályhoz tartozik, az elem és az osztálynévhez tartozó pont között nincs szóköz: `p.kifejezes`.

A SelectORacle egy ingyenes eszköz, amely az összetett CSS kijelölőket hétköznapi angolra fordítja le, hogy (némi angoltudás birtokában) könnyebben megérthessük, milyen CSS stílusokat választottunk ki. Az eszköz a `http://gallery.theopalgroup.com/selectoracle/` webhelyen érhető el.

## Kijelölők csoportosítása

Időnként ugyanazt a stílust a weblap több elemére is alkalmazni szeretnénk. Előfordulhat, hogy a weblapon lévő összes bekezdést, listát és idézetblokkot ugyanolyan betűmérettel szeretnénk megjeleníttetni. Ennek eléréséhez soroljuk fel a kijelölőket egymástól vesszővel elválasztva, majd adjuk meg a betűméretet:

```
p, li, blockquote {
font-size: medium;
}
```

Ez a stílus az összes bekezdés, lista és idézetblokk betűméretét közepesre (`medium`) állítja. Figyeljük meg, hogy a kijelölők vesszőkkel tagolt listájának elemei között szóköz van.

A fenti példában a vessző állítja be a stílust a lista minden elemére. Hasonló megoldást vessző nélkül is létrehozhatunk:

```
p blockquote {
font-size: medium;
}
```

Ha nem használunk vesszőt, az eredmény nagyon hasonlít a korábban látott, vessző nélküli `#lablec p` stílushoz, nemde? A `p blockquote` kijelölő vessző nélkül nem minden idézetblokkra lesz hatással, csak azokra, amelyek valamelyik bekezdés részei.

A vessző (vagy a vessző hiánya) tehát fontos megkülönböztető elem, ez tesz különbséget a csoportosított kijelölők és a származtatott kijelölők között. A csoportosított kijelölőkben van vessző, a származtatott kijelölőkben nincs.

**Ha odafigyelünk az olyan részletekre, hogy van-e vessző, pontosvessző, szóköz vagy zárójel, és ügyelünk a pontos beírásra, sokkal nagyobb sikerélményünk lesz a könyvben szereplő XHTML- és CSS-példák tanulmányozása során. Ha végrehajtunk egy gyakorlatot, és az nem úgy működik, ahogy kellene, nézzük meg, hogy nem gépeltünk-e el valamit. A nyelvtant pontosan be kell tartani!**

## Idézőjelek

Az utolsó eltérés az XHTML és a CSS írására vonatkozó nyelvtani szabályokban, amelyeket a 2. fejezet megkezdése előtt meg kell ismernünk, az idézőjelek használata közötti különbség.

Emlékezzünk vissza, hogy az XHTML-ben a címkék összes jellemzőértékét idézőjelek közt kell megadni. A CSS stílusmeghatározásokban azonban nincsenek idézőjelben a tulajdonságok értékei. A stíluslapokon többnyire nem találkozunk idézőjelekkel, kivéve azt az esetet, amikor több szóból álló betűtípusneveket adunk meg, és szóközök is vannak a betűtípus nevében.

Korábban láttuk az alábbi stílusmeghatározást, amelyben nem volt idézőjel:

```
p {
font-family: Helvetica, Arial, sans-serif;
```

```
font-size: small;
color: blue;
}
```

A példában szereplő betűtípusok nevei egy szóból álltak. A `sans-serif` ugyan kötőjeles, de szóköz nincs benne, így nyelvtani szempontból egy szónak számít. Ha azonban olyan betűtípusneveket is fel szeretnénk venni a listára, mint a Times New Roman, ami több szóból áll, és szóköz is van a szavak közt, akkor a nevet idézőjelben kell megadni:

```
p {
font-family: "Times New Roman", Times, Georgia, serif;
}
```

Figyeljük meg, hogy a Times New Roman és a Times betűtípusnevet elválasztó vessző a záró idézőjel *után* áll. Azt is jegyezzük meg, hogy míg a tényleges betűcsaládnevek – például a Times – nagybetűvel kezdődnek, addig az általános betűtípusnevek – például a serif – nem.

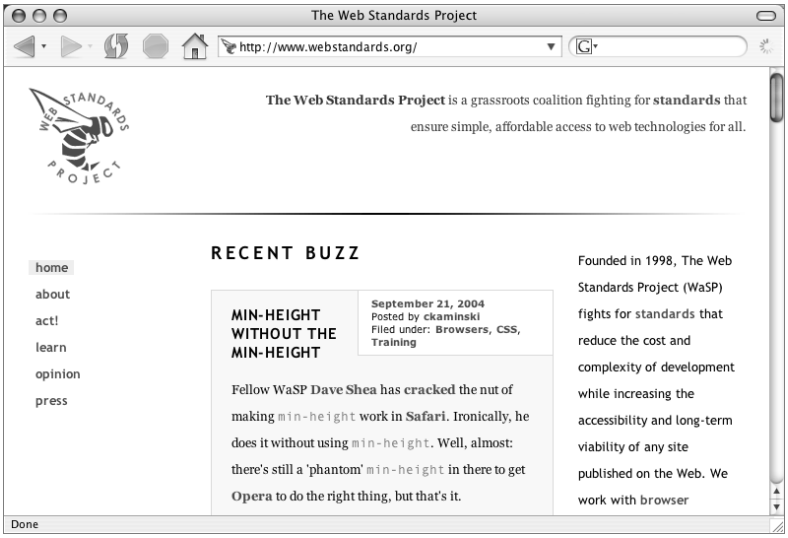
# Gyakorlati példa

Az egész könyvben hangsúlyozom, hogy a W3C által mind a weblapkészítők, mind a böngészőgyártók részére ajánlott szabványi előírások betartása megkönnyíti és felgyorsítja az XHTML és a CSS lapok megírását, és egyetemesebbé teszi az így elkészített lapokat.

Van egy olyan önszerveződő csoport is – a nevük The Web Standards Project (WaSP, Webes Szabványok Munkacsoport), amely erejét megfeszítve dolgozik a webes szabványok megvalósításáért, őket a `www.webstandards.org` webhelyen lehet elérni (1.6. ábra).

A Web Standards Project webhelyén olyan lehetőségeket találhatunk, amelyek segítségével mi is tehetünk a webes szabványok betartásáért, és olyan információkat, amelyek segítenek megtanulni ezeknek a szabványoknak az előírásait.

Ne feledjük, hogy a *Gyakorlati példa* című alfejezetekben megadott webhelyek – igazából a könyvben említett összes webhely – a másolást illetően szerzői jogi törvények védelme alá esnek. A szerzői jogi törvények védik a webhelyen található képeket és szövegeket is. Én arra biztatom az olvasókat, hogy nézzenek körül ezeken a webhelyeken és okuljanak, de ne másoljanak le onnan semmit. Pár kivétel persze akad – például azok a webhelyek, amelyek azzal foglalkoznak, miként lehet a CSS segítségével kialakítani a weblapok megjelenését, nagyon egyértelműen tudatják, hogy saját használatra lemásolhatjuk-e az ott található anyagokat –, de általában jobb azt feltételezni, hogy nem engedélyezik, hogy „kölcsönvegyük” a webhelyeken talált anyagokat.




**1.6. ábra**

*A Web Standards Project honlapjának Recent Buzz (A legfrissebb hírek) című része tartalmazza a friss híreket – ezt érdemes rendszeresen átolvasni*

# Összefoglalás

Az XHTML segítségével új weblapokat hozhatunk létre. Elvégezhetjük vele a weblapon elhelyezett információkat valamiféle logikai rendbe szervező címsorok, bekezdések, táblázatok, képek és listák formázását, hogy az üzenetünk szavak, képek és egyéb adatok formájában, érthetően jelenjen meg a böngészőkben. Az XHTML lapokat a böngészők akkor is megjelenítik, ha nem tartozik hozzájuk CSS. CSS nélkül a weblap esetleg lapos lesz és színtelen, de minden információ és tartalmi elem megjelenik rajta, logikailag jól szervezett elrendezésben.

A CSS a megjelenítést szolgálja, és a legtöbb weblaptervező már ezt használja a képernyős megjelenés formázására. A CSS határozza meg, hogy valami zöld legyen-e vagy kék, a bal oldalra kerüljön-e vagy a jobbra, nagy legyen-e vagy kicsi, látható-e vagy rejtett, legyenek-e előtte felsorolásjelek vagy ne. A stíluslapokat azonban valamilyen más dokumentumra, például egy XHTML lapra kell alkalmazni, hogy bármilyen hatásuk legyen.

A 2. fejezetben elkészítjük első XHTML dokumentumunkat és első stíluslapunkat. Miközben ezt végrehajtjuk, megtanuljuk, hova kell írni a stílusokat, és hogyan kell azokat az elemekhez kapcsolni. A 2. fejezetből megtudhatjuk azt is, miért szerepel (azaz mit jelent) a *Cascade* (*rangsor*) szó a Cascading Style Sheets (rangsorolt stíluslapok) elnevezésben.