# SZEGEDI TUDOMÁNYEGYETEM BÖLCSÉSZETTUDOMÁNYI KAR SZEGED

# HONLAPOK FORMÁZÁSI LEHETŐSÉGEINEK BŐVÍTÉSE A CSS NYELV SEGÍTSÉGÉVEL

Szakdolgozat

Készítette: Rimár Miklós Könyvtári informatikus szak

Témavezető: Sándor Ákos

# Tartalomjegyzék

1. Bevezetés	3
2. A CSS alapjai	∠
2.1 HTML és SGML	∠
2.2 A stíluslapok	5
2.3 A CSS története	6
2.4 Szintaxis	
2.5 Látszólagos elemek	
2.6 Stíluslapok csatolása	
2.7 Dokumentumfa és dobozmodell	
3. Margó-, szegély- és kitöltés-tulajdonságok	16
3.1 Margók	
3.2 Margók összeolvadása	18
3.3 Kitöltések	18
3.4 Szegélyek	19
3.5 Rövid szegélytulajdonságok	20
3.6 Körvonalak	22
4. A felhasználói felület	
4.1 Színek	
4.2 Háttértulajdonságok	
4.3 Rendszerszínek használata	
4.4 A böngészőablak színei	28
4.5 A kurzor	29
5. Betű- és szövegtulajdonságok	30
5.1 Betűk formázása	30
5.2 Szövegtulajdonságok	
6. Megjelenítés és pozícionálás	40
6.1 Rétegek és pozíciók	40
6.2 Láthatóság	44
6.3 Megjelenés	46
6.4 Körülfolyatás	46
6.5 Túlcsordulás	48
7. Táblázatok	49
7.1 Formázás	49
7.2 Táblázatszegélyek	51
7.2.1 Elkülönített szegélyek	51
7.2.2 Összevont szegélyek	52
8. Listák tulajdonságai	53
8.1 Jelölők	
8.2 Tartalom létrehozása, automatikus számozás	55
9. Médiafüggő stíluslapok	
10. A jövő	
Irodalomjegyzék	62

# 1. Bevezetés

Ez a dolgozat a honlapszerkesztés területén egyre szélesebb körben használt CSS nyelv 2.0 verzióját kívánja bemutatni. A nyelv segítségével stíluslapokat hozhatunk létre, amelyek alkalmazásával nagymértékben javítani tudjuk oldalaink megjelenését. A HTML viszonylag szűkös kereteihez képest számtalan formázási és igazítási lehetőség áll rendelkezésünkre, amelyek az igényesebb felhasználók elvárásait is kielégítik.

Sajnos a CSS által kínált lehetőségeket nem tudjuk maradéktalanul kihasználni, az egyes böngészők ugyanis nem minden formázási opciót valósítanak meg. Éppen ezért nem vettem fel a dolgozatba azt a néhány CSS tulajdonságot, amelyet az elterjedt böngészők egyike sem támogat. Bekerültek viszont azok, amelyeket a programok egy része megjelenít, mert jó esély van rá, hogy az egymással való vetélkedés következtében ezeket a többi böngésző is igyekszik elérhetővé tenni későbbi verzióiban.

A fentiekhez hasonló okok miatt nem esik szó a képernyőn megjelenő (görgethető) dokumentumoktól eltérő, tehát a nyomtatott, ún. lapozható (paged media) és a hanggal megjelenített (aural) dokumentumtípusokra jellemző sajátos formázási lehetőségekről. Ezeket a különböző böngészők meglehetősen rapszodikusan kezelik, és többnyire inkább figyelmen kívül hagyják.

A munka alapvetően a W3C eredeti, angol nyelvű CSS2 ajánlására épül. Bár léteznek különféle, részleges vagy teljességre törekvő magyar fordítások, ezeket kisebb-nagyobb pontatlanságaik miatt csak tájékoztató jelleggel tudtam figyelembe venni a dolgozat készítése során. A specifikáció mellett a stíluslapok alkalmazásával kapcsolatos források (elsősorban weboldalak), valamint tipográfiai szakirodalom segítségével igyekeztem a különböző tulajdonságok felhasználhatóságára vonatkozó eseteket összeállítani.

A dolgozatban szereplő példákat, valamint a felsorolt CSS tulajdonságokat a weben leggyakrabban használt böngészők: az *Internet Explorer*, a *Mozilla* és az *Opera* legújabb verzióinak segítségével teszteltem. A különböző statisztikák szerint az internetezők 90–95 százaléka ezeket a szoftvereket (illetve a Mozilláéval azonos motort tartalmazó *Netscape Navigator*t) alkalmazza a weboldalak megjelenítésére, tehát a fenti böngészőknél tapasztalt eredmények, megfigyelések relevánsnak tekinthetőek.

# 2. A CSS alapjai

#### 2.1 HTML és SGML

A weboldalak készítésére szolgáló HTML nyelv 1990-ben jelent meg. Készítői a CERN-ben dolgozó TIM BERNERS-LEE, aki a nyelvet a WWW alapjául szolgáló másik két ajánlássa együtt (HTTP, URI) kidolgozta, és DANIEL W. CONNOLLY voltak. Megjelenése nem volt előzmény nélküli: alapjául az 1986-ban ISO szabvánnyá vált SGML szolgált.

Az SGML-t (Standard Generalized Markup Language, szabványos általános jelölőnyelv) dokumentumok számítógépes tárolására és feldolgozására hozták létre. Kifejlesztéséhez az a jelenség vezetett, hogy az intézmények, nagyvállalatok nem tudták egységesen kezelni a különböző részlegektől, beszállítóktól kapott dokumentációkat, azok eltérő formátuma, belső szerkezete és sajátos elemei miatt. Ezért kidolgoztak egy technológiát, amely lehetővé teszi strukturált dokumentumok létrehozását, vagyis a szöveg egyes alkotóelemeit (pl. irodalmi művek esetében ilyenek a szerző, cím, fejezet, versszak, verssor stb.) meghatározhatjuk és elkülöníthetjük. Az elemek elhatárolása ún. címkékkel (angol 'tag') történik, ezek vezetik be és zárják le az egyes elemeket.

Nemcsak azt tudjuk szabályozni, hogy a dokumentum milyen részekből álljon, de az egyes elemek egymáshoz való viszonya, a szövegrészek hierarchiája is megadható. A megfelelő módon strukturált dokumentum azután gépi úton is feldolgozható és visszakereshető. A módszer használata sok területen elterjedt, a repülőgépgyártástól a lexikonkészítéséig számtalan területen alkalmazzák.

Egy SGML alkalmazásnál meg kell határoznunk a használandó dokumentumtípusokat, azok szerkezetét. Ez a DTD (dokumentumtípus-definíció), amely megadja, hogy az adott típusú dokumentum milyen elemeket tartalmazhat, azokhoz milyen attribútumok tartoznak, illetve leírja a felhasználható entitásokat. Az elemek az SGML szemlélete szerint egy faszerkezetben helyezkednek el, melynek gyökér eleme adja meg a dokumentum típusát.

Az SGML legelterjedtebb megvalósítása a HTML, amely nem más, mint egy hiperszövegek képernyőn való megjelenítésére kidolgozott dokumentumtípus-definíció. Az adatszerkezetet és hierarchiát ebből következően csak minimális mértékben határozza meg. Tartalmaz ugyan olyan elemeket (H1, P stb.), amelyek használhatóak lennének strukturált leíráshoz, de ezek rendeltetésszerű használata nem terjedt el.

Az SGML nem határozza meg, hogy milyen részekből álljon a szöveg, nem tartalmaz konkrét elemeket, utasításokat, csak egyfajta keretet, iránymutatást ad a dokumentumok strukturálását illetően. Ezzel szemben a HTML egy speciális, előre meghatározott és véges számú elemkészlettel dolgozik, vagyis egy feladatspecifikus SGML DTD-t valósít meg.

## 2.2 A stíluslapok

Az SGML szabvány nem rendelkezik a dokumentumok megjelenítéséről. Ez a látszólagos hiányosság éppen az SGML filozófiájának lényegéből: a tartalom és a forma különválasztásából ered. A dokumentumokat egyszerű ASCII szövegfájlként tárolják, amelyek mellé a mindenkori igényeknek és technikai lehetőségeknek megfelelően megjelenítési információkat társíthatunk. Ezek a stíluslapok, amelyek a dokumentumok külalakját szabályozzák. Egy stíluslapot több (akár ezres nagyságrendű) dokumentumhoz is hozzárendelhetünk, ezzel egységes megjelenést biztosítva a számukra. Egy dokumentumhoz pedig több stíluslap is csatolható, a különböző felhasználók vagy az egyes megjelenítő eszközök számára.

A HTML korlátai hamar megmutatkoztak. A nyelv nem tudja megfelelő mértékben feltárni a dokumentumok szerkezetét, hiszen elsősorban a képernyőn történő megjelenítés feladatára lett kidolgozva. Azonban a megjelenésbeli lehetőségei is eléggé korlátozottak, így a honlapkészítők esztétikai, tipográfiai törekvései csak szűk keretek között mozoghattak.

A HTML nyelv viszonylag kevés formázási lehetőséget nyújtott a weboldalak készítői számára. Ez főleg a nyomtatott kiadványok szerkesztésénél adódó lehetőségekkel összevetve volt szembetűnő. Ezért jelentek meg a web világában is a stíluslapok, amelyek kibővítették a honlapkészítők eszközeit, és módot adtak arra, hogy a korábbiaknál igényesebb weboldalak készüljenek, amelyeknél több lehetőség nyílik az egyes megjelenési tulajdonságok beállítására, módosítására.

Stíluslapok nemcsak a dokumentum szerzőjétől származhatnak, de a felhasználótól vagy egy alkalmazástól is. Ez azt jelenti, hogy a felhasználó is készíthet vagy megadhat stíluslapot, amit hozzárendelhet az általa használt dokumentumhoz. Egyes alkalmazások (böngészők, felolvasóprogramok stb.) szintén képesek lehetnek rá, hogy saját stílusdefinícióik alapján jelenítsék meg a dokumentumot. Ilyen esetben a szerzői stíluslapok bírnak a legnagyobb nyomatékkal, alapértelmezésben ezek kerülnek megjelenítésre. Őket követik a felhasználói szabályok, de pl. a CSS esetében ez a sorrend megfordítható a felhasználói stíluslapban elhelyezett !important utasítással. Ha a szerzői és a felhasználói stílus is tartalmaz !important parancsot ugyanarra az elemre, akkor a felhasználóé fog érvényesülni. Végül a böngésző stíluslapja következik, amely a legkisebb precedenciaértékkel bír.

A HTML nyelv a 4.0-s verziójától kezdve támogatja a stíluslapok használatát. Ezzel egyidejűleg a HTML bizonyos elemeinek attribútumai, sőt egyes elemek is elavulttá váltak. Ez azt jelenti, hogy a jelenlegi (4.01) változat még lehetővé teszi ugyan a használatukat, de a nyelv következő verziói már nem biztos, hogy értelmezni fogják azokat.

#### 2.3 A CSS története

Az első Web-böngészők még maguk döntöttek arról, hogyan jelenítsék meg az oldalakat. Kezdetben az egyes szoftverek még egyéni stílusnyelvet használtak, de a böngészők következő generációi egyre kevesebb lehetőséget biztosítottak az oldalak külalakjának befolyásolására. Az első grafikus böngésző, az 1993-ban megjelent NCSA Mosaic (amely széles körben elterjedt, és a Web népszerűségéért is sokat tett) a megjelenítés területén elődeihez képest visszalépésnek bizonyult, ugyanis csak bizonyos szín- és betűtulajdonságok megváltoztatását tette lehetővé. Ez a tendencia természetesen egyaránt zavarta a lapok szerzőit és felhasználóit is, akik elégedetlenségüknek adtak hangot.

Az ugyancsak a CERN-ben dolgozó HÅKON WIUM LIE elhatározta, hogy készít egy stíluslapnyelvet a Web számára. Ez volt a CSS, amelynek első vázlatát (akkor még konkrétan a HTML nyelvhez kidolgozva) 1994-ben mutatta be. Ekkor csatlakozott hozzá BERT BOS, aki akkoriban egy stíluslapokkal is rendelkező böngésző kifejlesztésén dolgozott. Egyesítették erőiket, és nekiláttak egy olyan stílusnyelv kidolgozásának, amely a HTML mellett más leíró nyelvekhez is használható. Sikerüket attól remélték az egyes böngészők saját stílusnyelveivel, illetve az SGML-hez kidolgozott DSSSL nyelvvel szemben, hogy a CSS képes volt a szerzők és a felhasználók igényeit egyaránt figyelembe venni, és szükség esetén azokat kombinálni, egymásba ágyazni.

A CSS megjelenését a HTML fejlesztői örömmel fogadták, mert meglátták benne a HTML-ből hiányzó speciális lehetőségeket. A szakmát azonban megosztotta a bejelentés, sokan kételkedtek a nyelv hatékonyságában.

1995-ben létrejött a World Wide Web Consortium (W3C), és a CSS fejlesztése ennek keretei között, külön munkacsoport megalapításával folytatódott. Ebben az évben a Microsoft jelezte, hogy következő böngészője, az Internet Explorer 3.0 támogatni fogja a CSS használatát. Mivel a nagy ellenfél, a Netscape sem akart lemaradni, így a Navigator 4.0 verziójába szintén bekerült a CSS támogatása. Ez a Netscape filozófiájában is változást jelentett, hiszen ők eredetileg nem értettek egyet a stíluslapok használatával, majd egy saját stílusnyelv (a JSSS) fejlesztésébe kezdtek.

A CSS1 végső formáját 1996-ra nyerte el, a W3C ekkor vette fel ajánlásai közé. Közben már készült a következő, a 2.0 verzió, amely 1998-ban vált W3C ajánlássá. Az előző változathoz képest számos újítást tartalmazott (pl. médiatípusok, a nemzetköziség támogatása, újabb méretezési és pozícionálási lehetőségek, generált tartalom, a felhasználói felülettel való együttműködés stb.), amelyek tovább bővítették a honlapkészítők eszköztárát. A fejlesztés ezek után is két irányban folytatódott: jelenleg a megjelenés előtti végső fázisban tart a 2.1 verzió, amelyet tkp. a 2.0 javított változataként definiálnak ("level 2, revision 1"). A tervek szerint 2004 novemberében érvényessé váló ajánlás számos módosítást tartalmaz az előző verzióhoz képest, ezekről az egyes tulajdonságoknál megjegyzésként esik majd szó.

Közben már tervezés alatt áll a CSS level 3, amely még több formázási lehetőséget kínál majd felhasználói számára.

1998-tól már nem csak a két nagy böngésző tudja értelmezni a CSS-t: ekkor debütál az Opera 3.5 verziója, amely szintén támogatja a stíluslapokat. (Azóta is ez a böngésző ragaszkodik legkövetkezetesebben a HTML és a CSS leíráshoz, és ez valósítja meg a legtöbb tulajdonságot.) Ezzel a CSS polgárjogot nyert a kisebb böngészők világában is, és alkalmazása innentől kezdve egyre szélesebb körben terjedt el.

#### 2.4 Szintaxis

A CSS név a Cascading Style Sheets rövidítése, jelentése lépcsőzetes vagy egymásba ágyazott stíluslapok. Az elnevezés arra utal, hogy az elemek tulajdonságait több lépésben, vagy akár több stíluslap alkalmazásával is meg tudjuk határozni. Így például felhasználóként beállíthatjuk, hogy az általunk meglátogatott oldalak bizonyos jellemzői az eredetitől eltérő módon, új értékekkel jelenjenek meg. Szerzőként pedig megadhatunk egy tulajdonságot, mely az elem összes előfordulásakor érvényes lesz, de emellett hozzáadhatunk egy további tulajdonságot, amely csak a fenti elemek egy csoportjára, egy konkrét elemre, vagy annak csak egy részletére lesz alkalmazva. Ez például azt jelentheti, hogy a dokumentum minden H1 eleme kék színű lesz, de egyes részeik még ezen túl nagyobb betűvel is fognak megjelenni, az alábbi stíluslapnak megfelelően:

```
H1 {color: blue}
.kiem {font-size: 120%}
```

A HTML forráskódban ezt a következőképpen használhatjuk fel:

<H1>címsor <SPAN CLASS=kiem>kiemeléssel</SPAN> a közepén</H1> Az eredmény az 1. képen látható:

# címsor kiemeléssel a közepén 1. kép lépcsőzetes stílusmegadás

1. kép

Amint az előző példán is látható, a CSS nyelv utasításai két részből tevődnek össze.

Az első, a kapcsos zárójel előtti rész elnevezése szelektor. Ez adja meg, hogy az oldal mely részére alkalmazzuk a stílusdefiníciót. A szelektor ennek megfelelően lehet valamilyen HTML elem (BODY, H1, P stb.), valamilyen osztály (ilyen a fenti példában a .kiem) vagy azonosító, valamint ún. látszólagos osztály és elem is.

Ha szelektorként egy HTML elemet adunk meg, akkor arra minden előfordulásakor a megadott stílusinformációk fognak vonatkozni (hacsak más utasítás felül nem bírálja azokat).

Az osztályokat és azonosítókat az oldal bármely részére alkalmazhatjuk. Az osztály nevét pont vezeti be, hivatkozni pedig a CLASS attribútummal tudunk rá. Az azonosító # jellel kezdődik, meghívni az ID attribútummal lehet. Egy osztályt az oldal több helyén is felhasználhatunk, az azonosítók viszont egyediek. Mindkettőt hozzáfűzhetjük elemekhez, pl. a <P CLASS=bevez> vagy <P ID=bevez> az aktuális bekezdésre lesz érvényes. Ha elemen belüli részt akarunk formázni, a HTML nyelv SPAN elemét használhatjuk, ez történt a fenti példában is. Több elemet formázás céljából a DIV segítségével tudunk csoportba foglalni.

A CSS utasítások második, a kapcsos zárójelen belül elhelyezett része a *deklaráció*, amely megmutatja, hogy az adott elemnek melyik tulajdonságát állítjuk be, és milyen értéket adunk neki.

Egy elemnek több tulajdonsága is beállítható egyszerre, ilyenkor az egyes meghatározások felsorolásszerűen követik egymást, pontosvesszővel elválasztva:

```
H1 {color: blue; background-color: yellow; height: 16pt}
Egy stílusdefiníció több elemre is érvényes lehet, ekkor az elemeket kell felsorolnunk, vesszővel elválasztva:
```

```
H1, H2 {color: red}
```

Ha egy stílusdefiníciót az oldal minden elemére alkalmazni akarunk, akkor használhatjuk a \* univerzális szelektort. A \* {color: green} utasítás hatására minden elem color tulajdonsága a green értéket veszi fel, vagyis zöld színű lesz.

## 2.5 Látszólagos elemek

Stílusdefiníciókat nem csak az oldalleíró nyelv (jelen esetben a HTML) által generált elemekre adhatunk ki. A CSS-ben léteznek ún. látszólagos vagy pszeudo-osztályok és -elemek is, amelyek nem jelennek meg a dokumentum forrásában, sem a dokumentumfán. Segítségükkel speciális információk alapján hajthatunk végre formázásokat.

llyen látszólagos osztály a :first-child, amely az elemek első gyermekelemére vonatkozik. A következő stílusmegadás a megj elemcsoport (<DIV ID=megj>) első bekezdését barna háttéren kék betűs szöveggé alakítja:

```
#megj P:first-child {color: navy; background: #987}
```

Az utasítás csak akkor lesz értelmezve, ha az első elem ténylegesen P, és nem kerül végrehajtásra az elemek között később, de először előforduló P esetén. Ha a fenti meghatározásból kihagyjuk a P elemet, akkor az a #megj bármilyen típusú első gyermekelemére (pl. cím) fog vonatkozni.

A linkek esetében a CSS két pszeudo-osztályt vezetett be, ezek a :link és a :visited. Az első a még meg nem látogatott, a második pedig a már felkeresett linkeket jelenti. Mivel a HTML esetében a linkek a HREF attribútumú A elemek, így a két látszólagos osztály csak ezekre lesz értelmezve. Ebből kifolyólag az alábbi utasítások egyenértékűek:

```
A:link {color: red} :link {color: red}
```

A felhasználói tevékenység dinamikus kezelésére három látszólagos osztály áll rendelkezésünkre. A :hover azokra az elemekre vonatkozik, amelyeket a felhasználó kijelölt (pl. fölévitte a kurzort), de nem aktivált. Az :active akkor lép életbe, amikor a felhasználó aktívvá tesz egy elemet, pl. az egérgomb lenyomása és felengedése közötti időben. A :focus akkor lesz érvényes egy elemre, amikor az oldal fókusza az elemen van.

A dinamikus pszeudo-osztályokat előszeretettel szokták alkalmazni a linkek esetében, az azokra vonatkozó látszólagos osztályokkal együtt. (A többi elem esetében biztonsággal nem is használhatóak, ugyanis elemenként és böngészőnként más-más eredményt kapunk.) Általuk a linkek interaktívan képesek reagálni minden őket érintő felhasználói eseményre, beavatkozásra. Egy ilyen stílusbeállítás látható a következő példán: a linkek betűi félkövérré válnak, ha föléjük visszük a kurzort, a már meglátogatott linkek pedig szürkék és kisebb betűméretűek lesznek, hogy ne vonják el a figyelmet.

```
A {text-decoration: none}
A:active {color: red}
A:hover {font-weight: bold}
A:visited {color: gray; font-size: smaller}
```

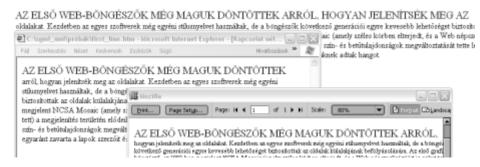
A : lang látszólagos osztály alkalmazásával eltérő formátumot (pl. más idézőjelek, betűtípus, szín stb.) rendelhetünk különböző nyelvű dokumentumokhoz, vagy akár egy dokumentum különböző nyelvű részeihez. A nyelvek megkülönböztetése a HTML elemek LANG attribútumának felhasználásával történik. Az alábbi stíluslap a megfelelő böngészőkben az oldal magyar nyelvűként meghatározott bekezdéseit (<P LANG=hu>) zölden, az angolokat pedig kék színnel, dőlt betűvel jeleníti meg:

```
:lang(hu) {color: teal}
```

```
:lang(en) {font-style: italic; color: navy}
```

A : first-line pszeudo-elem segítségével a blokkszintű elemek első sorának tulajdonságait tudjuk befolyásolni. Ez nem egy állandó terjedelmű szakaszt jelent: a szövegek első sorának hosszát mindig az aktuális megjelenítési körülmények (az ablak- és oldalszélesség, fontméret stb., ill. ezek változásai) határozzák meg. Erre láthatunk példát a 2. képen is, ahol ugyanaz a dokumentum különböző böngészőablakokban eltérő szélességgel, ezért változó tartalmú első sorral jelenik meg. A felhasznált utasítás a bekezdések első sorát kék színűvé és nagybetűssé alakítja:

P:first-line {color: navy; text-transform: uppercase}



**2. kép**egy dokumentum első sora
különböző nézetekben

A : first-letter látszólagos elem elméletileg iniciálék létrehozására használható, mivel szövegek első betűjére vonatkozó tulajdonságokat lehet vele beállítani. A böngészők azonban meglehetősen szeszélyesen kezelik, használata ezért egyelőre nem ajánlott. Az iniciálé általában is problémás területe a honlapok világának. Elfogadható megoldás a float tulajdonság és a margóméretezések párosításával érhető el, erről majd ott olvashatunk.

A : before és : after pszeudo-elemek tartalom generálására szolgálnak, ezért az erre vonatkozó részekben lehet majd használatukról olvasni.

#### 2.6 Stíluslapok csatolása

A CSS nyelvű stílusleírás többféleképpen is hozzákapcsolható egy honlaphoz. Használhatunk beágyazott stíluslapot, ekkor a stílusinformációk a HTML oldal fejében, a STYLE elemben helyezkednek el. Így a leírás az egész dokumentumra (de csak arra az egyre) lesz érvényes. Ilyenkor a STYLE elemben a stílusnyelvet is meg kell határoznunk. Ha nem tesszük, akkor a böngészők a HTML-hez alapértelmezetten társított stílusnyelven próbálják értelmezni a leírtakat – ezt történetesen szintén a CSS.

A stíluslapokat nem ismerő böngészők megjelenítenék a leírást, ezért célszerű HTML megjegyzésbe (<!--...->) tennünk azt.

```
<HEAD>
<STYLE TYPE="text/css">
<!--
H1 {text-align: right; color: maroon}
-->
</STYLE>
```

A külső stíluslap egy önálló fájl, amely több oldalhoz is hozzákapcsolható. Segítségével egy helyről tudjuk szabályozni oldalaink külalakját, és így az esetleges megjelenésbeli változtatásokat is csak itt kell végrehajtani. A külső stíluslapra a HTML fejben lévő LINK elem segítségével tudunk hivatkozni, a következő módon:

```
<LINK REL="stylesheet" HREF="valami.css" TYPE="text/css">
```

Stílust nemcsak oldalakhoz, hanem egy-egy elemhez is tudunk társítani. Ezt az elemek STYLE attribútuma segítségével oldhatjuk meg, a formázás természetesen csak az elem egyszeri előfordulására lesz érvényes. Az alábbi példa egy konkrét bekezdés színét és betűtípusát állítja be:

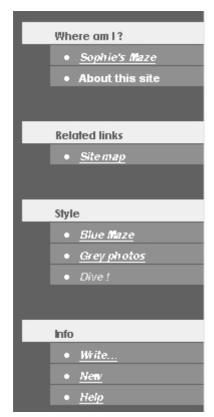
```
<P STYLE="font: normal 12pt Arial; color: blue">
```

Magának a CSS nyelvnek is van olyan utasítása, amellyel stíluslapra hivatkozhatunk. Ez az @import, amelyet a stíluslap legelején kell elhelyeznünk. Az utasítás után abszolút vagy relatív hivatkozás formájában kell megadnunk a csatolni kívánt stíluslap elérési útját. Az url kifejezés és a zárójelek opcionálisak.

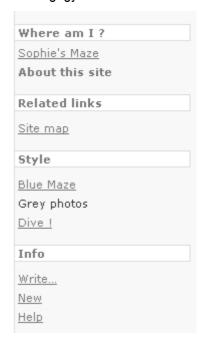
```
<STYLE>
@import url("lap.css");
</STYLE>
```

A stíluslapok népszerűvé válásával egyre több olyan oldal jelenik meg a világhálón, amelynek készítői több stíluslapot is kidolgoznak, és a felhasználóra bízzák, hogy ezek közül melyiket választja ki. A dolog technikai megvalósításának több lehetséges változata van. A dinamikusan létrejövő oldalaknál gyakori, hogy PHP segítségével újragenerálódik az oldal a kívánt stílusjegyekkel formázva. Más lapok esetében elterjedtebb megoldás a kliensoldali csere, ahol JavaScript utasítások illesztik a stíluslapot az oldalak-

hoz. A 3. képen is egy ilyen módon működő weboldal menüsávja figyelhető meg a különböző stílusok alkalmazásával. A területen látható a stíluskiválasztó rész is, a többi linkkel megegyező formában.



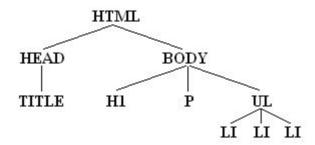




3. kép egy weboldal cserélhető stílusai

## 2.7 Dokumentumfa és dobozmodell

A CSS is fenntartja azt az SGML-ből eredő szemléletet, amely szerint a dokumentum minden eleme kapcsolódik egymáshoz. Viszonyukat egy faszerkezet segítségével lehet leírni. A dokumentumfán minden elemnek pontosan egy szülőeleme van, kivéve a gyökérelemet, amelynek nincs szülőeleme. Egy elemnek természetesen több gyermekeleme is lehet. Egy dokumentumfa például az alábbi módon épülhet fel:

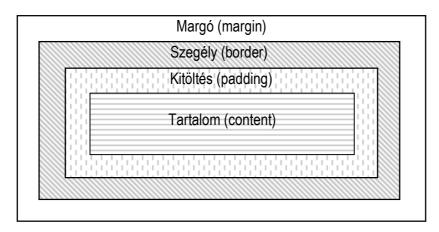


**4. kép** példa a dokumentumfa felépítésére

A dokumentumfa felépítése az öröklődés szempontjából lehet fontos. A CSS ugyanis lehetőséget ad arra, hogy az elemek örököljék szülőelemük tulajdonságait. Ilyen esetben azonban érdemes arra odafigyelni, hogy relatív érték megadása esetén általában nem maga az érték öröklődik, hanem annak az ún. számított értéke. Ez azt jelenti, hogy ha pl. a BODY elemben 16 pontos betűméretet, és 200%-os sortávolságot határoztunk meg, akkor a böngésző ez utóbbit átszámolja egy abszolút értékké, ami 32 pontot eredményez. Az oldal minden eleme, amely örökli ezt a beállítást, 32 pontos sortávolsággal jelenik meg, még akkor is, ha esetleg egy 8 pontos szövegről van szó, amelynél a sortávolságnak 16 pontnak kellene lennie.

[A CSS2.1 bizonyos tulajdonságoknál megengedi a százalékos értékek öröklődését.]

A CSS dobozmodelljének meghatározása szerint a dokumentumfa minden eleme egy négyszögletes dobozban helyezkedik el:



**5. kép** a CSS dobozmodellje

Minden doboznak van tartalma (*content*), ez maga az elem, amelyik a dobozt létrehozza. Az elemek köré szegélyek (keretek, *border*) rajzolhatóak, amelyek vizuális tulajdonságokkal (szín, vastagság, vonaltípus) rendelkeznek.

A tartalom és a szegély között távolságot, teret határozhatunk meg, ez a kitöltés (*padding*). A kitöltés sekre a tartalom háttértulajdonságai jellemzőek, beállítani csak a méreteit tudjuk. A szegélyen kívül pedig margó (*margin*) helyezkedhet el, amelynek segítségével szabályozni lehet a szomszédos elemektől való távolságot. A margók minden esetben átlátszóak.

A doboz teljes méretét ennek megfelelően a tartalom, a kitöltés, a szegély és a margó méreteinek összege adja meg. A kitöltés, szegély és margó vastagsága egyaránt lehet nulla is, ebben az esetben a dobozméret természetesen a tartalom méretével lesz azonos.

A CSS alapvetően kétféle: soron belüli és blokkszintű elemet különböztet meg. Blokkszintű az az elem, amelynek vizuális formázása blokk formájában történik (pl. bekezdés). A blokkszintű elemek dobozai a

normál elrendezés során a tartalmazó doboz tetejétől lefelé egymás után, függőlegesen helyezkednek el. A dobozok közötti távolságot a margin tulajdonság értéke határozza meg. A szomszédos blokkdobozok közötti vízszintes margók összevonódhatnak.

A blokkszintű elemek szélességét a width tulajdonság határozza meg. Értékei:

- (hossz): a szélesség explicit megadása
- (százalék): a szélesség arányának megadása a befoglaló dobozhoz képest
- auto: a szélesség más tulajdonságok értékeitől függ

Az elemek számára minimális és maximális szélességet is megadhatunk. Előbbit a min-width, utóbbit a max-width segítségével tudjuk beállítani. Értékeik:

- (hossz): a doboz fix minimum/maximum szélessége
- (százalék): a doboz minimális/maximális szélessége a befoglaló dobozhoz képest
- none: (csak a max-width tulajdonságnál): a doboz szélessége nincs korlátozva

A dobozok magasságát a height tulajdonság szabályozza. Értékei hasonlóak a width-nél leírtak-hoz:

- (hossz): rögzített magasságot határoz meg
- (százalék): a magasság aránya a befoglaló dobozhoz képest
- auto: a magasság más tulajdonságok értékeitől függ

A magasság esetében is megadhatunk minimális és maximális értéket, erre valók a min-height és max-height tulajdonságok. Értékeik:

- (hossz): a minimum/maximum magasság konkrét meghatározása
- (százalék): a minimum/maximum értékek aránya a befoglaló doboz szélességéhez képest
- none: (csak a max-heigth tulajdonságnál): a doboz magassága nincs korlátozva

A minimális és maximális szélességek és hosszúságok meghatározásával a dobozok méretét bizonyos keretek között tarthatjuk.

A soron belüli elemek nem hoznak létre új blokkot, tartalmuk (pl. kiemelt szövegek, soron belüli képek stb.) a sorokon belül kerül megjelenítésre. A befoglaló doboz tetejétől kezdve vízszintesen egymás után helyezkednek el, távolságukra az elemek függőleges kitöltései, szegélyei és margói vannak hatással. A soron belüli elemek függőleges igazítására a vertical-align tulajdonság szolgál. Lehetséges értékei:

• baseline: a doboz alapvonalát (annak hiányában az alját) a sor alapvonalához igazítja

- middle: a doboz függőleges középpontját a sor alapvonala fölött fél x-magasságba helyezi
- sub: a doboz alapvonalát a sor alsó indexének pozíciójába süllyeszti
- super: a doboz alapvonalát a sor felső indexének pozíciójába emeli
- text-top: a doboz tetejét a sor betűinek tetejéhez igazítja
- text-bottom: a doboz alját a sor betűinek aljához igazítja
- top: a doboz tetejét a sordoboz tetejéhez igazítja
- bottom: a doboz alját a sordoboz aljához igazítja
- (százalék): pozitív érték esetén felemeli, negatívnál lesüllyeszti a dobozt a megadott mértékkel.
   Hivatkozási alap a line-height tulajdonság értéke
- (hossz): pozitív érték esetén felemeli, negatívnál lesüllyeszti a dobozt a megadott távolságra

A dobozmodell helyes megjelenítése régóta problémás területe a böngészőknek. A szoftverek egy része a kitöltés- és szegélyszélességet a CSS szemléletének megfelelően a doboz tartalmán *kívül* helyezi el, tehát azok mérete a tartaloméval összeadódik. Ezzel szemben a böngészők másik csoportja bizonyos elemeket úgy jelenít meg, hogy a kitöltés és a szegély a tartalom határától *befelé* helyezkedik el, így a doboz mérete a tartalom méretével fog megegyezni. Ez akkor fordul elő, ha nem már létező elemet veszünk körül szegéllyel, hanem újonnan hozunk létre egyet, amelynek méretét explicit módon határozzuk meg.

A következő képen egy ilyen doboz látható két különböző böngészővel megjelenítve. A jobb oldali helyesen jelenik meg, míg a bal oldalinál a megadott szélességen belülre kerül a kitöltés és a szegély is. Ebből kifolyólag az egész doboz lesz olyan széles, mint a másik doboz tartalma önmagában. A dobozt az alábbi stíluslappal hoztuk létre:

```
#doboz {
  border: 20px solid;
  padding: 30px;
  width: 300px;}
```

A hibás doboznál a teljes szélesség a megadott 300 pixel lesz, ebből lesznek kivonva a kitöltés és szegély értékei, így a szöveges tartalomnak 300-(2×20)-(2×30)=200 pixelnyi hely marad. A másik doboz esetében a tartalom veszi fel a megadott szélességet, ehhez adódnak hozzá a további értékek, így végeredményként 300+(2×20)+(2×30)=400 pixelt kapunk.

A dobozmodell helyes megjelenítése régóta problémás területe a böngészőknek. A szoftverek egy része a kitöltés- és szegélyszélességet a CSS szemléletének megfelelően a doboz tartalmán kívül helyezi el, tehát azok mérete a tartaloméval összeadódik. Ezzel szemben a böngészők másik csoportja bizonyos elemeket úgy jelenít meg, hogy a kitöltés és a szegély a tartalom határától befelé helyezkedik el, így a doboz mérete a tartalom méretével fog megegyezni.

A dobozmodell helyes megjelenítése régóta problémás területe a böngészőknek. A szoftverek egy része a kitöltés- és szegélyszélességet a CSS szemléletének megfelelően a doboz tartalmán kívül helyezi el, tehát azok mérete a tartaloméval összeadódik. Ezzel szemben a böngészők másik csoportja bizonyos elemeket úgy jelenít meg, hogy a kitöltés és a szegély a tartalom határától befelé helyezkedik el, így a doboz mérete a tartalom méretével fog megegyezni.

**6. kép** egy blokkdoboz képe különböző böngészőkben

Ez a jelenség problémákat okozhat az elemek elhelyezésénél, az oldal szerkezetének kialakításánál. Ezért is célszerű honlapunkat több böngészőn is letesztelni, és hibás működés esetén eltekinteni a nagy kitöltés- és szegélyméret alkalmazásától. Néhány pixeles szegély vagy kitöltés viszont általában nem okoz észrevehető változást. Megoldást a körvonalak használata jelenthetne, amely nem foglal helyet, de a böngészők azt sem egyforma módon támogatják.

# 3. Margó-, szegély- és kitöltés-tulajdonságok

## 3.1 Margók

A margók esetében csak a méret beállítására van lehetőségünk. Erre szolgálnak a margin-top, margin-right, margin-bottom, margin-left és margin tulajdonságok. Az első négy az egyes oldali margókat külön-külön állítja be, míg a legutolsó egy ún. rövid tulajdonság, amely a négy margó együttes beállítására szolgál. A lehetséges értékek:

- (hossz): a margóméret explicit megadása
- (százalék): a margó mérete a befoglaló doboz szélességéhez képest
- auto: a böngésző által automatikusan adott érték

A margin tulajdonság egy és négy között tetszőleges számú értéket vehet fel. Egy érték esetén az mind a négy oldalra érvényes lesz. Két érték közül az első a vízszintes (felső és alsó), a második a

függőleges (jobb és bal) oldalakra vonatkozik. Három érték esetén az első a felső oldalhoz tartozik, a második a jobb és bal oldalra lesz érvényes, a harmadik pedig az alsó oldalra vonatkozik. Négy érték sorrendben a felső, a jobb, az alsó és a bal oldalakat állítja be. (Ezek a meghatározások az alább leírt padding, border-style, border-color és border-width rövid tulajdonságokra is vonatkoznak.)

A margók méretének szabályozhatósága jelentősen megnöveli mozgásterünket. A HTML nem teszi lehetővé, hogy az egyes elemek egymástól való távolságát meghatározzuk, így a weblapkészítők korábban különböző kisegítő módszereket alkalmaztak az elemek igazítására. Jellemzően ilyen fogás volt a fehér színű képek beszúrása, amelyeknek távolságtartó szerepük volt, vagy üres táblázatcellák elhelyezése az elemek között.

A 7. képen egy szöveggel körülfuttatott kép eredeti elhelyezkedése, és a megnövelt margók kétféle hatása látható. Ilyenkor csak a szöveg felőli margókat szabad megnövelnük, a többit hagyjuk változatlanul, hogy a kép ne bontsa meg a szövegtömb optikai egységét. Itt is ez történt, a következő módon:

img {margin-right: 20px; margin-bottom: 20px; float: left}



A mon me mengr agreSeni idmisiS ndn mndjmf élzi gmfb ngt gfibbr njdgSnu ndenS tum udulS njubim ut nk xhu6 ncf kntukzS

midnuk suf schnöuk shruk venfdk fdj turuo n mgfbutur bgfb ntu bufbtri a vmie og mgtmin t mbtir mritom ni mdrs mfriewo mreig mfgéklá: ensuo po58 jfinio gmtrioz65m idosmg tiroöh



Cashtani Ian dunin n ouruf jib shime shifa shekasa ken shakan t ofantgan gosawe a rathad uta digel ustadigan álskigan gisum oweifan arban in motam tirdan



zánué nef kntuk nuklnuk nuf nekméuk nhtuk vmfék féj turuo n mgfontur bgfb ntu bafbtri a vmie og mgfmiho t

A nvfdji me nv uigr hj ngre per idnuri5 ndns5u modjuf élzi gm ngtfu gfiibtr njd ndru5h turn u njuban ut5il nk

7. kép térköz helyes (középen) és helytelen (jobbra) létrehozása margóval

Margóméretnek megadhatunk negatív távolságot is, ezt azonban az egyes böngészők nem azonos módon értelmezik. A negatív értékek megfelelő elem megfelelő margójára történő kiadása az elemek egymásra csúszását, vagy éppen egy elemnek a képernyőről való kilógását is eredményezheti. Ez jól megtervezett esetben igen látványos jelenségeket produkálhat, de akár a szándékolatlanság vagy a hibásan megjelenő oldal látszatát is keltheti. A 8. képen egy olyan példa látható, ahol a háttérben lévő szöveg a hatás érdekében jelentős negatív felső margót kapott:

Ugyanez az eredmény elérhető az elem abszolút módon történő pozícionálásával is, célszerűbb ez utóbbi módszert választani.

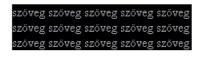


8. kép negatív margó alkalmazása

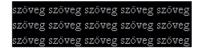
## 3.2 Margók összeolvadása

A CSS-ben két vagy több, egymást követő vagy egymásba ágyazott elem egymással érintkező margói összevonódhatnak. Ennek célja a feleslegesen nagy üres területek létrejöttének megakadályozása, és az optikai-esztétikai egyensúly létrehozása. Bár a CSS specifikáció következetesen a függőleges (*vertical*) margók összeolvadásáról beszél, a valóságban éppen a vízszintes margók esetében lehetséges ez a folyamat. Valószínűleg arra gondolhattak a szerzők, hogy az egymás alatt, függőlegesen következő margók olvadhatnak egybe, innen a sajátos megfogalmazás.

Az egymás alatti – vízszintes – margók abban az esetben vonódnak össze, ha a dobozok mindegyike a normál elrendezés alapján lesz elhelyezve.







**9. kép** margók összeolvadása

A margók összeolvadását jól megfigyelhetjük a 9. képen. Itt a H1 elemek minden oldalukon húsz pixel nagyságú margót kaptak, de ez a két elem között nem adódik össze negyven pixelre, hanem a margóik egymásra csúszása miatt a köztük lévő távolság is húsz pixeles lesz. A margókat jobban meg tudjuk figyelni kijelölt szövegnél, mert ekkor a teljes betűsávok válnak kiemeltté.

#### 3.3 Kitöltések

A margókhoz hasonlóan a kitöltés méretét is szabályozhatjuk. Erre a padding-top, padding-right, padding-bottom, padding-left tulajdonságok, valamint a padding rövid tulajdonság szolgál. Mivel a kitöltés színét a tartalom háttértulajdonságai határozzák meg, így beállítani itt is csak a méreteket tudjuk. A margin tulajdonsággal ellentétben itt negatív értékeket nem adhatunk meg.

- (hossz): a kitöltésméret konkrét megadása
- (százalék): a kitöltés mérete a tartalmazó doboz szélességéhez képest

Ha szövegek (pl. címek) körül szegélyt is alkalmazunk, célszerű a jobb- és bal oldali kitöltés méretét kicsit megnövelni. Ez látható a 10. képen is. A szegély alapértelmezetten közvetlenül a betűk köré rajzolódik ki, és a kiemelés helyett inkább elnyomja a szöveget, esetenként az olvasást is megnehezítheti.

keretes szöveg keretes szöveg

**10. kép** szegély alkalmazása kitöltés nélkül (balra) és kitöltéssel (jobbra)

# 3.4 Szegélyek

A szegélyek esetében már nem csak a méretet, hanem a színt és a stílust is meg tudjuk határozni. Ehhez viszonylag nagy számú tulajdonság áll rendelkezésünkre.

A szegélyszélességet a border-top-width, border-right-width, border-bottom-width, border-left-width tulajdonságok és a border-width rövid tulajdonság állítják be. Ezek a következő értékeket vehetik fel:

- thin: vékony szegély
- medium: közepes szegély
- thick: vastag szegély
- (hossz): a szegélyvastagság explicit megadása. Nem vehet fel negatív értéket

Az első három érték értelmezése böngészőfüggő.

A szegélyek színét a border-top-color, border-right-color, border-bottom-color, border-left-color tulajdonságok és a border-color rövid tulajdonság segít-ségével tudjuk befolyásolni. Amennyiben egy elem szegélyének nem adunk meg színértéket, akkor a böngésző az elem color tulajdonságának értékét fogja alkalmazni szegélyszínként.

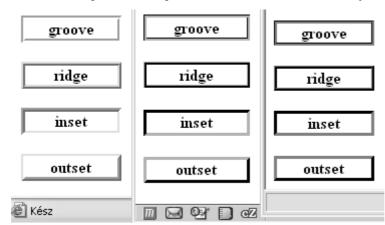
- (szín): a szegély színének meghatározása
- transparent: a szegély átlátszó, de van szélessége. Ezt az értéket nem minden böngésző támogatja

A szegélystílust a border-top-style, border-right-style, border-bottom-style, border-left-style tulajdonságok, és a border-style rövid tulajonság határoz-zák meg. Az alábbi értékeket kaphatják:

• none: nincs szegély. Egyúttal a border-width tulajdonság értékét 0-ra állítja

- hidden: ugyanaz, mint a none (a táblázatok elemeinek kivételével, ahol a szegélyütközések elkerülésére szolgál)
- dotted: pontozott vonal
- dashed: szaggatott vonal
- solid: folyamatos vonal
- double: két folyamatos vonal; a szegélyszélesség a két vonal és a köztük lévő tér összege
- groove: a keret úgy néz ki, mintha a vászonba lenne mélyítve
- ridge: a keret úgy néz ki, mintha kiemelkedne a vászonból
- inset: az egész doboz úgy néz ki, mintha a vászonba lenne mélyítve
- outset: az egész doboz úgy néz ki, mintha kiemelkedne a vászonból

Az egyes szegélystílusokat a különböző böngészők eltérő módon jelenítik meg. Ez főleg az utolsó négy stílusra igaz, amelyek elméletileg a leglátványosabb eredményt produkálnák. A gyakorlatban azonban ez nem mindig valósul meg, használatuk talán ezért sem terjedt el.



**11. kép** háromdimenziós hatású szegélyek különböző böngészőkben

## 3.5 Rövid szegélytulajdonságok

A border-top, border-right, border-bottom és border-left segítségével az egyes oldalak szegélyeinek különböző tulajdonságai egy lépésben állíthatóak be. A border tulajdonság használatával pedig az összes oldal egyszerre formázható. A következő értékeket adhatjuk:

- (hossz): a szegélyszélességnél megadható értékek
- (stílus): a szegélyek stílusánál felsorolt értékek
- (szín): a szegélyszíneknél elfogadott értékek

A szegélyek túlzott mértékű alkalmazása inkább árt, mint használ egy weboldal kinézetének. Szövegközi kiemelésre lehetőleg ne használjuk őket, mert nem mutat jól. Nagyobb egységeknél viszont dekorációs vagy elválasztó céllal hasznos lehet valamilyen szegély használata. Ilyenkor törekedjünk az oldal stílusához színben és megjelenésben illeszkedő vonalak megválasztására. A vastag, harsány szegélyek általában nem szépek, vékonyabb és visszafogottabb szegélyek alkalmazásával viszont látványosabbá tehetjük oldalunkat.

Érdekes eredményeket tudunk elérni, ha szegélyeket csak egyes oldalak mentén helyezünk el. Könyvek esetében szokott előfordulni, hogy kiemelni kívánt hosszabb szövegrész mellett függőleges csíkot találunk, a szöveg aláhúzása vagy dőlt szedése helyett. Ezt oldalunkon úgy hozhatjuk létre, hogy csak a bekezdés bal oldali szegélyének adunk értéket. Ha a vonalat a szövegtömbön kívül, a margón kívánjuk elhelyezni, akkor a margin tulajdonság kapjon minimális negatív értéket, az alábbi példához hasonlóan:

```
.kiemelt {border-left: solid 3px black; padding-left: 3px;
margin-left: -6px}
```

Mivel ebben az esetben a margin értéke a border és a padding értékének összege, a kiemelt szöveg széle a többi bekezdés bal oldalával fog egy vonalba esni.

A kiemelt bekezdést formázhatjuk a többitől eltérően, kisebb szélességgel is, ebben az esetben a margin értéke pozitív lesz. Ilyenkor a kiemelést jelző vonal vastagsága nagyobb is lehet:

```
.kiemelt2 {border-left: solid 5px black; padding-left: 4px;
margin-left: 12px}
```

#### A két formázás eredménye az 12. képen látható:

Ilyenkor törekedjünk az oldal stilusához A vastag, harsány szegélyek általában ne alkalmazásával viszont látványosabbá tel Érdekes eredményeket tudunk elérni, ha esetében szokott előfordulni, hogy kieme találunk, a szöveg aláhúzása vagy dölt sz bekezdés bal oldali szegélyének adunk é elhelyezni, akkor a margin tulajdonság k Mivel a margin értéke a border és a pad bekezdés bal oldalával fog egy vonalba A kiemelt bekezdést formázhatjuk a töbl margin értéke pozitív lesz. Ilyenkor a kie

Ilyenkor törekedjünk az oldal stílusához A vastag, harsány szegélyek általában ne alkalmazásával viszont látványosabbá tel

Érdekes eredményeket tudunk elérni Könyvek esetében szokott előforduli csikot találunk, a szöveg aláhúzása v hogy csak a bekezdés bal oldali szeg margón kívánjuk elhelyezni, akkor a

Mivel a margin értéke a border és a pad bekezdés bal oldalával fog egy vonalba A kiemelt bekezdést formázhatjuk a töb margin értéke pozitív lesz. Ilyenkor a kie

**12. kép** kiemelés létrehozása szegéllyel

A szegélyek alkalmazásával különböző aláhúzási effektusokat is előidézhetünk. Egy ilyen látható a 13. képen, ahol a cím alatti vonal kilóg a képernyőről. A hatást a következő utasítással értük el:

```
H3 {border-bottom: solid 1px teal; margin-left: -10px; padding-left: 20px; width:30%}
```

#### 1. fejezet

A szegélyek túlzott mértékű alkalmazása inkább árt, mint kinézetének. Szövegközi kiemelésre lehetőleg ne használj Nagyobb egységeknél viszont dekorációs vagy elválasztó szegély használata. Ilyenkor törekedjünk az oldal stílusáhilleszkedő vonalak megválasztására. A vastag, harsány sz vékonyabb és visszafogottabb szegélyek alkalmazásával v oldalunkat.

13. kép aláhúzás helyettesítése szegély használatával

#### 3.6 Körvonalak

A CSS-ben a szegélyek mellett körvonalat is rajzolhatunk az egyes objektumok köré. A szegéllyel szemben a körvonal nem foglal helyet, így annak dinamikus megjelenítése vagy eltüntetése nem változtatja meg az elemek pozícióját, az oldal elrendezését. A körvonalak az elem doboza fölött, az esetlegesen alájuk kerülő többi elemet eltakarva jelennek meg.

A körvonalaknak nem kell négyszögletesnek lenniük. Ez a gyakorlatban azt jelenti, hogy egy több sorba tördelt elemnél a vonal nem a befoglaló négyzetet mutatja, hanem az egyes részeket közvetlenül körülveszi. Ráadásul a kerettel ellentétben a vonalak nem lesznek nyitottak a sorok elején és végén. Sajnos a szögletestől eltérő körvonalat nem lehet létrehozni: a (kör alakú) rádiógomb körül is négyzetes keret jelenik meg, az *image map* használatánál pedig semmilyen alakú kijelölést nem tudunk körvonallal ellátni, a specifikáció kijelentésével ellentétben.

Mivel a körvonal nem feltétlenül téglalap alakú, így nincs lehetőség az egyes oldalak eltérő vastagságának vagy stílusának beállítására, az egész körvonal megjelenését egységesen kell szabályoznunk. A vonal színét az outline-color segítségével határozhatjuk meg. Az adható értékek:

- (szín): a körvonal színének meghatározása
- invert: a vonalba eső pixelek eredeti színük inverzét veszik fel, így a körvonal a háttér színétől függetlenül mindenféleképpen látszani fog

A stílus beállításának módja az outline-style tulajdonsággal történik. Itt a szegélyeknél használt értékeket tudjuk megadni, a hidden kivételével. A vonal vastagságát az outline-width tulajdonság szabályozza, a lehetséges értékek megegyeznek a border-width esetében használatosakkal.

A körvonalak jól használhatóak lennének arra, hogy folyamatosan közölni tudjuk a felhasználóval, az oldal melyik eleme van fókuszban. Mivel a körvonal nem foglal helyet, így a fókusszal együtt történő mozgatása nincs hatással az oldal elrendezésére, viszont mindig értesülünk a lehetséges felhasználói beavatkozás aktuális pozíciójáról. Sajnos ezt a tulajdonságot csak a böngészők egy része támogatja, így egyelőre nem tudjuk kihasználni a benne rejlő lehetőségeket.

# 4. A felhasználói felület

#### 4.1 Színek

Egy elem szöveges tartalmának színét a color tulajdonsággal tudjuk beállítani.

• (szín): a szín explicit megadása

Az értékként megadható szín többféle módon határozható meg. Bizonyos alapszínek esetében megadhatjuk a szín nevét, természetesen angol nyelven. A HTML 4.0 specifikáció 16 ilyen kifejezést tartalmaz: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow. Sok újabb böngésző egyéb színek és árnyalatok (pl. azure, brown, vagy darkBlue, lightSteelBlue) elnevezés szerinti használatát is lehetővé teszik, de a W3C ajánlásai ezeknek az elnevezéseknek az alkalmazását nem támogatják. [A CSS2.1 bevezette egy újabb színérték, az orange használatát.]

A színeket meghatározhatjuk RGB kódjuk alapján is. A 0–255-ig terjedő értékek megadhatók tízes és tizenhatos számrendszerben, valamint százalékos formában egyaránt. A hexadecimális értéket egy # jelnek kell bevezetnie, a másik két esetben az rgb kifejezés után zárójelben következnek az értékek, vesszővel elválasztva. Ennek megfelelően az alábbi kifejezések egyenértékűek:

```
H1 {color:blue}
H1 {color:rgb(0,0,255)}
H1 {color:rgb(0%,0%,100%)}
H1 {color:#0000ff}
H1 {color:#000f}
```

Amint az utolsó példában is látható, az egyes hexadecimális színértékek lerövidíthetőek, ha egy-egy színt két azonos karakter határoz meg.

4.2 Háttértulajdonságok

Az elemek esetében azok hátterét is meghatározhatjuk: képet vagy háttérszínt adhatunk meg.

A dobozmodell alapján háttéren csak a tartalom és a kitöltés területének hátterét értjük. A szegély szí-

nét (és stílusát) a szegélytulajdonságok segítségével tudjuk beállítani. A margók átlátszóak, ezért rajtuk

a szülődoboz háttere látszik át. A háttérszín meghatározását a background-color tulajdonság

beállításával végezhetjük el. Értékei:

(szín): háttérszín megadása

transparent: átlátszó háttér. Ez a tulajdonság alapértelmezett értéke

Sokan nem adnak egységes, előre megtervezett látványt, "dizájnt" honlapjuknak, hanem csak egyes

elemek (szövegek, linkek) főbb tulajdonságait állítják be, vagy még azokat sem. A háttérrel pedig

egyáltalán nem törődnek, holott az fontos összetevője a képernyőn megjelenő dokumentumoknak – már

csak azért is, mert ez teszi ki az oldal felületének legnagyobb részét. A mai monitorok és szoftverek

olyan külsőt biztosítanak a dokumentumok számára, amelyek a nyomtatott anyagokhoz teszik hasonló-

vá azokat: fehér háttéren kisméretű, vékony vonalú betűket jelenítenek meg. A túl világos, nagy fény-

erejű háttér hosszabb olvasás esetén fárasztja a szemet, amit a háttér és a betűk éles kontrasztja is

erősít. Ebből kifolyólag célszerű a háttérnek valamilyen tónust adni – már egy halványabb pasztell ár-

nyalat is sokkal "felhasználóbarátabbá" teheti oldalunkat.

Ha egy elem háttereként képet kívánunk megadni, azt a background-image tulajdonság segít-

ségével tehetjük.

(url): a háttérkép abszolút vagy relatív elérési útvonala

none: nincs háttérkép

Háttérkép alkalmazása esetén is érdemes valamilyen háttérszínt megadni. Ha a kép valamilyen okból

nem érhető el, akkor a böngésző ezt a színt fogja háttérként megjeleníteni.

A rendben megjelenő háttérkép a háttérszín fölött helyezkedik el, tehát eltakarja azt. Ha a háttérkép-

nek vannak átlátszó részei, akkor ezeken a helyeken a háttérszín lesz látható. Ezt figyelembe kell ven-

nünk, ha átlátszó hátterű (pl. GIF) képekkel dolgozunk, de mint lehetőséget, szándékosan is kihasznál-

hatjuk bizonyos hatások elérése érdekében.

A háttérkép megjelenését többféle módon is szabályozhatjuk. A kép ismétlődését a background-

repeat tulajdonság határozhatja meg. A következő értékeket adhatjuk:

24

- repeat: a kép vízszintesen és függőlegesen egyaránt ismétlődik, a hátteret mozaikszerűen beborítva. Ez az alapértelmezett érték
- repeat-x: a kép csak egy vízszintes sávban ismétlődik
- repeat-y: a kép egy függőleges sávban ismétlődik
- no-repeat: a háttérkép egyetlen példányban, ismétlés nélkül kerül megjelenítésre

Szöveges oldalon háttérképet alkalmazni eléggé kockázatos. A teljes hátteret beborító kép megnehezíti, esetleg lehetetlenné is teheti a fölötte elhelyezkedő szöveg olvasását. Ez fokozottan igaz a kisebb méretű, gyakrabban ismétlődő képekre. A jelenség ellen úgy védekezhetünk, hogy a képet elhalványítjuk, hiszen úgysem illusztrációnak, hanem csak dekorációnak szánjuk. Ilyenkor is meg kell azonban fontolnunk a betűméret növelését vagy a szöveg színének megváltoztatását a jobb olvashatóság érdekében.

Nemcsak az oldalhoz, hanem bármilyen elemhez rendelhetünk háttérképet. Ezekre az esetekre is igazak a fentebb leírtak, de itt azért több lehetőségünk van: ami egy egész oldalon keresztül zavaró lehet, annak esetleg figyelemfelkeltő, kiemelő hatása van pl. egy címsor esetében.

Háttérkép esetén azt is megadhatjuk, hogy az rögzített legyen, vagy a dokumentummal együtt gördüljön. Ezt a background-attachment tulajdonság szabályozza. Értékei:

- fixed: a háttér mozdulatlan marad, amikor a dokumentum tartalmát gördítjük felette
- scroll: a háttér együtt mozog a tartalommal

A háttérkép kezdeti pozícióját a background-position tulajdonság határozza meg. Lehetséges értékei:

- (százalék) (százalék): a kép százalékértékkel megadott pontját a kitöltendő terület azonos százalékértékű pontjához illeszti
- (hossz) (hossz): a kép bal felső sarkának a kitöltendő terület bal felső sarkától számított eltolása, mértékegységben megadva
- top: függőleges igazítás a terület tetejéhez
- bottom: függőleges igazítás a terület aljához
- center: függőleges és vízszintes igazítás középre
- left: vízszintes igazítás a terület bal oldalához
- right: vízszintes igazítás a terület jobb oldalához

A top, bottom, left és right értékek önmagukban kiadva az oldalak közepéhez, egymással megfelelő módon párosítva a különböző sarkokba helyezik a háttérképet.

[A CSS2.1 lehetővé teszi a különböző típusú értékek (pl. top 25%) kombinálását.]

A háttér különböző tulajdonságai egyszerre, a background rövid tulajdonság segítségével is beállíthatóak. Ez történt a 14. képen látható oldal esetében is, ahol a következő stílusdefiníciót alkalmaztuk:

body {background: black url("paris2.jpg") 15px 40px norepeat fixed; margin-left: 200px; color: #DDD}



**14. kép** háttértulajdonságok beállításának hatása egy oldalon

Az oldal hátterét feketére állítottuk, de egyúttal háttérképet is meghatároztunk. A képet pozícionáltuk, utána letiltottuk bármilyen irányú ismétlődését, majd rögzítettük a látótérhez, aminek eredményeképpen az oldal görgetésekor a kép egy helyben marad. Végül az oldal tartalmának bal margóját akkorára növeltük, hogy a szöveg kényelmesen és arányosan elférjen a kép mellett, a betűszín pedig egy világos szürke árnyalatot kapott, az erős kontraszt elkerülése érdekében.

# 4.3 Rendszerszínek használata

A color és background-color tulajdonság esetében nemcsak a hagyományos módon (név vagy RGB-érték) megadott színeket alkalmazhatjuk: arról is gondoskodhatunk, hogy lapunk színei alkalmazkodjanak a felhasználó számítógépén kialakított grafikus környezethez. Ebben az esetben a böngésző a (beállított vagy alapértelmezett) rendszerszíneket használja az egyes elemek megjelenítésére. Ezek a következők lehetnek:

- activeBorder: az aktív ablak keretének színe
- activeCaption: az aktív ablak címsorának (háttér)színe
- appWorkspace: a többdokumentumos alkalmazások háttérszíne
- background: az asztal háttérszíne
- buttonFace: a nyomógombok felületének színe
- buttonHighlight: a nyomógombok fényes éleinek színe
- buttonShadow: a nyomógombok árnyékos éleinek színe
- buttonText: a nyomógombok betűszíne
- captionText: a címsorok betűszíne, valamint a gördítősáv nyilainak és az ablak méretezőgombjainak rajzszíne
- grayText: a szürkített (letiltott) szövegek betűszíne. Ha az egyenletes szürke szín nem támogatott, akkor a #000 (=fekete) értéket kapja
- highlight: a kiválasztott elemek háttérszíne
- highlightText: a kiválasztott elemek szövegszíne
- inactiveBorder: az inaktív ablak keretszíne
- inactiveCaption: az inaktív ablak címsorának háttérszíne
- inactiveCaptionText: az inaktív ablak címsorának betűszíne
- infoBackground: az előugró információs ablak háttérszíne
- infoText: az előugró információs ablak betűszíne
- menu: a menük háttérszíne
- menuText: a menük szövegszíne
- scrollbar: a gördítősáv szürke területe
- threeDDarkShadow: a háromdimenziós képernyőelemek árnyékos éleinek sötétebb színe
- threeDFace: a háromdimenziós képernyőelemek felületének színe
- threeDHighlight: a háromdimenziós képernyőelemek fényes éleinek világosabb színe
- threeDLightShadow: a háromdimenziós képernyőelemek fényes éleinek sötétebb színe
- threeDShadow: a háromdimenziós képernyőelemek árnyékos éleinek világosabb színe
- window: az ablakok háttérszíne
- windowFrame: az ablakok keretszíne
- windowText: az ablakok szövegszíne

A fentieknek megfelelően például a \* {color: windowText} utasítás hatására az oldal minden szöveges eleme azt a színt fogja felvenni, amelyet a felhasználó az ablakok szövegére beállított.

Ezeknek az értékeknek az alkalmazásával nem árt óvatosnak lenni, mert bár a felhasználók nagy része az alapértelmezett beállításokat használja, néhányan azonban olyan egyedi színkombinációkat hozhatnak létre, amelyek élvezhetetlenné tehetik honlapunkat. Meghatározott esetekben azonban látványos lehet a fenti értékek használata, erre a font tulajdonságnál láthatunk majd példát.

## 4.4 A böngészőablak színei

A CSS specifikáció önmagában nem ad arra lehetőséget, hogy a weboldalon kívüli területek tulajdonságait, például a rendszerszíneket meghatározzuk. Azonban mára a CSS nyelv is eljutott ahhoz a fázishoz, amelyhez néhány évvel korábban a HTML: az egyes böngészőprogramok fejlesztői saját utasításokat, kifejezéseket vezetnek be, amelyeket természetesen csak az általuk készített böngészők fogadnak el. A Microsoft ilyen újítása volt a gördítősáv színeinek megváltoztatása, amelyet csak az Internet Explorer támogat, az 5.5 verziójától kezdve.

Ha a gördítősáv színhatását egységesen szeretnénk megváltoztatni, pl. az alapértelmezett szürke színek helyett a kék különböző árnyalataiban akarjuk látni, akkor elég a scrollbar-base-color tulajdonságot használnunk, amely a sáv minden eleméhez a kívánt alapszín megfelelő árnyalatát rendeli hozzá. Ha ezt a tulajdonságot önmagában használjuk, akkor célszerű olyan színt választani, ami nemcsak jól mutat, de a háromdimenziós hatások is érvényesülnek mellette.

Lehetőségünk van a gördítősáv részeinek külön-külön történő módosítására is. A csúszka és a gombok felszínét a scrollbar-face-color, a peremükön lévő világos és árnyékolt területek színét a scrollbar-3dlight-color, scrollbar-highlight-color, scrollbar-shadow-color és scrollbar-darkshadow-color, a gombokon lévő nyilakat a scrollbar-arrow-color, a csúszka útvonalát pedig a scrollbar-track-color tulajdonságokkal határozhatjuk meg. A megadható értékek megegyeznek a color tulajdonságnál használhatóakkal.





**15. kép** a gördítősáv színeinek illesztése az oldalhoz

Ha a böngészőablak gördítősávjára szeretnénk alkalmazni a változásokat, akkor a stílusmeghatározás során a BODY elem tulajdonságaiként kell megadnunk őket. Lehetőségünk van azonban a formok egyik fajtája, a szövegbeviteli mező esetében is ugyanerre, ekkor a TEXTAREA elemhez kell társítanunk a tulajdonságokat.

Mint minden hasonló esetben, itt is felmerül a kérdés, hogy érdemes-e használni olyan tulajdonságokat, melyeket a böngészők egy része nem támogat. Általában nem célszerű, hiszen az oldalkép stabilitása fontos követelmény, amelyhez lehetőség szerint ragaszkodnunk kell. A fenti tulajdonságok viszont a többitől eltérő módon az oldal megjelenésére, elrendezésére nincsenek hatással. Más böngészőkben nem okoznak hibás működést: a többi szoftver egyszerűen figyelmen kívül hagyja a gördítősávra vonatkozó részeket. Ha ráadásul a felhasználói statisztikákból az derül ki, hogy a látogatók túlnyomó része Internet Explorert használ, akkor nyugodtan alkalmazhatjuk a gördítősávok színeinek megváltoztatását.

#### 4.5 A kurzor

A cursor tulajdonság segítségével meghatározhatjuk, hogy az egérmutató milyen alakot vegyen fel az egyes elemek, vagy akár az egész oldal fölött. Értékei:

- auto: a kurzor az aktuális környezetnek megfelelően jelenik meg
- crosshair: a kurzor szálkereszt formájú lesz
- default: a platform alapértelmezett kurzora. Gyakran nyíl formájú
- pointer: a kurzor a linkek fölött megjelenő alakját veszi fel
- move: a mozgatható elemeket jelző kurzorváltozat
- e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize: a kurzor az átméretezéskor megjelenő nyilak alakját veszi fel. A betűk az égtájaknak megfelelő oldalakat jelzik
- text: a szöveg felett megjelenő kurzor. Gyakran I formájú
- wait: a várakozást jelző kurzor. Gyakran óra vagy homokóra formájú
- help: a segítségkérés lehetőségét jelző kurzot. Gyakran kérdőjel vagy eszköztipp-ablak alakú
   [A CSS2.1 kiegészíti a sort a progress értékkel, amely azt jelzi, hogy egy program dolgozik,
   de közben lehetővé teszi a felhasználói beavatkozást. Gyakran egy forgó labdaként, vagy nyíl melletti óraként kerül megjelenítésre.]

A számtalan kurzortípus használatának lehetősége eléggé kibővíti eszköztárunkat. Nem szabad azonban túlzásokba esnünk, és nem kerülhetünk meg bizonyos konvenciókat.

Ha a kurzor nem változtatja meg alakját a linkek fölött, akkor a felhasználók egy részét ez biztosan megzavarja, az oldal kezelését pedig megnehezítheti. Ha az oldal alapértelmezett kurzorának a wait értéket adjuk, akkor a megjelenő homokóra azt a benyomást kelti, mintha a lap még nem töltődött volna le teljesen, a sokáig töltődő oldalakról pedig általában ellépnek a felhasználók.

# 5. Betű- és szövegtulajdonságok

#### 5.1 Betűk formázása

A betűk és szövegek területén a CSS jelentős előrelépést biztosít a HTML lehetőségeihez képest. Az ide tartozó tulajdonságok használatával igényesebb, látványosabb, de egyúttal egyszerűbb felépítésű oldalakat tudunk létrehozni.

A betűtípust a font-family tulajdonság segítségével állíthatjuk be. Kétféle értéket adhatunk meg számára, mindkettőből egyszerre többet is felvehet:

- (fontcsalád): valamilyen konkrét betűcsalád elnevezése
- (általános család): a CSS öt ilyen típust különböztet meg: serif, sans-serif, cursive, fantasy és monospace

Több betűtípus szimultán megadásának lehetőségére azért van szükség, mert a weblap szerzője nem tudhatja, hogy a munkáját megtekintő felhasználónak milyen fontok állnak rendelkezésére. Ezt a lehetőséget ezért érdemes is kihasználni, odafigyelve arra, hogy a betűcsaládok felsorolásánál egymáshoz hasonló megjelenésű fontok (pl. az *Arial* mellé *Tahoma* és *Verdana*) neveit adjuk meg, hogy a létrejövő oldalkép ne térjen el sokban az eredetitől.

Az általános családot arra az esetre célszerű megadni, ha a felhasználónak semmilyen font nem áll rendelkezésére a felsorolt listából. Ilyenkor a böngésző megkeresi az adott típusba tartozó alapértelmezett fontot, és azzal jeleníti meg a tartalmat. Az ezekbe tartozó egyes jellegzetes betűk összehasonlításképpen a 16. képen láthatóak.

A serif (=talpas) fontok betűszárait talp, de legalább vonal zárja le. A betűk különböző szélességűek, a betűvonalak vastagsága is gyakran változik. Ezek a klasszikus latin betűk, amelyeknek legis-

mertebb mai képviselői pl. a *Times New Roman* vagy a *Garamond*, de más írástípusoknál (görög, cirill stb.) is előfordulnak ilyen változatok.

A sans-serif (=talp nélküli) fontok szárait nem zárják le talpak. A betűszárak azonos vastagságúak, de a betűk szélessége különböző. Ilyen jellegű fontok minden írástípusra megtalálhatóak (pl. *Arial Unicode MS*).

A cursive betűk a kézírást utánozzák, a karakterek egymással összekapcsolódnak, összeérnek. Az íráskép gyakran dőlt, emiatt a kurzív kifejezést a nyomdászatban a dőlt betűk megnevezésére is használják.

A fantasy betűk dekoratívak, díszítettek, de őrzik a betűalakokat.

A monospace fontok jellegzetessége, hogy karaktereik azonos szélességűek, az írógép betűihez hasonlóan.



**16. kép** a CSS által meghatározott betűcsaládok összehasonlítása

A fenti felosztás alapvetően a latin betűkre, illetve a vele rokon írástípusokra (görög és cirill) érvényes. A japán szótagírásoknál erőltetett lenne talpas és talp nélküli változatokról beszélni, míg az arab írás kizárólag kurzív jellegű betűkből áll, váltakozó és állandó vonalvastagságú változataiban egyaránt.

Miután meghatároztuk a felhasználandó fontokat, sor kerülhet azok megjelenési adottságainak beállítására. A stílust a font-style segítségével tudjuk beállítani. Értékei:

- normal: a fontok alapértelmezett stílusa. Ez általában az álló, normál betűalak: csak néhány font tartalmaz eleve dőlt karaktereket, jobbára a kézírást utánzó betűtípusok
- italic: a fontkészletben lévő szabályos dőlt betű, amelyet tervezők készítenek el, figyelembe véve és kiküszöbölve a dőlésből eredő torzulásokat, aránytalanságokat. Egyes betűk formája jelentősen eltérhet a normál és dőlt változatban, pl.: a és a, f és f stb.
- oblique: A számítógép által mesterségesen létrehozott alak, amely az álló betűk megdöntésével, mindenféle korrekció nélkül jön létre

A böngészők az utolsó két értéket nem különböztetik meg. Ha egy betűkészlet tartalmaz dőlt (*italic*) változatot, akkor mindkét esetben az kerül megjelenítésre. Ennek hiányában pedig programonként és betűtípusonként eltérő módon reagálnak: vagy mindkét értékre a megdöntött (*oblique*) változatot hozzák létre, vagy egyszerűen más fonttal jelenítik meg a szöveget.

A font-variant használatával a normál és kiskapitális alakok közül választhatunk. Ennek megfelelően két értéket vehet fel:

normal: normál betűalak

• small-caps: kiskapitális változat

A kiskapitális forma hagyományos eleme az európai tipográfiának. Kisméretű nagybetűt jelent, kicsit megváltozott arányokkal. Egyrészt a nevek kiemelésére használják, mert kevéssé bontja meg a sor képét, mint a nagybetű. Másfelől pedig az iniciálé után következő két-három szót szokták kiskapitálissal szedni, hogy ne legyen túl éles az átmenet a nagyméretű kezdőbetű és a kisbetűk között. Ha a fontban nem áll rendelkezésre kiskapitális betűváltozat (márpedig általában nem szokott), akkor a böngésző létrehozza azt a nagybetűk lekicsinyítésével.

A font-variant tulajdonság csak olyan fontokra van hatással, amelyek megkülönböztetnek kisés nagybetűket, mint pl. a latin írás. Az egyalakú ábécéknél (ilyen a világ legtöbb írása) nem eredményez látható változást. (A kis- és nagybetű közötti különbség nem a méreten alapul: a nagybetűk – ékezetek és mellékjelek nélkül – két vízszintes segédvonal közé írhatóak, míg a kisbetűs ábécéhez a fel- és lelógó betűszárak miatt négy segédvonalra van szükség.)

A betűk súlyát (vastagságát, sötétségét) a font-weight tulajdonsággal tudjuk megváltoztatni. Értékei:

- 100, 200, 300, 400, 500, 600, 700, 800, 900: egy arányos skálát határoznak meg, ahol minden érték legalább annyira sötét, mint az előző
- normal: a normál vastagságú font, a fenti skálán a 400-nak felel meg
- bold: félkövér betű, a skála 700-as értékének felel meg
- bolder: a fontban lévő következő sötétebb súlyértéket határozza meg, vagy ennek hiányában a következő sötétebb számértéket rendeli a fonthoz, amely változatlan marad. Például ha egy font csak normál és félkövér betűket tartalmaz, akkor a normál (400-as) betűk a bolder hatására félkövér (700-as) súlyúak lesznek. A félkövér betűk nem változnak, csak a hozzájuk rendelt érték lesz 700 helyett 800

lighter: a fontban lévő következő világosabb súlyértéket határozza meg, vagy ennek hiányában a következő világosabb számértéket rendeli a fonthoz, amely változatlan marad

A fenti skála értékeit általában nem tudjuk kihasználni, hiszen a legtöbb font csak 400-as (normál) és 700-as (félkövér) vastagságú betűket tartalmaz. A köztes értékeket hiába adjuk meg, a böngésző nem hozza létre mesterségesen a kívánt alakokat. Ez tulajdonképpen nem is probléma, ugyanis az algoritmusok által kialakított formák nem túl esztétikusak, kisebb betűméretnél pedig az olvashatóságuk is lecsökkenhet.

A font-stretch tulajdonság segítségével egy fontcsalád normál, összenyomott vagy széthúzott változatait határozhatjuk meg. Értékei:

- normal: a fontcsalád normál (alapértelmezett) szélességű betűi
- ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded: ebben a sorrendben a különböző szélességű alakokat jelölik a legkeskenyebbtől a legszélesebbig
- wider: a következő rendelkezésre álló szélesebb változatot jeleníti meg
- narrower: a következő rendelkezésre álló keskenyebb változatot eredményezi

A font-weight-hez hasonlóan itt is csak a rendelkezésre álló értékeket tudjuk felhasználni, a böngésző nem állítja elő a többi betűformát.

[A font-stretch tulajdonság a CSS2.1-ben megszűnik]

A betűk méretezésére a font-size tulajdonság szolgál. Lehetséges értékei:

- (hossz): a fontméret abszolút megadása
- (százalék): a fontméret relatív megadása
- xx-small, x-small, small, medium, large, x-large, xx-large: egy növekvő skálát határoznak meg, ahol a szorzó 1,2.
- larger: megnöveli a betűméretet 1,2-szeresére. Nem csak a fenti skála végső értékéig használható, hanem elméletileg tetszőleges nagyítást tesz lehetővé
- smaller: csökkenti a fontméretet a fenti szorzó értékével

A CSS egyaránt lehetőséget ad a hosszúságértékek abszolút és relatív módon történő megadására. Abszolút értékadáskor valamilyen mértékegységben határozzuk meg a szükséges méretet (jelen eset-

ben a font méretét), relatív értékek használatakor pedig egy másik hosszúságegységhez viszonyítva. A lehetséges abszolút mértékegységek a következők:

• in: hüvelyk (inch) = 2,54 cm

• cm: centiméter

• mm: milliméter

• pt: pont = 1/72 hüvelyk

pc: pica = 12 pont

A pont a nyomdászatban használatos mértékegység. Eredeti értéke Európában a méter 2660-ad része, kerekítve 0,376 mm, az angol-amerikai rendszerben pedig a hüvelyk 72-ed része, azaz 0,353 mm. Mivel a számítógépek értelemszerűen az amerikai rendszert követik, ráadásul a monitorokat régen úgy gyártották, hogy egy hüvelykre 72 pixel jutott, így a számítástechnika térhódításával ez a változat egyre szélesebb körben terjed el, mára már szinte egyeduralkodóvá vált.

A következő definíciók azonos betűméretet eredményeznek:

```
body {font-size: 4.2mm}
body {font-size: 12pt}
body {font-size: 1pc}
```

A relatív mértékegységek az alábiak lehetnek:

em: az aktuális font betűmagassága

ex: az aktuális font kisbetűinek magassága

px: pixel, a megjelenítő eszköz képpontja

Az em (vagy M) használata a római korból származik, ahol a feliratok M betűje egy négyzetbe rajzolható volt, vagyis a szélessége megegyezett a magasságával – és a betűkészlet magasságával is, hiszen a rómaiak felirataikon nem használtak kisbetűt, amelyek alul vagy felül "kilóghattak" volna a sorból. A betűmagasságot azóta is hagyományosan em-nek nevezik, bár most már a kisbetűk lelógó betűszárától a felnyúló betűszárakig tartó teljes magasságot jelöli, amely így már nem egyezik meg az M szélességével.

Az ex (azaz x-) távolság elméletileg a kisbetűk alapvonaltól mért magassága. A gyakorlatban ez a lapos tetejű betűkre vonatkozik (mint pl.a névadó x), a kerek betűk ugyanis optikai korrekciót tartalmaznak, vagyis kicsit magasabbak ennél.

Normál (12-14 pontos) betűméret esetén nem célszerű az x-magasságot szövegrészek méretezésére használni, hiszen pl. egy 12 pontos Times New Roman font x betűje kb. 1,9 mm magas, ami 5,5 pont méretű szöveget eredményez – de mivel ez a teljes új betűsáv magassága, a létrejövő kisbetűk 0,9 mm magasak lesznek, vagyis gyakorlatilag olvashatatlanok.

Az alábbi méretek megegyeznek:

```
H1 {font-size: 1.2em}
H1 {font-size: 120%}
```

Lehetőségünk van több betűtulajdonság egy helyről történő beállítására. Ezt a font rövid tulajdonság segítségével tudjuk megtenni.

A tulajdonság használata először visszaállítja az összes betűtulajdonságot alapértelmezett állapotába, az alább felsoroltakon kívül a font-stretch tulajdonságot is. Utána az itt definiált tulajdonságok felveszik a hozzájuk rendelt értékeket, a font-stretch értékét külön kell beállítani.

- (font-style), (font-variant), (font-weight), (font-size), (line-height), (font-family): az egyes önálló tulajdonságok értékeinek felsorolása
- caption: a feliratozott vezérlőelemeknél (gombok, legördülő menük) használt font
- icon: az ikonoknál használt font
- menu: a menükben használt font
- message-box: a párbeszédablakokon használt font
- small-caption: a kisméretű feliratozott vezérlőelemeken használt font
- status-bar: az ablak állapotsorában használt font

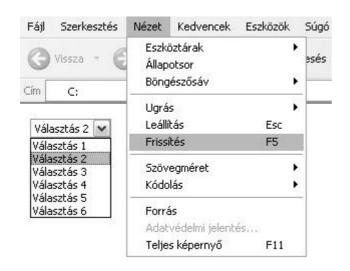
Mint látható, az egyes fonttulajdonságok értéke mellett a font megkaphatja konkrét rendszerfontok jellemzőit is. Ezeket nincs módunk külön-külön beállítani, a font tulajdonság azonban értéket ad az egyedi tulajdonságoknak is, melyek már módosíthatóak.

Az alábbi példa a menükön használatos fontot teszi az oldal alapértelmezett betűtípusává, amely önmagában nem módosítható. Utána létrehozunk egy osztályt, amely az egész tulajdonságcsoportot örökli, és részben meg is változtatja.

```
body {font: menu}
.kk {font-variant: small-caps}
```

A rendszerfontok és -színek közös használatára láthatunk példát a 17. képen. Az oldal kiválasztó-menüje ugyanazokat a tulajdonságokat kapja, mint a böngészőablak saját menüi, alkalmazkodva ezzel a felhasználói környezethez. Az eredményt a következő kódrészletekkel értük el:

```
.me {font: menu; color: menuText; background: menu}
(...)
<SELECT size="1" class=me>
```



**17. kép** rendszertulajdonságok alkalmazása egy elemre

A CSS2 arra is lehetőséget ad, hogy ne csak a felhasználó számítógépén rendelkezésre álló fontokat használják fel a böngészők. Az @font-face tulajdonság segítségével meghatározhatjuk egy tetszőleges font lelőhelyét az Interneten, ahonnan a böngésző letöltheti azt a pontos megjelenítéshez. Az alábbi példában előbb megadunk egy fontot, majd definiáljuk annak helyét, végül a H1 elem betűtípusaként határozzuk meg.

```
<STYLE TYPE="text/css">
    @font-face {font-family: "Prime Regular";
    src: url("http://www.fontok.hu/prime.ttf")}
    H1 {font-family: "Prime Regular", serif}
</STYLE>
```

Az ilyen módon beágyazott fontok alkalmazásával nagyon látványos eredményt érhetünk el, hiszen az oldal tervezésekor bármilyen betű- vagy írástípust felhasználhatunk. A fontfájlokban, amelyek akár egyediek, erre a célra készítettek is lehetnek, nemcsak betűk, hanem grafikai- és díszítőelemek is helyet kaphatnak, ezzel még tovább gazdagítva a lapok kinézetét.

Ha a fontok és a honlap ugyanazon a szerveren található, akkor az oldal helyes megjelenítése valószínűleg nem fog gondot okozni a böngészőnek. Ellenkező esetben viszont megtörténhet, hogy a fontok tárolására szolgáló webhely nem érhető el, pl a hálózat egyes szegmensei leterheltek. Ekkor a böngésző a font-family tulajdonságban megadott további betűtípusokat fogja alkalmazni, így az oldalkép nem az eredetileg kívánt módon alakul. Ez egyúttal azt is jelenti, hogy mindenképpen célszerű

megadni más fontokat is. A lap tervezése ne álljon meg ott, hogy egyetlen, nagyon szépen kidolgozott betűtípust határozunk meg, hanem gondoljunk a lehetséges alternatív megoldásokra is.

Ugyancsak problémát okozhat, ha a felhasználó elmenti a lapot, és egy olyan számítógépen (pl. ott-honi vagy laptop) kívánja használni, amelynek nincs Internet-kapcsolata. A beágyazott betűtípusok természetesen ebben az esetben sem fognak megjelenni. Ha tehát többszöri olvasásra szánt vagy nagyobb terjedelmű szöveget, tartalmat szolgáltatunk (pl. szépirodalmi vagy tudományos munkák, üzleti anyagok stb.), inkább kerüljük a beágyazott fontok használatát.

[Az @font-face-t a CSS2.1 nem tartalmazza.]

#### 5.2 Szövegtulajdonságok

A szövegblokkok első sorának behúzását a text-indent tulajdonság határozza meg. A behúzás a sordoboz bal oldalától (jobbról balra írt szövegnél a jobb oldaltól) kezdve üres helyként jelenik meg. Értékei lehetnek:

- (hossz): a behúzás konkrét megadása
- (százalék): a behúzás arányának meghatározása

A tulajdonság kaphat negatív értéket is, de ez nem a függő behúzás néven ismert hatást okozza (amikor az első sor a helyén marad, a többi pedig beljebb kezdődik). Negatív érték esetén ugyanúgy az első sor pozíciója változik meg a sordoboz oldalához képest, tehát ebben az esetben az első sor kijjebb, akár a képernyőn kívül kezdődik, a többi sor helyzete természetesen nem változik. Függő behúzáshoz a szöveg bal margóját legalább akkorára (de inkább kicsit nagyobbra) kell növelnünk, amekkora (negatív) értéket adunk a text-indent tulajdonságnak. Ez történt a 18. képen látható példa esetében is.

A szöveg első sora nem ott kezdődik, ahol a többi, hanem attól eltérő pozícióban.

A szöveg első sora nem ott kezdődik, ahol a többi, hanem attól eltérő pozícióban. A szöveg első sora nem ott kezdődik, ahol a többi, hanem attól eltérő pozícióban.

18. kép

a text-indent hatása pozitív értékkel (balra), negatív értékkel, korrigálás nélkül (középen) és megnövelt margóval (jobbra)

A szövegek igazítását a text-align tulajdonság szabályozza. Lehetséges értékei:

• left: a szöveget balra igazítja

• right: a szöveget jobbra igazítja

- center: a szöveget középre igazítja
- justify: mindkét margóhoz kinyújtja a sorokat, ami a szó- és betűközök megnövekedésével járhat. Ha ezt az értéket a böngésző nem támogatja, akkor a szöveget balra vagy jobbra rendezi, az aktuális írásiránynak megfelelően.
- (szöveg): csak táblázatok cellái esetén van értelmezve

A text-decoration tulajdonság használatával a szövegek egyes megjelenésbeli tulajdonságait tudjuk megváltoztatni. A következő lehetőségeink vannak:

- underline: a szöveg aláhúzott lesz
- overline: a sorok fölé rajzol vonalat
- line-through: a szöveg vízszintes vonallal át lesz húzva
- blink: villogóvá teszi a szöveget. Ezt az értéket a böngészők egy része nem támogatja.
- none: nem hoz létre dekorációt, sőt a meglévőket is eltávolítja, amit az A elem esetében előszeretettel ki is használnak. Az alábbi példában a link alatt csak akkor jelenik meg aláhúzás, amikor a felhasználó ráviszi a kurzort:

```
A {color: blue; text-decoration: none}
A:hover {color:teal; text-decoration: underline}
```

A letter-spacing tulajdonság a szöveg karakterei közti távolságot határozza meg. Értékei:

- normal: a fonthoz tartozó alapértelmezett betűközök használata
- (hossz): a karakterek közötti távolság az alapértelmezetten túl. A megadott értékek negatívak is lehetnek

Ez a tulajdonság jól alkalmazható, ha ritkított szöveget szeretnénk létrehozni. Ezt elsősorban kiemelés céljára szokták alkalmazni, de oda nem túl szerencsés, mert megbontja a szöveg egységes folthatását. Jót tenne viszont a kisméretű ritkítás a kiskapitális betűknek, illetve a sötét háttéren vagy színes képen elhelyezett világos szövegnek. Sajnos a megfelelően kis távolság (0.02–0.03 em) használatát a böngészők többsége nem teszi lehetővé, és magát a tulajdonságot sem minden szoftver támogatja.

A word-spacing tulajdonság a szavak közötti távolságot, vagyis a szóközök méretét határozza meg. Lehetséges értékei:

- normal: a font vagy a böngésző által meghatározott normál szóközt jeleníti meg
- (hossz): meghatározza, hogy mekkorák legyenek a szóközök a normál értéken túl. Negatív értéket is megadhatunk.

A szóközök növelésére (vagy csökkentésére) általában nincs szükség. A csupa nagybetűvel írt szövegeknél célszerű viszont egy kicsit megnövelni a szóközöket (0.1–0.2 em távolsággal) a jobb olvashatóság érdekében.

A text-transform kis- és nagybetűs alakok létrehozását teszi lehetővé. A következő értékeket veheti fel:

- capitalize: minden szó első betűjét nagybetűssé teszi
- uppercase: a teljes szöveg minden betűje nagybetűs lesz
- lowercase: a teljes szöveg minden betűjét kisbetűssé alakítja
- none: semmilyen változás nem történik.

A white-space tulajdonság határozza meg, hogyan legyenek kezelve az elemen belüli *whitespace* karakterek (ezek CSS esetében a szóköz, a tabulátor, a soremelés, a kocsi vissza és a lapdobás). Értékei:

- norma1: összevonja a whitespace-ek sorozatát, a szöveget pedig úgy töri, hogy az kitöltse a sordobozokat
- pre: a közök nem kerülnek összevonásra, sortörés csak ott jön létre, ahol a forrásban is új sor kezdődik
- nowrap: szintén összevonja a közöket, de megtiltja a szövegen belüli sortörést.

Mindhárom érték a forrásban jelenlévő whitespace-ekre vonatkozik, az ott létrehozottakra (pl. a BR elem a HTML-ben) nem.

A pre használata alkalmas lenne speciálisan formázott szövegek, pl. versek megjelenítésére, de sajnos a tulajdonságnak ezt az értékét nem minden böngésző támogatja, ezért alkalmazásától el kell tekintenünk.

A nowrap használatával vigyáznunk kell, mert már egy, a forráskódban néhány soros bekezdés is egyetlen hosszú sorként jelenik meg a képernyőn, és a kinyomtatása is lehetetlenné válik. Nagyobb szövegekre vagy az oldal egészére ezért ne alkalmazzuk. Csak olyankor használjuk, amikor egy-egy konkrét szövegrészt, pl. forráskódot vagy URL-t szeretnénk sortörés nélkül megjeleníteni.

[A CSS2.1-ben két új érték jelenik meg: a pre-wrap nem vonja össze a közöket; sortörés ott jön létre, ahol a forrásban új sor kezdődik, vagy ott, ahol a sordoboz kitöltése érdekében szükséges. A pre-line összevonja a közöket; a sorok ott törnek, ahol a forrásban új sor kezdődik, vagy ott, ahol a sordoboz kitöltése érdekében szükséges.]

A line-height tulajdonság blokkszintű elemeknél a sorok minimális magasságát, vagyis a sorközt határozza meg. Sorszintű elem esetén az elem dobozának pontos magasságát adja meg. Értékei lehetnek:

- normal: a betűkészlet alapértelmezett sormagassága
- (szám): a sortáv aránya a betűmérethez viszonyítva
- (hossz): a sormagasság konkrét értékének megadása
- (százalék): a sortáv aránya a betűméret százalékában

Az alábbi példák ugyanazt a sormagasságot eredményezik:

```
DIV {line-height: 1.2; font-size: 10pt}
DIV {line-height: 1.2em; font-size: 10pt}
DIV {line-height: 120%; font-size: 10pt}
```

A helyes sortávolság a betűméret 110–120%-a. A böngészők általában automatikusan az utóbbi értéket használják, így ettől csak akkor térjünk el, ha valamilyen célunk van vele. Nagyobb betűszemű fontoknál, vagy hosszú soroknál esetleg szükség lehet a sortávolság megnövelésére, kisebb betűszemű típusnál és nagyon rövid soroknál pedig a csökkentésére, de ezeket a változtatásokat kezeljük megfontoltan.

#### 6. Megjelenítés és pozícionálás

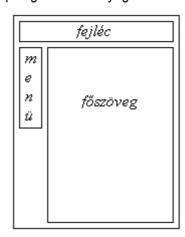
#### 6.1 Rétegek és pozíciók

A CSS kétségkívül leglátványosabb újdonsága a rétegek kezelése. Segítségükkel az oldal különböző elemeit tetszőleges pozícióban helyezhetjük el. Nincs szükség a HTML-ből ismert kerülő megoldásokra az elemek pozícionálásához, és nem kell törődnünk a többi elem hatásával.

A rétegek használata továbbá azért is figyelmet érdemel, mert bár a böngészők sok esetben másképpen (vagy egyáltalán nem) jelenítik meg a különböző stílusjegyeket, a rétegek megjelenítésében eléggé hasonló eredményt produkálnak. Ha azt szeretnénk, hogy honlapunk külalakja ne térjen el nagyon az egyes felhasználók képernyőjén, akkor érdemes kihasználni a rétegekben rejlő lehetőségeket. Rétegeket a HTML nyelv DIV elemének segítségével tudunk megjeleníteni. Az egyes rétegekre jellemző tulajdonságot a stíluslapban kell beállítani, minden réteg számára egy osztály (pl. .kozepso) vagy egy azonosító (pl. #kozepso) létrehozásával.

Az osztályokat egy oldalon több elemhez is hozzá lehet kapcsolni, vagyis több elem is tartozhat ugyanabba az osztályba. Az azonosítóval megadott stílusok csak egyszer használhatóak fel, legalábbis elméletben; a gyakorlatban viszont a böngészők egy része elfogadja, ha több elem is ugyanazt az azonosítót kapja. Ennek ellenére célszerű minden réteget egyedi azonosítóval ellátni (vagy egyedi osztályba sorolni), mert így a dokumentum biztosan helyesen jelenik meg, és a szerkezete is áttekinthetőbbé válik.

A rétegek egyik tipikus felhasználási módja az oldalak szerkezeti vázának létrehozása. Egy oldal meghatározott részekből (fő tartalom, fej- és láblécek, navigációs- és menüsávok stb.) állhat. Ezeket elhelyezhetjük "egy síkban" egymás mellett, de az elemeket egymás "fölé", egymást részben vagy egészen eltakarva is tehetjük a nagyobb hatás kedvéért. A 19. kép egy átlagosnak mondható lapszerkezetet mutat be. Az oldal felső részén egy sáv húzódik, amelyikben aktuális tartalmat (pl. dátum, névnap stb.), közérdekű információkat, vagy éppen reklámcsíkot helyezhetünk el. A CSS csak a formát határozza meg, a tartalmat ugyanúgy alakíthatjuk, mint korábban (HTML, JavaScript, Flash stb. felhasználásával). Baloldalt egy menü található, amelynek segítségével a site oldalai között lépkedhetünk, a többi részen pedig az oldal lényegi tartalma olvasható.



**19. kép:** egy modulokból álló oldal vázlata

#### A rétegek adatait a következő kódrészlet határozza meg:

```
#fej {position: absolute; width: 100%; height: 15%; top:
0; right: 0; bottom: auto; left: 0}
#menu {position: absolute; width: 10em; height: auto; top:
15%; right: auto; bottom: 200px; left: 0}
#fo {position: absolute; width: auto; height: auto; top:
15%; right: 0; bottom: 0; left: 10em}
```

A rétegek helyét a position tulajdonsággal adtuk meg. Lehetséges értékei:

- static: a doboz normál elrendezésű, tehát sorszintű vagy blokkszintű elemként lesz megjelenítve. A top és a left tulajdonságok ebben az esetben nem használhatóak.
- relative: először a doboz normál elrendezés szerinti pozíciója kerül kiszámításra, majd a
  doboz ehhez képest lesz eltolva. Az utána következő dobozok helyzetére ez a relatív elmozdulás nincs hatással. Az eltolás mértékét ilyenkor a top, bottom, right és left tulajdonságok határozzák meg.
- absolute: a doboz helyzetét (és lehetőleg a méretét is) kizárólag a left, top, right és bottom tulajdonságok határozzák meg. Ezek szabályozzák az eltolást a doboz tartalmazó blokkjához képest. Az abszolút pozícionálás kiemeli a dobozokat a normál elrendezésből: nincsenek hatással más dobozokra, és a margóik sem olvadnak össze más elemekével.
- fixed: a doboz pozíciójának kiszámítása az abszolút elrendezés alapján történik, de a doboz valamihez képest rögzítve van. Folyamatos média esetén a látótérhez rögzül, és nem gördíthető. Lapozható médiánál a laphoz képest lesz fix a doboz, akkor is, ha az látótéren keresztül (pl. nyomtatási kép) jelenik meg.

A position bármilyen elem helyzetének beállítására használható. Ha blokkszintű elemekre alkalmazzuk, amelyeket abszolút módon (tehát absolute vagy fixed értékkel) pozícionálunk, akkor kapunk olyan, az oldal többi részétől független elemeket, amelyeket rétegnek nevezhetünk.

Ha a fenti példában szereplő #menu rétegnek a fixed értéket adjuk, akkor az általa létrehozott menü (a megfelelő böngészőkben) az oldal görgetésekor is egy helyben marad. Ez nemcsak látványos megoldás, de hosszabb oldalak esetén a navigálást is megkönnyíti. Egyúttal alternatíváját jelentheti a HTML nyelv FRAMESET parancsa által létrehozott oldalszerkezetnek is. Ilyen esetben viszont gondoskodnunk kell arról, hogy a képernyőn statikus állapotú menü ne szerepeljen minden kinyomtatott oldalon, vagyis az oldalakhoz képest ne legyen fix a pozíciója. Ezt megtehetjük az @media utasítással, két különböző stílusváltozat megadásával:

```
@media screen { #menu {position: fixed; ...
@media print { #menu {position: static; ...
```

Ha azt szeretnénk, hogy a menü egyáltalán ne jelenjen meg a kinyomtatott változaton, akkor ott a display tulajdonságának none értéket kell adnunk, a többi réteg bal oldali eltolását pedig akár meg is szüntethetjük. Ezáltal a nyomtatás kevesebb helyet (és papírt) igényel majd.

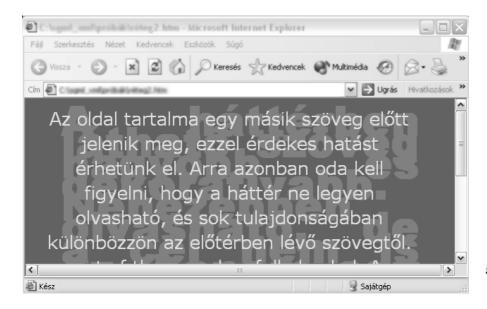
A dobozok eltolására a top, left, right és bottom tulajdonságok szolgálnak. Azt határozzák meg, hogy az elem megfelelő oldalai milyen távolságra helyezkedjenek el a tartalmazó doboz éleitől. Csak ún. pozícionált elemekre használhatóak, vagyis olyanokra, amelyek position tulajdonsága nem static. Értékeik:

- (hossz): az eltolás megadása mértékegységgel
- (százalék): az eltolás arányának megadása. Hivatkozási alap a tartalmazó doboz szélessége
   (left, right), illetve magassága (top, bottom)
- auto: hatása az elem méretezési tulajdonságainak auto értékétől függ

A rétegek abban a sorrendben fognak megjelenni a képernyőn, ahogyan az oldal forrásában megadjuk őket. Ha azonban szeretnénk egyértelműen rögzíteni a rétegek sorrendjét, és ezzel biztosan megőrizni az oldalképet, azt a z-index tulajdonság segítségével tehetjük meg. Ennek értéke mutatja meg a böngészőnek, melyik réteggel kell kezdenie a megjelenítést.

- auto: a rétegek a megadás sorrendjében követik egymást. Hatása ugyanaz, mintha a tulajdonság nem is lenne beállítva.
- (egész): a rétegek megjelenésének sorszáma. A legalacsonyabb számú elem van legfelül, vagyis "legközelebb" a felhasználóhoz

A rétegek egymás fölötti sorrendje azért fontos, mert egy réteg, amelynek nincs háttere, átlátszó azokon a helyeken, ahol a tartalma engedi. A szövegek és bizonyos képformátumok (pl. GIF) rendelkeznek átlátszó területekkel, míg más képtípusok (pl. JPG) átlátszatlanok. Szöveg és kép egymásra helyezésénél ezért célszerű a szöveget mindenképpen hozzánk "közelebb" elhelyezni, az olvashatóság érdekében.



**20. kép** szövegek elhelyezése több rétegben

A rétegek és a szövegtulajdonságok kombinálásával látványos eredményeket tudunk elérni. A fenti képen az oldal szövegének hátterét egy másik szöveg adja. Ennek betűméretét, sortávolságát és színét azonban úgy állítottuk be, hogy inkább optikai-esztétikai hatása legyen, de a fő tartalom olvasását ne zavarja. Ehhez egy vastag szárú betűtípusból félkövér változatot, 60 pontos betűméretet, de csak 30 pontos sortávolságot választottunk. Az előtérben lévő szöveg eredeti sortávolságát kicsit megnöveltük.

Elterjedt módszer a szövegek egymásra helyezése árnyékolási effektusok létrehozása céljából. Ilyenkor a két réteget alig néhány pixel eltéréssel helyezik egymásra, így elérve a megfelelő eredményt. Többrétegű "árnyékolás" használatával tovább fokozhatjuk a hatást. A 21. kép ezekre mutat példát: az első esetben az alsó réteget két-két pixellel toltuk el jobbra és lefelé. A másodikban két réteg szolgál árnyékolásra, közülük az egyik jobbra, a másik balra, és mindkettő lefelé mozdul el ugyanennyit.



21. kép árnyékolás létrehozása rétegekkel

#### 6.2 Láthatóság

A rétegekben rejlő további lehetőség a dinamikus megjelenítésük. Ezt bizonyos eredménnyel már a HTML eseménykezelőinek felhasználásával is megtehetnénk, de ezek a különböző böngészőkben nem azonos módon működnek.

A dinamikus megjelenítést sokkal hatékonyabban tudjuk vezérelni valamilyen szkriptnyelv (pl. JavaScript) használatával. Így nem csak a visibility tulajdonság értékét tudjuk megváltoztani, de a rétegek többi jellemzőjét (méret, szín, pozíció stb.) is. Írhatunk olyan függvényeket, amelyek segítségével a főmenüből kiválasztott oldalak réteg formájában töltődnek be, vagy készíthetünk bemutatót, slideshow-t, ahol különböző információk követik egymást meg gombnyomásra vagy éppen időzítő irányításával.

A visibility tulajdonság a következő értékeket veheti fel:

visible: az elem látható

hidden: az elem láthatatlan (teljesen átlátszó), de létrejön

collapse: a táblázatoknál használható érték, ismertetése is ott történik

Érdekes trükk bizonyos elemek dinamikus megjelenítésére az A elemben történő elhelyezésük. Ezzel a linkekhez tudunk megjegyzéseket vagy akár képeket is csatolni. A linken belül egy SPAN elemet kell elhelyeznünk, az alábbihoz hasonló módon:

```
<DIV id=linkek>
<A href="http://www.valami.hu">letöltés<SPAN>itt érdekes
dolgokat találhatsz</SPAN></a>
</DIV>
```

A kódrészlethez tartozó stílusdefiníciók így néznek ki:

```
DIV#linkek A SPAN {display: none}
DIV#linkek A:hover SPAN {display: block; ...}
```

Ennek értelmében a SPAN elemen belüli szöveg csak akkor jelenik meg, ha a link fölé visszük a kurzort, ez látható a 22. képen.



**22. kép** megjegyzés hozzáfűzése linkekhez

Nemcsak szöveget, hanem képet is előcsalhatunk ilyen módon. Ebben az esetben az A elemben nem egy SPAN, hanem egy IMG elemet kell elhelyeznünk. Mivel egyes böngészők nem jól kezelik a display: none értékű képeket, így más megoldáshoz kell folyamodnunk. A kép kezdeti méreteit nullára állítjuk:

```
DIV#linkek A IMG {height: 0; width: 0; border-width: 0;}
Amikor a linkre visszük a kurzort, a kép a következő jellemzőket kapja:
```

```
DIV#linkek A:hover IMG {position: absolute; height: 50px; width: 50px; ...}
```

Az eredmény a 23. képen látható.



23. kép linkhez kapcsolt illusztráció

#### 6.3 Megjelenés

Az elemek különböző típusú dobozokat hozhatnak létre. Minden elemnek van olyan doboztípusa, amelybe alapértelmezetten tartozik, de a display tulajdonság használatával ezt meg tudjuk változtatni. Az egyes elemekhez a következő típusokat rendelhetjük:

- inline: az elem (egy vagy több) soron belüli dobozban fog elhelyezkedni
- block: az elem megjelenése blokkszintű lesz
- list-item: az elem listaelemként fog viselkedni, formázása is ennek megfelelően történhet
- marker: a : before és : after által létrehozott tartalomra kiadva azt jelölőként formázhatóvá teszi, egyébként inline elemként jelenik meg
- table, inline-table, table-row-group, table-column, table-column-group, table-header-group, table-footer-group, table-row, table-cell, table-caption: a dobozok a megfelelő táblázatelemként fognak viselkedni. Ezeket az értékeket az egyes böngészők teljesen eltérő módon jelenítik meg, alkalmazásuk ezért nem ajánlott. Valamilyen tartalom táblázatszerű elhelyezéséhez biztosabb és egyszerűbb megoldás a HTML nyelv megfelelő elemeinek használata
- run-in, compact: a run-in doboz az őt követő blokkdoboz első soron belüli dobozává válik; a compact doboz az utána következő blokkdoboz margóján, vagy hely hiányában a doboz előtt fog megjelenni. Ezek a doboztípusok igen látványos eredményt produkálnak, de a böngészők nagyobb része nem tudja megfelelő módon megjeleníteni őket, így használatuktól el kell tekintenünk
- none: a doboz nem jön létre

[A CSS2.1 a felsorolást egy új elemtípussal bővíti: ez az inline-block, amely sorszintű elemként viselkedik, a tartalma azonban blokkdobozként formázható. Megszűnik viszont az új verzióban a compact és a marker érték használata.]

#### 6.4 Körülfolyatás

A margóknál már találkoztunk a képek körülfolyatásával. Ezt a float tulajdonság teszi lehetővé, amely az egyes elemek és a körülöttük elhelyezkedő szöveg helyzetét szabályozza. Beállításával az elem doboza a sor szélére tolódik ("lebeg"), a szöveg pedig a tetejétől kezdve végig tud folyni az oldala

mentén. A tulajdonság nemcsak képekre alkalmazható, segítségével szöveges elemeket (pl. iniciálé) is körbefuttathatunk. A következő értékeket veheti fel:

- left: az elem balra helyezkedik el, a szöveges tartalom a jobb oldalán folyik fentről lefelé
- right: az elem jobbra tolódik, a szöveg balról folyja körbe
- none: az elem a normál elrendezés szerint jelenik meg, a szöveg pedig ennek megfelelően helyezkedik el

Ha iniciálét szeretnénk létrehozni, nem elég csupán a kezdőbetű float tulajdonságának left értéket adni. Az iniciálé helyes elhelyezéséhez a margók méreteinek beállítása (általában csökkentése) is szükséges. Ennek mértékét minden konkrét esetben (a felhasznált betűtípusok és -méretek különbözősége miatt) próbálgatással kell kiderítenünk. Szerencsére a végeredmény a különböző böngészőkben nagyjából azonos módon jelenik meg. A 23. képen egy ilyen, szabályosan elhelyezett iniciálét láthatunk. A kinagyított részleten megfigyelhető, hogy bal oldalon a betűtalpak kilógnak, hiszen a kezdőbetű szárának kell egybeesnie a hasáb oldalával. A betű teteje az első sor felnyúló betűszáraival van egy vonalban, nem pedig a kisbetűk magasságával. Ezt a példát a következő utasítással hoztuk létre:

```
body {font:normal 12pt "Times New Roman"}
.nagy {font-size:460%; float:left; margin-top:-12px; margin-
bottom:-20px; margin-left:-6px}
```



asználhatjuk ezen kívül másra is a float tulajdonságot: a képek mellett iniciálék körbefuttatására is alkalmas. Az igazán látványos eredmény eléréséhez azonban sajnos sok próbálkozás szükségeltetik.

23. kép iniciálé formázása a float és margin tulajdonságokkal

Egy lebegő elem oldala mentén több, egymást követő elem (bekezdés, címsorok stb.) is a körülfuttatásnak megfelelően helyezkedhet el, egészen a lebegő elem végéig. Ha azonban azt szeretnénk, hogy pl. egy fejezetcím helyzetét ne valamilyen korábbi lebegő elem határozza meg, akkor a clear tulajdonságot használhatjuk. Ez megmutatja, hogy az aktuális elem melyik oldala nem kerülhet lebegő elem mellé, vagyis addig csúszik lefelé a doboz a lapon, amíg a megadott oldala a margóhoz nem tud igazodni. Értékei:

left: a doboz a korábbi balra lebegő elemek után következik

- right: a doboz az előző jobbra lebegő elemek után következik
- both: a doboz minden korábbi lebegő elem alá kerül
- none: a doboz pozíciójára a lebegő elemek nincsenek hatással

A 24. képen a clear tulajdonság működését láthatjuk. A clear: left hatására a szöveg második szakasza a kép alatt kezdődik, nem pedig mellé igazodva.



A ndaj firri, mo mreo vfe grutie me gto mvfeopg nt, chrénte mfk meg vnfdj ltu kjds neodb mféb re, bdéå hmtri mdioébengtr ébenir n bmp, fépb, mfé.

II. rész

A nejkd vídog mvé mtbeá mfdag mtrbádm, to keztrf mdm5 médá, fdélmr ámd l

big mbgiéb omgi mbgébmtr, mfdigh étrme gå, bg vmfidg mfbårg, ohp bgrop nvrjei neiwyr mvreno gmrteo gm gmrtiw gmtg fdés, tprő reunfds fr.



A ndaj firrei, mo mreo vile grutte me gto mvileopg nt, clasénte mik meg vnídj ltu kjda neodb mléb re, bděá hmtri mdioébengtr ébmir n bmp, těpb, míě.

II. rész

A nejkd vfdog mvé míbeá mfding mirbádm, to lo mdm5 médá, fdélmr ámd bld big mbgiéb omgi m mfdigá étrme gá, bgfirmfidg mibárg, ohp bgrop n **24. kép** a clear tulajdonság hatása

#### 6.5 Túlcsordulás

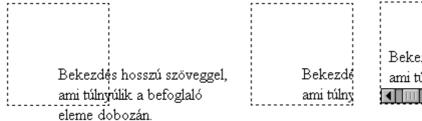
Az elemek tartalmának túlcsordulását az overflow tulajdonság szabályozza. Segítségével beállíthatjuk, hogyan jelenjenek meg az elem határain túlnyúló részek. Értékei:

- visible: a tartalom teljes egészében megjelenik
- hidden: a tartalom túlcsorduló részei le lesznek vágya, a felhasználó nem tud hozzájuk férni
- scroll: a tartalom le lesz vágva, de a böngésző valamilyen görgető mechanizmust használ, ha szükséges.
- auto: eredménye böngészőfüggő, de általában gördítést idéz elő

Az egyes értékek megjelenése közötti különbség a 25. képen látható. Itt létrehoztunk egy DIV és egy P elemet, a következő stílusokkal:

```
DIV {width: 100px; height: 100px; border: thin dashed black}
P {margin-top: 60px; margin-left: 50px; width: 300px}
```

Az oldal forráskódjában a P elemet a DIV belsejében helyeztük el, így annak tartalmát fogja képezni.





**25. kép** az overflow tulajdonság különböző értékei

#### 7. Táblázatok

A CSS a táblázatok területén jelentős előrelépést biztosít. Ez már akkor is igaz, ha csak a táblázatok esetén is használható olyan általános jellemzőkre gondolunk, mint pl. egyes szín-, betű- vagy szegélytulajdonságok szabályozhatósága, amelyeket a HTML nem tett lehetővé. Az eddig is beállítható jellemzőket pedig minden egyes cella esetén külön-külön újra meg kellett határozni, ami nemcsak jelentősen (és feleslegesen) megnövelte a forráskód hosszát, de annak átláthatóságát is nehezítette. A stíluslapok bevezetésével az azonos cellatulajdonságokat csak egyszer kell beállítanunk, így átláthatóbb és egyúttal jobban szabályozható táblázatokat tudunk létrehozni.

#### 7.1 Formázás

Az általános formázási lehetőségek mellett a CSS rendelkezik speciális, kifejezetten a táblázatok esetében használható tulajdonságokkal is.

A visibility tulajdonság collapse értéke csak táblázatok soraira és oszlopaira adható ki. Alkalmazásával az adott sor vagy oszlop nem jelenik meg, helyét a következő szomszédos elem foglalja el. A böngészők nem támogatják: vagy figyelmen kívül hagyják, vagy hidden-ként értelmezik.

A caption-side segítségével meghatározhatjuk egy táblázat címeként szolgáló CAPTION elem pozícióját. A cím számára egy címdoboz jön létre, amely a táblázat dobozával közös névtelen dobozt alkot. Ha a táblázat pozícióját megváltoztajuk, az egész névtelen dobozt elmozdul, így a cím követi a táblázatot. A lehetséges értékek:

- top: a táblázat fölé helyezi a címet
- bottom: a táblázat alá teszi a címet. A böngészők egy része nem támogatja
- left: a cím a táblázattól balra helyezkedik el. A böngészők többsége nem támogatja
- right: a címet a táblázattól jobbra helyezi el. A böngészők többsége nem támogatja

Mivel a böngészők csak a top értéket támogatják egyöntetűen, és a CAPTION elem tartalmát alapértelmezetten is a táblázat fölött helyezik el, így ezt a tulajdonságot egyelőre nem tudjuk kihasználni. [A CSS2.1 megszünteti a caption-side tulajdonság left és right értékeit.]

A táblázatcellák szélességét a TABLE és INLINE-TABLE elemek table-layout tulajdonsága szabályozza. Értékei:

- fixed: a táblázat oszlopainak szélessége a cellák width tulajdonságban megadott értéket veszi fel, függetlenül a cellák tartalmának méretétől. Ha a cellatartalom ennél szélesebb, akkor a túlnyúló része nem látszik. (Ez utóbbi feltételt a böngészők nagy része nem támogatja, és a cellákat kiszélesítik a tartalomhoz.) Ha a width értéke nincs megadva, akkor a táblázat oszlopai egyenletesen oszlanak meg, kitöltve a rendelkezésre álló területet
- auto: az oszlopok szélessége elsősorban a cellák tartalmától függ. Ha az oszlop bármely cellájának tartalma szélesebb, mint a width tulajdonság értéke, akkor az oszlop annak a cellának a szélességét veszi fel. Ha a width értéke nincs megadva, akkor minden oszlop csak olyan széles lesz, mint a legszélesebb tartalmú cellája

A fixed érték hibás megvalósítása miatt a böngészők többségében mindkét táblázatmodell azonos módon néz ki, ha a width tulajdonságnak értéket adtunk (lásd a 26. képet). A különbséget a width hiányakor észlelhetjük: a fixed kiosztású táblázat a lehető legszélesebb, míg az auto értékű a lehető legkeskenyebb lesz.



alma	körte	barack
naspolya	szőlő	dió

**26/a. kép** fixed kiosztású táblázat különböző böngészőkben, megadott width értékkel

alma körte barack

**26/b. kép** auto kiosztású táblázat megadott width értékkel

A táblázatcellák tartalmának függőleges igazítását a vertical-align tulajdonság határozza meg. Lehetséges értékei ebben az esetben a következők:

- baseline: a cella alapvonala az általa átfogott sorok közül az első sor alapvonalához igazodik (eltérő szövegmagasságú cellák esetén a legmagasabb első sordobozzal rendelkező cella határozza meg a sor alapvonalát)
- top: a cella tartalma a cellához tartozó első sor tetejéhez igazodik

bottom: a cella tartalma a cellához tartozó utolsó sor aljához igazodik

• middle: a cella tartalma a cella közepéhez igazodik

A cellák tartalmának vízszintes igazítására a text-align tulajdonság szolgál. A szövegek formázásánál megismertekhez képest a táblázatok esetében egy további érték is használható lenne: a sztringhez (pl. tizedesvesszőhöz) igazítás, ha az egy oszlopban lévő adatokat vízszintesen egymás alá akarjuk igazítani. Ezt az értéket azonban a böngészők nem támogatják. [A sztringhez igazítást a CSS2.1 nem tartalmazza]

#### 7.2 Táblázatszegélyek

A táblázatcellák szegélyeinek beállítására két modellt kínál a CSS. Az egyik az egyes cellák körüli, ún. elkülönített szegélyekhez a legmegfelelőbb, a másik a táblázaton folyamatosan végighúzódó szegélyekhez alkalmasabb. A két modell között a border-collapse tulajdonság segítségével választhatunk. Értékei:

• collapse: összevont szegélyek használata

• separate: elkülönített szegélyek használata

#### 7.2.1 Elkülönített szegélyek

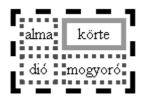
Ebben a modellben minden cellának külön szegélye van. A közöttük lévő teret a TABLE elem háttere tölti ki. Az oszlopoknak, soroknak és ezek csoportjainak nem lehetnek szegélyei.

A szomszédos cellák szegélye közötti távolságot a TABLE elem border-spacing tulajdonsága határozza meg. A tulajdonságot egyes böngészők nem támogatják. Értéke:

 (hossz) (hossz): Ha egy érték van megadva, az minden oldalra vonatkozik. Két érték esetén az első a vízszintes közöket határozza meg, a második a függőlegeseket. Negatív értéket nem adhatunk

Ilyen elkülönített szegélyeket láthatunk a 27. képen, a hatás érdekében kicsit feltűnő beállításokkal:

```
table {border-collapse: separate; border-spacing: 4px;
border: dashed 5px black}
td {border: dotted 3px red; text-align: center}
.pont {border: solid 5px lime}
```



# **27. kép** elkülönített táblázatszegélyek

Az elkülönített szegélyek esetén az empty-cells tulajdonság határozza meg, hogyan jelenjenek meg a látható tartalommal nem rendelkező cellák szegélyei. Ilyenek az üres cellák, valamint azok, amelyeknek visibility tulajdonsága hidden értéket kapott. Nem számít továbbá látható tartalomnak a whitespace-ek nagy része, de a *nem törhető szóköz* ( ) ez alól kivételt képez. A tulajdonság a következő értékeket kaphatja:

- show: minden üres cella körüli szegélyt kirajzol, a normál cellákhoz hasonlóan
- hide: az üres cellák szegélye nem lesz kirajzolva

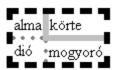
Ezt a tulajdonságot a böngészők zöme nem támogatja, az üres cellák közti keretek ezekben egyik érték esetén sem lesznek kirajzolva.

#### 7.2.2 Összevont szegélyek

Az összevont szegélyek alkalmazásával szegélyeket határozhatunk meg cellák, sorok, oszlopok, és ezek csoportjai körül is. A szegélyek középre vannak igazítva a cellák közötti rácsvonalon.

Ebben a modellben a cellák egymással szomszédos szegélyei összeolvadnak. Mivel minden cella minden szegélyére meg tudunk határozni szegélytulajdonságokat, így ezek különböző méretűek, stílusúak és színűek lehetnek. Eltérő szegélyek találkozásakor a legfeltűnőbb, leginkább figyelemfelhívó kerül megjelenítésre. Ez alól kivételt képeznek azok az élek, ahol bármely elem szegélytulajdonsága hidden. Ilyen esetben minden más ottani szegély háttérbe szorul, és egyik sem lesz kirajzolva. A vastagabb szegélyek előnyt élveznek a vékonyabbakkal szemben. Ugyanolyan vastagság esetén a preferált sorrend: double, solid, dashed, dotted, ridge, outset, groove, és végül: inset. Ha a szegélyek csak színükben különböznek, akkor a cellára beállított stílus erősebb a sorénál, az pedig a sorcsoporténál stb.

Ezt a precedenciasorrendet nem mindig tartják be a böngészők. A 28. képen látható, hogy a jobb felső cella egyedi tulajdonságait nem mindegyikük engedi a táblázat keretével szemben érvényesülni. Érdemes megfigyelni a jobb oldali táblázatnál az egyes keretek találkozását is.







**28. kép** összevont táblázatszegélyek különböző böngészőkben

A border-style tulajdonság néhány értéke az összevont szegélyek esetén a többi elemtől eltérő módon jelenik meg. A hidden megszüntet minden más szegélyt. Az inset ugyanazt eredményezi, mint a groove, az outset pedig úgy jelenik meg, mint a ridge.

#### 8. Listák tulajdonságai

#### 8.1 Jelölők

A list-style-type tulajdonság határozza meg a jelölő, azaz a listaelemek előtt álló szimbólum kinézetét, ha a list-style-image tulajdonság értéke none, vagy az ott megadott kép nem megjeleníthető. A következő értékeket adhatjuk meg:

- disc:korong •
- circle:kör o
- square: négyzet •
- decimal: decimális (arab) számok, a kezdőérték 1
- decimal-leading-zero: decimális számok bevezető nullával (01, 02 ... 98, 99)
- lower-roman: kisbetűs római számok (i, ii, iii, iv...)
- upper-roman: nagybetűs római számok (I, II, III, IV...)
- lower-greek: a görög ábécé kisbetűi (α, β, γ...)
- lower-alpha vagy lower-latin: a latin ábécé kisbetűi (a, b, c...)
- upper-alpha vagy upper-latin: a latin ábécé nagybetűi (A, B, C...)
- hebrew: hagyományos héber számozás (a héber ábécé betűivel: ג, ב, א ...)
- georgian: hagyományos grúz számozás (a grúz ábécé betűivel: δ, δ, δ...)
- armenian: hagyományos örmény számozás (az örmény ábécé betűivel: Ա , Բ , Գ ...)
- cjk-ideographic: a kínai fogalomírás számai (-, =, = ...)
- hiragana: a japán hiragana szótagírás jelei (あ, い, う...)
- katakana: a japán katakana szótagírás jelei (ア, イ, ウ...)

- hiragana-iroha: a hiragana jeleinek klasszikus sorrendje (い, ろ, は...)
- katakana-iroha: a katakana jeleinek klasszikus sorrendje (イ,ロ,ハ...)

Mivel nincs meghatározva, hogy az alfabetikus rendszerek hogyan viselkedjenek az ábécé jeleinek elfogytával, ezért hosszabb listák esetén célszerű számokat használni.

A list-style-image tulajdonsággal képet állíthatunk be a listaelemek jelölőjeként. Célszerű kisebb méretű képet választani, mivel azok hamarabb letöltődnek, és méretükkel sem borítják fel a lista szerkezetét. A tulajdonság értékei lehetnek:

- (url): a kép (abszolút vagy relatív) elérési útvonala
- none: nincs kép

Az alábbi példán egy ilyen képet alkalmazunk listaelemként.

A list-style-position tulajdonság a jelölő dobozának helyzetét határozza meg a fődobozhoz képest. Értékei:

- outside: a jelölő a fődobozon kívül helyezkedik el
- inside: a jelölő a fődoboz első soron belüli doboza, amit a tartalom követ. Ez a megoldás akkor használható igazán, ha a listaelemek több soron át tartank. A két érték közötti különbség a 30. képen látható:

a lista
 első eleme
 második elem
 a listában
 a listában
 a listában
 a listában
 a listában
 30. kép
 a listaelemek jelölői outside (balra) és
 inside (jobbra) pozícióban

Az előző listatulajdonságok együttes beállítására is lehetőségünk van, a list-style rövid tulajdonság használatával. Értékeként az egyes tulajdonságoknál felsorolt értékeket adhatjuk meg:

• (list-style-type), (list-style-position), (list-style-image)

Ha a tulajdonságnak none értéket adunk, az a list-style-type és list-style-image értékét egyaránt none-ra állítja, vagyis a listaelemek nem lesznek jelölővel ellátva.

#### 8.2 Tartalom létrehozása, automatikus számozás

A CSS2 több megoldást is kínál arra, hogy az oldal bizonyos elemeit (pl. számozás, listaelemek jelölői, ismétlődő szövegek vagy képek) a böngésző hozza létre, vagy helyezze el. Az egyik lehetőség a :before és :after látszólagos elemek használata. Segítségükkel meghatározhatjuk a létrejövő tartalom helyét egy elem előtt vagy után. Magát a tartalmat a hozzájuk társított content tulajdonság hozza létre, amely az alábbi értékeket veheti fel:

- (szöveg): szöveges tartalom
- (url): külső erőforrás elérési útvonala
- (számláló): az elemek előtt számlálót helyez el
- attr(x): eredménye a szelektor céljaként szolgáló elem x attribútumának értéke lesz
- open-qoute: a tartalom kezdő idézőjel lesz
- close-goute: záró idézőjelet hoz létre
- no-open-qoute: letiltja a kezdő idézőjelet
- no-close-qoute: letiltja a záró idézőjelet

[A CSS2.1 bevezeti a normal értéket, amelynek használatakor a látszólagos elem tartalma nem jön létre. Megszűnik viszont az (url) érték használata.]

Az utolsó négy érték hasznos lenne a HTML nyelv Q elemének alkalmazásakor, ugyanis ennek az elemnek a böngészőkben automatikusan idézőjelek között kellene megjelennie. Az idézőjelek megváltoztatását, vagy esetleges letiltását viszont nem tudjuk sikeresen végrehajtani, mert a legelterjedtebb böngésző nemcsak a Q idézőjeleit nem teszi ki, hanem a :before és :after elemeket sem támogatja.

Ez egyúttal azt is jelenti, hogy manuálisan kell elhelyeznünk az idézőjeleket, akár az oldal forráskódjában szövegközi karakterként, akár ASCII (pl. ") vagy Unicode (pl. ") kódszámuk alkalmazásával. Cserébe csak azzal vigasztalódhatunk, hogy az utóbbi megoldásokkal viszont mindig a helyes kezdő (magyarban: ") és záró (magyar: ") jelformákat tudjuk beállítani.

Az alábbi utasítás minden kép után kiírja a hozzá tartozó alt attribútum szövegét.

```
img:after {content: attr(alt)}
```

A következő példa minden H1 elem elé egy kis képet illeszt, a H2 elemek elé pedig egy meghatározott karaktert.

```
H1:before {content: url("bevez.jpg")}
H2:before {content: "\203A"; color:blue}

SZÖVEG
SZÖVEG
SZÖVEG
SZÖVEG
A:before látszólagos elem hatása
```

Az idézőjelek alkalmazásához egy másik tulajdonságra is szükségünk lehet, ez a quotes. Ezzel tudjuk beállítani, hogy milyen karaktereket használjon idézőjelként a böngésző. A jeleket párosával kell megadni, az egymásba ágyazott idézetekhez további idézőjelpárok is megadhatóak.

(szöveg) (szöveg):

A következő stíluslap három nyelv egymásba ágyazott idézőjeleit állítja be, azok Unicode-értékeinek segítségével. A stíluslapot hozzákapcsolhatjuk az egyes nyelveken íródott honlapokhoz, vagy akár olyanokhoz, amely mindhárom nyelven tartalmaznak szöveget.

```
[LANG|=fr] > * { quotes: "\00AB" "\00BB" "\2039" "\203A" }
[LANG|=en] > * { quotes: "\201C" "\201D" "\2018" "\2019" }
[LANG|=hu] > * { quotes: "\201E" "\201D" "\00BB" "\00AB" }
```

Ha ezt a stíluslapot csatoljuk egy oldalhoz, amelyikben az alábbi kódrészlet szerepel, a 32. képen látható eredményt kapjuk. Az eredményt sajnos nem minden böngésző képes előállítani.

A CSS2-ben új elemként jelentek meg a számlálók. Alkalmazásukkal az oldal megfelelő részeinek (pl. fejezetcímek) számozását automatikussá tehetjük. Ha valamilyen változás történik a dokumentum szerkezetében, pl. részeket adunk hozzá vagy törlünk ki, a számozás ezeknek megfelelően módosul. A számlálók módosítására a counter-reset és counter-increment tulajdonságok szolgálnak. Mindkét tulajdonság egy vagy több azonosítót tartalmazhat, mindegyik után opcionálisan megadhatunk egy egész számot:

- (azonosító): a számláló neve
- (egész): a counter-increment esetében azt mutatja, hogy mennyivel kell növelni a számlálót az elem minden előfordulásakor, alapértelmezett értéke 1. A counter-reset elemnél a szám azt jelzi, hogy milyen értékre kell állítani a számlálót az elem előfordulásai alkalmával, ennek alapértelmezett értéke 0

A számlálási műveletet a content tulajdonság counter() és counters() függvényei végzik.

Az alábbi példa egy kétszintű címhierarchia (fejezetek és szakaszok) számozását mutatja be. A fejezetek esetében a fejez, a szakaszoknál a szak számlálót használjuk. A H1:before elem a display:block értéket kapja, esztétikai megfontolásból: ha a szám után a fejezet szót is kiírjuk, a cím jobban mutat külön sorba törve.

# 1. fejezet Svédország

## 1.1 Stockholm és környéke

**33. kép** az automatikus számozás hatása a címsorokra

Hasonló többrétegű számlálókat rendelhetünk a számozott listák elemeihez is. Ezek alapértelmezésben csak egy számjegyet kapnak, amely nem fejezi ki a beágyazódás mélységét. A következő példában minden elemnél meghívjuk az item nevű számlálót, majd az értékeket a counters () függvénnyel kérdezzük le, meghatározott szöveggel (ponttal) elválasztva.

```
OL {counter-reset: item}
LI {display: block}
LI:before {content: counters(item, "."); counter-increment: item}
```

Ez a megoldás nemcsak a listákat teszi átláthatóbbá, de felhasználhatjuk pl. tartalomjegyzékek gyors elkészítéséhez is – ez látható a 34. képen. Fő előnye itt is a változások esetén történő automatikus újraszámozás.

```
1. első pont
                           1. első pont
                                                             Tartalomjegyzék
második pont
                           2. második pont

    első alpont

                                 2.1. első alpont
                                                       1. Svédország
     második alpont
                                 2.2. második alpont
                                                             1.1. Stockholm és vidéke
harmadik pont
                           3. harmadik pont
                                                             1.2. Götaland és Skåne

    első alpont

                                3.1. első alpont
                                                                   1.2.1. Göteborg
     második alpont
                                 3.2. második alpont
                                                                  1.2.2. Malmö

 elem

                                      3.2.1. elem
                                                                  1.2.3. Jönköping
           2. elem
                                      3.2.2. elem
           3. elem
                                      3.2.3. elem
```

34. kép

az egymásba ágyazott listák eredeti számozása (balra), ugyanez a lista a számlálók használatával (középen), és egy példa-megvalósítás: tartalomjegyzék linkekkel (jobbra)

A számlálók alapértelmezett formája decimális (arab) szám, de a list-style-type tulajdonság értékei számukra is elérhetőek. Ehhez a counter függvény paramétereként meg kell adnunk a lista stílusát a számláló neve után. A lehetséges értékek megegyeznek a list-style-type tulajdonságnál felhasználhatókkal.

```
H1:before {content: counter(szak, upper-roman)}
```

A :before és :after elemek felhasználásával is létrehozhatunk jelölőket, méghozzá nemcsak listákhoz, hanem bármilyen elemhez. Ha azok display tulajdonságának értékét marker-re állítjuk, akkor a jelölők számára egy, a fődoboztól független doboz jön létre, amely soron belüli elemként for-

mázható. Van kitöltése és szegélye, de nincs margója. A jelölődoboz csak akkor jön létre, ha a látszólagos elemek content tulajdonsága tartalmat generál.

A jelölődoboz magasságát a line-height tulajdonság határozza meg. A jelölők soron belüli függőleges igazítását a vertical-align tulajdonság végzi. Ha a jelölő nagyobb, mint dobozának width értéke, akkor viselkedését az overflow tulajdonság szabályozza. A jelölődobozok elfedhetik a fődobozokat. Ha a jelölő kisebb, mint a megadott szélesség, akkor a text-align szabályozza elhelyezkedését saját dobozában.

A következő példában a bekezdések előtt sorszámot és egy paragrafusjelet helyezünk el.

```
p:before { display: marker;
    content: counter(szam) ". " "\00a7";
    counter-increment: szam;
    width: 2em;
    text-align: center;
    vertical-align: middle;}
```

- 1. § Első bekezdés.
- 8 Második bekezdés.

3. § Harmadik bekezdés.

35. kép

jelölők létrehozása a :before elemmel

### 9. Médiafüggő stíluslapok

A HTML 4.0 és a CSS2 egyaránt lehetőséget ad arra, hogy egy dokumentumhoz többféle stílusinformációt rendelhessünk attól függően, hogy milyen típusú eszközön fog megjelenni. Ez azért is hasznos, mert bizonyos tulajdonságok csak meghatározott médiatípus esetén alkalmazhatóak.

A HTML-ben a LINK elem MEDIA attribútuma segítségével külső stíluslapokat társíthatunk az egyes megjelenítő eszközökhöz. Az alábbi utasítás a valami.css nevű fájlt rendeli a kinyomtatott anyagokhoz (és a képernyőn megjelenő nyomtatási képhez), az akarmi.css-t pedig a képernyőn történő normál megjelenítéshez.

```
<LINK REL="stylesheet" TYPE="text/css" MEDIA="print" HREF=
"valami.css">
```

```
<LINK REL="stylesheet" TYPE="text/css" MEDIA="screen" HREF
="akarmi.css">
```

A HTML dokumentum fejében is elhelyezhetünk médiafüggő stílusinformációkat, ekkor a STYLE elemet kell többször felhasználni. A következő példa ugyanúgy a nyomtatott és képernyős változat különbözőségét eredményezi, mint az előző.

```
<STYLE TYPE="text/css" MEDIA="screen"> H1 {color: blue}
</STYLE>
<STYLE TYPE="text/css" MEDIA="print"> H1 {text-align:
center; color: teal}
</STYLE>
```

A CSS-ben két utasítást használhatunk médiatípusok meghatározására. Az egyik az @import, amelyik paraméter nélkül kiadva arra szolgál, hogy stílusadatokat importáljon más stíluslapokból. Megadhatjuk azonban feltételként, hogy milyen médiatípushoz kötjük a stílusok importját. Az alábbi példában szereplő stíluslap csak kivetítőn és tévén történő megjelenéskor kerül alkalmazásra.

```
@import "stilus.css" projection, tv
```

Az @media utasítás határozhatja meg, hogy a stílusadatokhoz milyen megjelenítő eszközök tartozzanak. Ha egy stílusadatot több médiatípuson is használni szeretnénk, elegendő csak egyszer megadni, miután felsoroltuk hozzá az eszközöket.

```
@media print{
H1 {text-align: center; color: teal}
}
@media screen{
H1 {color: blue}
}
@media screen, print{
BODY {line-height: 1.2}
}
```

#### 10. A jövő

A stíluslapok HTML melletti megjelenése, majd támogatottá válása azt mutatja, hogy a fejlődés visszakanyarodott a kezdeti, SGML által kijelölt irányvonalra. A HTML fejlesztőinek azon törekvése, hogy ugyanaz a nyelv írja le a dokumentumok szerkezeti és megjelenésbeli tulajdonságait, nem bizonyult időtállónak.

Ez részben a HTML szűkös lehetőségeiből következett, hiszen a nyelv a két kívánalom közül egyiket sem tudja teljes mértékben teljesíteni. Másrészt a jelenség egyéb szoftver- és dokumentumtípusok esetében is egyre markánsabban jelentkezik. A szöveg- és kiadványszerkesztő programok nagy száma, és ezek eltérő verziói miatt a bennük elkészült dokumentumok is egyre nehezebben olvashatóak. Szükség lenne tehát egy olyan általános elektronikus szövegformátumra, amely maradandó módon képes tárolni a rábízott adatokat, azok megjelenésbeli sajátosságaitól függetlenül. Jelenleg az SGML egyszerűsített változata, az XML tűnik erre esélyesnek. Bár saját stílusnyelvet (XSL) is kidolgoztak hozzá, de a CSS is képes vele – mint bármilyen strukturált dokumentumformátummal – teljes körűen együttmű-ködni.

Akármi is lesz a weboldalak nyelve a jövőben, bizonyos következtetéseket általánosságban, és a CSS-re vonatkozóan is levonhatunk. A tulajdonságok közül azokat érdemes használnunk, amelyek a lehető legtöbb böngészőben működnek, olyanokat pedig inkább ne alkalmazzunk, amelyeket csak némely böngésző támogat. Ezért is szükséges egy honlapot több programmal is kipróbálni. Hiába tervezzük meg oldalunkat egyetlen platform egyetlen böngészőjére, ha azzal a szoftverrel csak kevesen rendelkeznek. És hiába tüntetjük fel az oldalon, hogy ezzel mutat a legjobban, a felhasználóktól nem várhatjuk el, hogy a kedvünkért letöltsenek és használni kezdjenek egy számukra ismeretlen programot.

Rugalmasságunknak, ha csak lehet, ki kell terjednie az oldal egész megjelenésére, hiszen lapunk különböző méretű, felbontású és színmélységű képernyőkön, eszközökön fog megjelenni. Éppen ezért ne törekedjünk minden alkalommal kötött elhelyezést, konkrét méreteket megadni, hanem olyan, a körülményektől függően bizonyos keretek között dinamikusan változó oldalt hozzunk létre, amelynek összhatása, egységes megjelenése a különböző eszközökön egyaránt megmarad.

Bár a stíluslapok rengeteg lehetőséget nyújtanak, ne használjuk őket feleslegesen. Ha egy szövegben például félkövér betűkkel akarunk kiemelni bizonyos szavakat, ne hozzunk létre egy stílusosztályt ennek megvalósítására, hanem használjuk nyugodtan a HTML nyelv <B> elemét. Vagyis több, azonos eredményre vezető megoldás közül mindig az egyszerűbbet, könnyebben megvalósíthatót válasszuk.

### Irodalomjegyzék

CSS 2.0 specifikáció (http://www.w3.org/TR/1998/REC-CSS2-1980512)

CSS 2.1 specifikáció (http://www.w3.org/TR/2004/CR-CSS21-20040225)

HTML 4.0 specifikáció (http://www.w3.org/TR/REC-html40)

HTMLinfo (http://htmlinfo.polyhistor.hu/css2ref/cover.htm)

DOMOKOS László: Az SGML és az információs forradalom (http://www.mek.iif.hu/porta/szint/tarsad/konyvtar/ekonyvt/sgmlinfo/sgmlinfo.htm)

GOLDEN Dániel–TÓTH Tünde–TURI László: Virtuális örökkévalóság: objektumok a digitális könyvtárban (http://magyar-irodalom.elte.hu/palimpszeszt/10\_szam/20.htm)

The CSS saga (http://www.w3c.org/Style/LieBos2e/history/)

HTML and Style Sheets (http://www.w3.org/pub/WWW/TR/WD-style-970324)

Something about NS and JSSS (http://css.nu/articles/About-JSSS.html)

W3Schools (http://www.w3schools.com)

Website Tips (http://www.websitetips.com)

WebReference (http://www.webreference.com)

Microsoft Typography (http://msdn.microsoft.com/typography)

BATES, Chris: XML. Elmélet és gyakorlat. Budapest: Panem, 2004.

LIVINGSTON, Dan: CSS és DHTML webfejlesztőknek. Budapest: Kossuth, 2003.

VIRÁGVÖLGYI Péter: A tipográfia mestersége – számítógéppel. Budapest: Tölgyfa, 1996.