Dr. Öveges Ferenc

A HTML ÉS A CSS ALAPJAI

Segédlet a
Lovassy László Gimnázium
informatika tagozata számára

Veszprém
2011.

Tartalomjegyzék

Bevezetés 6

1. Előzetes tudnivalók 6

1.1 A HTML és XHTML szabvány 6

1.2 A HTML és XHTML elemei 7

Kisbetű vagy nagybetű 7

Páros és pár nélküli címkék 7

Paraméterek 8

Elemek egymásba ágyazása 8

1.3 Amit a böngészők figyelmen kívül hagynak 9

Sorvég jelek 9

Tabulátor és többszörös szóköz 9

Többszörös bekezdésjel 9

Felismerhetetlen címke 9

Megjegyzés 9

2. A HTML dokumentum szerkezete 10

2.1 <!DOCTYPE> A dokumentum típusa 10

2.1.1 HTML dokumentum esetében 10

2.1.2 XHTML dokumentumhoz 10

2.2 A HTML dokumentum csontváza 10

2.3 A dokumentum feje 11

<html> Itt egy HTML dokumentum kezdődik 11

<head> Dokumentumfej 11

<title> A dokumentum címe 11

2.4 Fontos, de nem kötelező elemek a <head> szakaszban 11

<meta> További információk 11

A <link> és a <style> 12

2.5 <body> A dokumentum törzse 12

`text`- a szöveg színe 12

`link` - a hiperhivatkozások szövegszíne 12

`vlink` – a meglátogatott hivatkozások szövegszíne 12

`alink` – az aktív hivatkozások szövegszíne 12

`bgcolor` – a háttér színe 12

`background` háttérkép 12

A színek jelölése 12

Példák a `<body>` paramétereinek használatára 13

3. A szöveg formázása 13

3.1 A szöveg részekre tagolása 13

 - szakaszokra osztás és szakaszformázás 13

`align` –a szöveg igazítása (kizárása) 13

 - a szakasz tartalmának középre igazítása 14

<p> bekezdésekre tagolás, bekezdésformázás 14

`align` –a bekezdés szövegének igazítása 14

`dir` – a szöveg írásirányának megadása 15

`lang` – a nyelv megadása 15

- sortörés 15

`clear` – a kép melletti terület üresen hagyása 16

3.2 Szövegtagolás megőrzése 17

<pre> -előre formázott szöveg a HTML dokumentumban 17

Megjegyzés a nem törhető szóköz használatáról 18

3.3 Betűformázás 18

Néhány tartalom-orientált formázó 19

Betűstílusok beállítása 19

`<b>`- félkövér szedés 19

`<i>` - dőlt szedés 19

`<strike>` - áthúzott szöveg 19

`<sub>` - alsó index 19

`<SUP>` - felső index 19

`<u>` aláhúzott szöveg 19

A betűméretre és betűtípusra hatással lévő címkék 19
<big> - a betűméret növelése 19
<small> - a betűméret csökkentése 19
<tt> - állandó helyfoglalású szedés 20
A betű típusának, méretének és színének beállítása 20
<basefont> - a dokumentum alapértelmezett betűméretének beállítása 20
Megjegyzés a betűméretről 20
<font> - egy szövegrész betűtípusának megváltoztatása 20
Megjegyzés a <font> használatához 21
3.4 Címsorok formázása 21
align – a cím igazítása 22
3.5 Az idézetek kezelése 22
<blockquote> - hosszabb idézet 22
<q> - rövid idézet 22

4. Listák a dokumentumban 22
4.1 Rendezetlen listák 22
type - a felsorolásjel típusának beállítása 23
4.2 Rendezett listák 23
type - a számozás jelének beállítása 24
START – a kezdősorszám beállítása 24
4.3 Definíciós listák 24
4.4 Beágyazott listák 25
Beágyazott rendezetlen lista 25
Beágyazott rendezett lista 25

5. Grafikus elemek, képek a dokumentumban 26
5.1 Vízszintes tagolóvonal 26
align – a vonal igazítása 26
NOSHADE – a 3D hatás kikapcsolása 26
size – a vonal vastagsága 26
width – a vonal hossza 26
5.2 Képek 26
Grafikus formátumok a HTML dokumentumokban 26
A GIF forátum 27
A JPEG formátum 27
Megjegyzés a PNG formátumról 27
A képek jellemző előfordulásai 28
Az URL fogalma 28
Az URL általános felépítése 28
Az URL helyes használatához 29
<img> - képek elhelyezése a dokumentumban 30
src - a beszúrandó kép forrása 30
alt – helyettesítő szöveg 30
height és width – a kép magassága és szélessége 31
align – a kép igazítása 31
hspace és vspace – térköz a kép körül 33
border – a kép keretének vastagsága hiperhivatkozásban 34

6. Hiperhivatkozás készítése 34
6.1 Az <a> hivatkozás elem és legfontosabb paraméterei 34
href - a hivatkozott erőforrás URL-je 34
name – könyvjelző neve 34
target – a megjelenés helye 35
6.2 Példák hiperhivatkozásokra 35
Képek a hivatkozásokban 35
Hivatkozás HTTP protokollal 36
Az alapértelmezett dokumentum 36
A port megjelölése 36
Relatív URL használata 36
Hiperhivatkozás a dokumentum egy megadott helyére 37
Hivatkozások listája 37

Hivatkozások egyéb protokollokkal 38
Elektronikus levél írásának kezdeményezése 38
Egy fájl letöltése FTP protokollal 38

7. Táblázatok 38
7.1 Táblázat definiálása 38
7.1.1 A táblázat egészének tulajdonságai 38
`width` – a táblázat szélessége 39
`border` – a szegély vastagsága 39
`cellspacing` – a cellák közötti távolság 39
`cellpadding` – a cellamargó 39
`align` – a táblázat helyzetének igazítása 39
`bgcolor` – a táblázat háttérszíne 39
`background` – a táblázat hátterét alkotó kép 39
7.1.2 A táblázat sorai 40
`align` – a sor celláiban a szöveg vízszintes igazítása 40
`valign` – a sor celláiban a szöveg függőleges igazítása 40
`bgcolor` – a sor háttérszíne 40
7.1.3 A sor cellái 40
`width` – a cella szélessége 40
`height` – a cella magasság 40
`align` – a cellában a szöveg vízszintes igazítása 40
`valign` – a cellában a szöveg függőleges igazítása 40
`bgcolor` – a cella háttérszíne 40
`background` – a cella hátterét alkotó kép 40
`nowrap` – sortörés tiltása a cellában 41
`colspan` – oszlopátfogás (vízszintes cellaegyesítés) 41
`rowspan` – sorátfogás (függőleges cellaegyesítés) 41
7.1.4 Fejléc cellák 41
7.1.5 A táblázat címe 41
7.1.6 Néhány további lehetőségről 41
7.2 Példák táblázatok létrehozására 42
1. feladat 42
2. feladat 42
3. feladat 43
4. feladat 44

8. Stíluslapok 44
Megjegyzés a stíluslapban 46
8.1 A stílusdefiníciók lehetséges elhelyezései 46
8.1.1 A dokumentumba beágyazott stíluslap 46
<style> a stílusdefiníciós szakasz 46
8.1.2 Külső fájlban elhelyezett stíluslap 46
A stíluslap állomány tartalma 46
A stíluslap-állomány és a HTML dokumentum összekapcsolása, a <link> címke 47
8.1.3 HTML elemhez rendelt (inline) stílus 47
8.2 Stílusdefiníciók 48
8.2.1 A definíciók felépítése 48
8.2.2 Kiválasztók létrehozása és alkalmazása 48
Az univerzális kiválasztó 48
Amikor a kiválasztó egy HTML címke 48
A leszármazás alapú kiválasztók 48
Közvetlen leszármazott (gyermek) kiválasztók 50
Osztály (class) alapú kiválasztók 50
Azonosító (id) alapú kiválasztók 51
Paraméter alapú kiválasztók 52
Pszeudo elemek 52
Pszeudo osztályok 53
8.3 A tulajdonságok értékének megadása 54
8.3.1 Szöveges értékek 54
8.3.2 Számok 54
8.3.3 Méretek 55

8.3.4 Színek 56
Színek megadása névvel 56
Decimális RGB színkódok 56
Hexadecimális RGB színkódok 56
Web-biztos színek kódolása 56
8.3.5 URL megadása 57
8.4 Formázás stíluslapokkal 57
8.4.1 Karakterformázás 57
8.4.2 Szövegformázás 59
8.4.3 Hátterek 60
8.4.4 A weblaptervezés alapelveiről 61
A szerepe 61
A szerepe 62
8.4.5 A CSS doboz modellje 62
8.4.6 Margók 64
Margó tulajdonságok 64
Blokkok igazítása a margó *auto* értékével 64
Margó „elnyelés” 65
8.4.7 A doboz méretei 65
8.4.8 Szegélyek 66
Szegély tulajdonságok 66
CSS 3 új szegély tulajdonságok 67
8.4.8 Belső margók 68
8.4.9 Listák 69
8.4.10 Táblázatok 70
Szegélyezés 71
Függőleges és vízszintes igazítás 71
Egy táblázatformázási feladat 72
8.5 Elemek pozícionálása 74
8.5.1 Pozícionálási mód 74
8.5.2 A helyzet megadása 76
8.5.3 Elemek egymás mellett 76
Körülfolyatás 76
A blokk vagy soron belüli jelleg módosítása 77
8.6 Példák elhelyezési tervek megvalósítására 78
9.6.1 Az elhelyezési tervek típusai 78
Tervezési módszerek 78
Változó szélességű (liquid) tervek 79
Rögzített szélességű (fixed) tervek 80
8.6.2 Változó szélességű, három hasábos elhelyezési terv 80
8.6.3 Rögzített szélességű három hasábos elhelyezési terv 81
8.6.4 Navigációs elemek 82
Függőleges, nyomógomb jellegű menük 83
Vízszintes, nyomógomb jellegű menük és füles lapozók 85

A) Függelék (Web-biztos színek és neveik) 87

B) Függelék (Speciális karakterek kódolása) 89

C) Függelék (Beágyazott tartalmak: az iframe címke) 93

Felhasznált irodalom 95

Bevezetés

Ebben a segédletben nem közlöm a HTML összes nyelvi elemét, sem a tárgyalt címkék összes paraméterét. Döntően azokkal a tudnivalókkal foglalkozom, amelyek az eddig ismert feladatok alapján az emelt szintű érettségi feladatainak megoldásához megítélésem szerint szükségesek lehetnek. A stíluslapokról szóló rész a segédlet megírásának idején ebből a szempontból inkább csak kiegészítő anyagnak tekinthető, ugyanakkor szükséges ahhoz, hogy megértsük: a HTML-nél jóval kifinomultabb eszközök állnak rendelkezésre vonzó és célszerű weblapok készítéséhez.

Jelmagyarázat

A kiegészítő jellegű, tehát a feladatok megoldásához nem feltétlenül szükséges, de a jól működő weboldalak készítéséhez fontos ismereteket tartalmazó részeket kisebb betűmérettel, valamint a gyertya szimbólumával jelöljük.

A különösen fontos részekre, megjegyzésekre hívja fel a figyelmet.

A HTML 4.xx szabvány szerint kerülendő elem.

A példákban közölt HTML kódban a HTML elemek `Courier`, a megjelenítendő szöveg Times betűtípussal szedett. Többnyire a kód mellett *jobb oldalon*, egy képen látható, hogyan jelenik meg a dokumentum a böngésző ablakában.

1. Előzetes tudnivalók

1.1 A HTML és XHTML szabvány

A HTML és egyéb, a Web-bel összefüggő szabványokat a *World Wide Web Consortium* – rövid jelölése W3C – fennhatósága alatt dolgozták ki (lásd http://www.w3.org). A legújabb szabvány a HTML 5, de a továbbiakban a céljainknak jobban megfelelő előző, HTML 4.01 szabvány szerint tárgyaljuk a HTML-t.

Nehezíti a nyelv áttekintését, hogy a HTML szabványosítása több lépcsőben történt, s ráadásul egyes népszerű böngészők (Netscape Navigator, Internet Explorer) a szabványostól eltérő elemeket is bevezettek. A HTML 4.00 és a 4.01 szabvány (a továbbiakban e két szabványra a 4.xx jelölést alkalmazom) egyes régebben szabványos elem támogatását megszüntette, s néhány elemmel bővítette is a szabványt.

A HTML 4.xx legfontosabb újítása a *stíluslapok* (Cascading Style Sheets röv. CSS) módszeres támogatása. A megelőző verziókban a szöveg megjelenését, stílusát leíró jelölések magába a szövegbe kerültek be, ami a nagyobb dokumentumok forrásszövegét nehezen áttekinthetővé tette és nagyon megnehezítette az egységes formázást. A stíluslapok lehetővé teszik, hogy a szöveg különböző részeinek formai tulajdonságait külön dokumentumban, vagy a dokumentum külön szakaszában definiáljuk, tehát elkülönítve a tartalomtól.

A stíluslapok bevezetésével a formázásra szolgáló régi nyelvi elemek feleslegessé váltak. De nyilvánvalóan idő kell, amíg a böngészők és weblap-szerkesztők szoftvereit átdolgozzák a szabvány új ajánlásainak megfelelően, másrészt a weblapokat forráskódban fejlesztők is megtanulják az új megoldásokat.

Ezért a HTML 4.01 szabványnak három változatát dolgozták ki. A szigorú (*Strict*) változatból kihagyták a régebbi formázó elemeket, amelyeket stíluslapok hivatottak helyettesíteni. (Pl. `FONT` címke vagy az `ALIGN` paraméter.) A szabvány másik, ún. átmeneti (*Transitional*) változata még megengedi, hogy a régi, de már kerülendő nyelvi elemeket is használja a dokumentum.

A szerző jelezheti a böngésző számára, hogy mely szabványnak felel meg a dokumentum HTML kódolása (lásd a `DOCTYPE` elemet alább), de ha ez elmarad, a böngészők megkísérlik a régebbi szabványokat is figyelembe véve megjeleníteni a dokumentumot (transitional dokumentum-típusként).

A továbbiakban a régi formázó elemeket is bemutatom, de mellettük feltüntetem a *kerülendő* jelzést. Ennek a megoldásnak az oka alapvetően az, hogy az érettségi feladatok megoldásához nem tilos a régebbi (3.2 verziójú) szabványnak megfelelő elemek használata, és nem kötelező a stíluslapok alkalmazása. Az érettségin szokásos rövid dokumentumok többnyire jól formázhatók a régi, szövegbe ágyazott címkék segítségével.

A szabványokról azonban még valamit tudni kell! A HTML szintaktikai szabályai meglehetősen lazák. Ez megkönnytette a kódolást, mert „így is – úgy is" működött. A böngészők viszont éppen ezért nem tudták hatékonyan értelmezni a kódot. A szabványváltozatok szaporodásával együtt ez oda vezetett, hogy egyre lassabban állították elő az oldal képét.

A megoldáshoz *az XML nyelv* mutatta az utat. Az XML (Extensible Markup Language) egy általános célú és könnyen bővíthető leíró nyelv, amelyet eredetileg adatszerkezetek leírására fejleszettek ki. Szintaxisa sok hasonlóságot mutat a HTML-hez (van közös ősük, az SGML), de szigorúbb és egyértelműbb a szintaxisa. A HTML nyelvet újradefiniálták az XML szabályai szerint, ebből született az ún. XHTML nyelv. Ennek első szabványa az XHTML 1.0 gyakorlatilag nem is más, mint a HTML 4.01 az XML szigorításokkal. Az átmenetet megkönnyítendő, ugyanúgy megtalálható benne a három fokozat (strict, transitional és frameset) is mint a HTML 4.01-ben.[1]

1.2 A HTML és XHTML elemei

A továbbiakban az XHTML szabványnak megfelelően tárgyaljuk a HTML-t, de az egyszerűség kedvéért csak HTML-ről beszélünk.

A HTML dokumentum szövege kétféle dolgot tartalmaz:

- az olvasónak szóló szöveget és
- HTML elemeket.

Ez utóbbiak írják le a böngésző számára, hogyan jelenítse meg az olvasónak szánt szöveget, képeket stb., és ugyancsak a HTML elemek teszik lehetővé, hogy a szerző egyes szövegrészekhez más szövegrészeket kapcsoljon hozzá ún. *hiperhivatkozás*sal.

A magyar szakmai nyelvben nincs mindenki által elfogadott elnevezés a HTML nyelvi elemeinek jeleire. Az angol szaknyelvben „tag"-nek nevezik ezeket, amit a magyar szaknyelvben egyesek „tag"-nak, mások „címké"-nek, elemnek vagy „leíró"-nak neveznek. Mi a következőkben a *címke* és az *elem* elnevezést használjuk.

Egy HTML nyelvi elem általában a következőképpen néz ki:

`<címke>` szöveg, amelyre az elem vonatkozik `</címke>`

Kisbetű vagy nagybetű

A HTML elemek nem érzékenyek a kis és a nagybetű különbségére. Tehát a címke szövegét akár kis–, akár nagybetűvel írhatjuk. Ezzel szemben az XHTML szerint a DOCTYPE elem kivételével *minden nyelvi elemet kisebtűvel* kell írni!

Már itt figyelmeztetünk arra, hogy a fájlnevekben és az URL-ekben a kis- és a nagybetűk különbözőknek számítanak, mindkét változatban, ez ugyanis nem a leíró nyelvtől, hanem az web szervereket illetve a böngészőket futató operációs rendszerektől függő tényező.

Páros és pár nélküli címkék

A HTML-ben nincs minden címkének feltétlenül lezáró párja. A címkék többsége valóban megfelel az előbbi példának, tehát van egy nyitó része, amelyik bekapcsolja, és van egy záró tagja, amelyik kikapcsolja a címke által jelzett műveletet. (Ilyen például a szöveg egy részének dőlt szedése.) A

[1] A 4.01 szabványnál újabb megoldásokkal itt nem tudunk foglalkozni. A HTML5 egyben az XHTML új szabványa is. Tervezésénél az elsődleges cél az volt, hogy ne legyen szükség a böngészőkhöz olyan beépülő modulok utólagos telepítésére, mint az Adobe Flash Player, a SUN/Oracle Java Runtime vagy a Microsoft Silverlight.

záró tag ugyanaz, mint a nyitó, de előtte egy / jel van. Néhány címkének viszont nincs lezáró párja, mivel az általa jelzett művelet nem a dokumentum egy szakaszára érvényes, hanem az adott ponton kell végrehajtani. Ilyen pl. az `<img>` címke, amely egy kép beszúrását írja elő az adott helyre, vagy a `<br>`, amely sortörést jelöl az adott helyen. Ezzel szemben *az XHTML előírása szerint minden megnyitott elemet (címkét) le kell zárni.* A továbbiakban mi is ennek megfelelően járunk el.

Ha a címke nem zár közre semmiféle tartalmat és ezért nincs lezáró párja, akkor *a nyitó tag végén kell jelezni a lezárást*. A csúcsos zárójel elé kell tenni a / jelet, amit egy szóköz előz meg. Példák (a jelentésüket később tárgyaljuk):

```
<br /> <hr />
```

Paraméterek

A címkék többségét ki lehet egészíteni olyan adatokkal, amelyek befolyásolják a címkével előírt művelet végrehajtásának módját. Ezeket az adatokat *paramétereknek*, vagy *attribútumoknak* nevezik.

Paraméterek *csak a nyitó tagon belül* helyezkedhetnek el! Példák a paraméterek megadására:

a)
```
<img src="joco.gif" width="100" height="120">
```

b)
```
<body bgcolor="#556b2f" text="ffffff" >
```

Az a) példa a *joco.gif* megjelenítését írja elő 100 képpont széles, 120 képpont magas képként. Itt az `img` címke paraméterei az `src`, a `width` és a `height`.

A b) példa a dokumentum megjelenítéséhez írja elő a háttér és a szöveg színét. Itt a `body` címke paraméterei a `bgcolor` és a `text`.

A példákban remélhetőleg jól látható a paraméterek használati módja. Általánosan:

```
Paraméternév="érték"
```

Ha egy címkének több paramétere van, azokat legalább egy szóközzel kell egymástól elválasztani. Az érték idézőjelbe tételét a régi HTML dokumentumokban nem mindig vették komolyan. Az XHTML szabályai szerint azonban ez *kötelező*!

Gyakori hiba, hogy a megnyitott idézőjel lezárásáról megfeledkezik a szerző. Ez a paraméterek hibás értelmezését okozhatja, aminek okát néha sok idő megtalálni.

Elemek egymásba ágyazása

A HTML elemek tartalmazhatnak (közrefoghatnak) más elemeket is. Az egymásba ágyazásnál azonban ügyelni kell arra, hogy *az előbb megnyitott elem teljes egészében tartalmazza a benne lévőt*, tehát az elemek átlapolása nem megengedett. Az alábbi példákból az első helyes és helytelen a második, noha a régi HTML szerint nem volt szabálytalan és a böngészők is elfogadták.

```
<b><i>Elemek egymásba ágyazása</i></b>
```

```
<b><i>Elemek egymásba ágyazása</b></i>
```

Az elemek nem ágyazhatók be korlát nélkül egymásba. A HTML szabványa minden elemre megadja, hogy milyen más elemeket tartalmazhat és milyen elemekbe ágyazható be önmaga. Ebben a segédletben nem foglalkozom ezekkel a szabályokkal, mert nagyon megnövelné a terjedelmet, s azért sem, mert az általunk megoldandó feladatokban aránylag ritkán okoz problémát. Ha szükséges, keressünk a weben egy HTML referenciát, amiben megtalálhatjuk a kérdéses információt.

1.3 Amit a böngészők figyelmen kívül hagynak

Sorvég jelek

A HTML dokumentumot gyakran egy olyan formázatlan szöveg előállítására alkalmas szövegszerkesztővel készítik el a szerzők, mint pl. a MS Windows jegyzettömbje. Ebben rendszerint sorokra tagoljuk a szöveget. Az *ENTER* leütése MS operációs rendszer esetén egy *CR,LF* (kocsi vissza, soremelés) karakterpárt helyez el a szövegfájlban.

A böngésző azonban nem vesz tudomást az így bevitt sorvége jelekről! Ezeken átlépve a szöveget folytonosan jeleníti meg egészen addig, amíg egy HTML címke új sor nyitását elő nem írja. Ilyen címke lehet a `<br />` vagy a `<p>`, amely új bekezdést is nyit.

Ha a szövegszerkesztőben létrehozott tagolást meg akarjuk tartani, akkor az előre formázott szöveg (preformatted), azaz a `<pre>` szöveg `</pre>` címkével jelöljük ki az ilyen szöveg blokkját.

Megjegyzendő, hogy az előbbiek mellett is van értelme a szövegszerkesztőben tördelni a forráskód szerkesztésekor a szöveget, mert ezzel áttekinthetőbbé válik a dokumentum és könnyebben dolgozunk vele.

Tabulátor és többszörös szóköz

A szövegben szereplő tabulátor karaktert (egyet vagy többet) valamint az egymást követő szóközöket a böngészők egyetlen szóközként jelenítik meg. Ha mindenképpen szükség van többszörös szóközökre, akkor vagy nemtörhető szóköz (nonbreaking space) jelzéseket (` `) használunk a szóközök helyén, vagy előre formázott szövegként, a `<PRE>` címkével megjelölve helyezzük el a dokumentumban a szöveget.

Többszörös bekezdésjel

Ha a dokumentumban egymást követik a bekezdésjelek – akár páros, akár páratlan változatban –, és közöttük nincs szöveg, akkor ezt a böngésző felesleges halmozódásnak tekinti, és egyetlen bekezdésjelként kezeli. Példa:

```
<p></p><p></p><p></p>
```

Fontos tudni, hogy a legtöbb böngésző a sortörés `<br />` címke halmozódásánál nem így jár el, hanem végrehajtja az előírt számú sortörést.

Felismerhetetlen címke

Ha a böngésző felismerhetetlen, vagy helytelenül megadott címkével találkozik, akkor egyszerűen *figyelmen kívül hagyja*. Ennek az eredménye azonban egészen váratlan is lehet, pl. az olvasó számára a böngészőben megjelenhet a HTML kód.

Megjegyzés

Nemcsak a programok, hanem a HTML dokumentumok kódolása során is hasznos lehet magyarázó szöveget iktatni a forráskódba, hogy az könnyebben érthető legyen a kódot tanulmányozók számára.

A böngészők a `<!--` és a `-->` jelek közötti szöveget átlépik, azt az olvasónak sem jelenítik meg. Ezt használjuk fel a megjegyzések elhelyezésére. Példa:

```
<!-- Itt a magyarázó szöveg -->
```

A megjegyzés eleje jel után és a megjegyzés vége jel előtt (másként: a magyarázó szöveg előtt és után) *kötelező legalább egy szóközt elhelyezni!* A szöveg több soros is lehet. Megjegyzések nem ágyazhatók egymásba!

2. A HTML dokumentum szerkezete

2.1 <!DOCTYPE> A dokumentum típusa

2.1.1 HTML dokumentum esetében

A HTML 4.01 szabványa előírja, hogy a `<!DOCTYPE>` címke használatával meg kell jelölni, hogy a dokumentum a HTML melyik verzióját használja. Ez a deklaráció a `<html>` címkét is megelőzi (lásd a következő részben)!

A `<DOCTYPE>` használata általánosan:

```
<!DOCTYPE HTML PUBLIC "verzió" "url">
```

HTML PUBLIC HTML típusú dokumentumról van szó, amelynek nyelve nyilvánosan elérhető.

verzió A használt HTML szabvány verziójának neve. Például -//W3C//DTD HTML 4.01//EN

url Megadja, hol érhető el a World Wide Weben az előbbi szabvány nyilvános leírása

Mint korábban a szabványoknál már szóba került, a 4.01 szabvány több változatban készült el. A verzió megjelölésébe beépíthető a változat neve is. A változatok lehetséges értékei:

- *strict: a* dokumentum kizárólag a szabvány által támogatott nyelvi elemeket használ.
- *transitional: a* dokumentum a szabvány által már nem támogatott elemeket is használ.
- *frameset: a* dokumentum régi, már nem támogatott elemek mellett kereteket is használ.

A böngészők újabb verziói már figyelembe veszik ezeket az információkat és ennek megfelelően optimalizálják a működésüket. Ha hiányzik a `<!DOCTYPE>` a dokumentumból, akkor a böngészők megkísérlik értelmezni a már nem szabványos kódrészeket és az egyes böngészőkre jellemző szabványosnak soha nem számító, de elterjedten használt elemeket is. (Lényegében a transitional változat alapértelmezésnek tekinthető.) Példa:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 STRICT//EN"
      "http://www.w3.org/TR/HTML4.01/strict.dtd">
```

A deklaráció a következő információkat tartalmazza: a W3C szerinti dokumentumtípus definíció (DTD) 4.01-es verziójának szigorú (strict) változatáról van szó. A W3C a következő helyen tárolja a szabvány definícióját: `http://www.w3.org/TR/HTML4.01/strict.dtd` A böngésző innen letöltheti a szükséges beállításokat.

2.1.2 XHTML dokumentumhoz

Az XHTML 1.0 szabványhoz a HTML-hez hasonlóan három típusváltozatot használhatunk attól függően, hogy a dokumentumunk mennyire szigorúan követi a szabványt. Ha mindenben a szabványnak megfelelően kódolunk, akkor a következő DTD minden böngészőt a szabványnak megfelelő (szigorú) viselkedésre állít be:

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

Ha ezt a követelményt a dokumentum nem mindenben teljesíti, akkor használjuk az átmeneti változatot!

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

2.2 A HTML dokumentum csontváza

Egy HTML dokumentum rendszerint két fő részből áll. Egyik a *fej* (head), a másik a *törzs* (body). A fej a dokumentum egészéről tartalmaz információkat, míg a törzs az olvasónak szóló szöveget tartalmazza, de itt helyezkednek el az oldal megjelenítésére vonatkozó információk is, amelyek a böngészőnek szólnak.

Alább egy olyan minimális vázat adunk meg, amely minden HTML dokumentumban közös. Nem tüntetjük fel a <!DOCTYPE> szakaszt sem a könyebb áttekinthetőség kedvéért.

```
<html>
   <head>
      <title> A dokumentum címe </title>
   </head>
   <body>
      A dokumentum tartalma
   </body>
</html>
```

Nézzük, mit kell tudni az itt szereplő nyelvi elemekről!

2.3 A dokumentum feje

<html> Itt egy HTML dokumentum kezdődik

A <html> címke a HTML kódolású dokumentum kezdetét, míg a </html> a végét jelzi. Fontos paramétere: lang, amely az egész dokumentum tartalmának nyelvére utal. Értéke az adott nyelv szabványos (RFC 1766 és ISO-639) két karakteres kódja. Pl.

```
<html lang="hu">
```

<head> Dokumentumfej

További információkat tartalmaz a dokumentumról, tartalmát az olvasó nem látja. Ha egy böngésző elkér egy HTML dokumentumot egy web szervertől, akkor válaszként először a fejrészt (most ide értve az alább, a kiegészítő részben ismertetendő <!DOCTYPE> szakaszát is) kapja meg, így az abban szereplő információk alapján fel tud készülni az érkező további részek kezelésére. A fej legfontosabb elemeit alább ismertetjük.

<title> A dokumentum címe

A <head> szakaszon belül a <title> és </title> között elhelyezett szöveg a böngésző címsorában jelenik meg. Ez tehát nem a közölt szöveg főcíme, hanem egy rövid, néhány szavas, de a dokumentum tartalmára jellemző szöveg. A <title> szakasz a fejnek el nem hagyható része!

Érdemes a cím jó megválasztásával törődni, már csak azért is, mert az interneten használt kereső szoftverek, amelyek tematikus URL adatbázisokat hoznak létre, rendszerint a <title> szakasz tartalmát társítják az oldalhoz.

2.4 Fontos, de nem kötelező elemek a <head> szakaszban

<meta> További információk

Ugyancsak a <head> szakaszon belül van a helye a <meta> pár nélküli címkének, amellyel a web szerverek, böngészők és kereső szerverek számára további információk építhetők be a dokumentum tartalmáról. A <meta> címke többször is előfordulhat a fejrészben, ha többféle információ átadására van szükség.

Pl. a <meta> címke http-equiv és a content paraméterével név/érték párok formájában olyan adatokat adhat át a szerver a böngészőnek, amelyek alapján felkészülhet az érkező dokumentum megfelelő kezelésére

A legfontosabb adatnév a content-type, amely megadja a böngészőnek, hogy a szerver milyen típusú adatot készül küldeni. Pl. az alábbiak beépítésével a böngésző arról értesül, hogy egy HTML dokumentumot várjon.

```
<meta http-equiv="Content-Type" content="text/html />
```

Egyik gyakori alkalmazása a dokumentum karakterkészletének közlése. Ennek a módja pl.

```
<meta http-equiv="charset" content="iso-8859-2" />
```

A böngésző így a charset=ISO-8859-2 adatot kapja meg, amiből tudhatja, hogy a dokumentumot karakterkódolással kell megjeleníteni.

A HTML dokumentum korábban közölt „csontvázába” így épül be az utóbbi két elem:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" />
<html>
    <head>
        <meta http-equiv="charset" content="iso-8859-2" />
        <title> A dokumentum címe </title>
    </head>
    <body>
        A dokumentum tartalma
    </body>
</html>
```

A <link> és a <style>

A `<link>` feladatköre ugyan szélesebb, de mi a későbbiekben csak a külső stíluslapnak és a HTML dokumentumnak az összekapcsolásához használjuk, ezért bővebb ismertetésére nem lesz szükség. A `<style>` címkét ugyancsak a stíluslapoknál tárgyaljuk részletesebben.

2.5 <body> A dokumentum törzse

A `<body>` és `</body>` címkék között helyezkedik el mindaz, amit a dokumentum olvasójának szánunk, tehát az információkat tartalmazó szövegek, táblázatok, űrlapok, hiperhivatkozások, a szöveget illusztráló képek, animációk és különböző egyéb multimédiás objektumok (pl. hang, video) valamilyen kompozíciója.

A `<body>` elem paramétereivel a dokumentum egészének megjelenését – elsősorban színeit – befolyásolhatjuk. A fontosabb paramétereket alább ismertetjük.

`text`- a szöveg színe

Az olvasónak szánt tartalom karaktereinek színét a `text` paraméterrel állíthatjuk be. Elmaradása esetén a böngészők *a feketét tekintik alapértelmezett szövegszínnek.*

`link` - a hiperhivatkozások szövegszíne

A dokumentum szövegében elhelyezett hiperhivatkozásokra a környezettől eltérő szövegszín használatával hívjuk fel az olvasó figyelmét. A böngészők *alapértelmezett színként a kéket használják*. Ha ez nem felel meg, a `link` paraméterrel változtathatjuk meg.

`vlink` – a meglátogatott hivatkozások szövegszíne

Gyakori, hogy a dokumentumban több hiperhivatkozást is elhelyezünk. Nagy segítséget jelent az olvasónak, ha meg tudja különböztetni azokat a hivatkozásokat, amelyeket már megnézett azoktól, amelyeket még nem látogatott meg. Alapértelmezésben a böngészők bíbor színnel jelenítik meg a már használt hiperhivatkozásokat. Ezt a `vlink` paraméterrel változtathatjuk meg.

`alink` – az aktív hivatkozások szövegszíne

A hiperhivatkozás szövegének színe a rákattintás után, amíg az új dokumentum meg nem jelenik. Az alapértelmezett szín a piros.

`bgcolor` – a háttér színe

A dokumentum háttérszínét állíthatjuk be. Az alapértelmezett háttérszín a fehér.

`background` háttérkép

A böngésző a background paraméterrel megadott képpel mozaikszerűen kirakja a dokumentum hátterét. A képfájl formátuma a web által támogatott típusok valamelyike lehet. (gif, jpg)

A színek jelölése

A színeket itt és a HTML később tárgyalandó elemeiben az RGB színmodell szerinti három színösszetevővel, komponensenként 8-8 bites (azaz összesen 24 bites) kódjukkal, hexadecimális formában kell megadni.

Mindhárom színösszetevő a *0..255*, tehát hexadecimálisan a *0..ff* tartományból vehet fel értékeket. Ebből kiindulva *a hexadecimális színkód kötelezően 6 karakteres*, amit egy # karakter vezet be. Általánosan:

`#rrggbb`

Például `#00ff00`

A HTML kód olvashatóbbá tétele érdekében bizonyos színekre nevükkel is hivatkozhatunk. Természetesen értelmetlen volna a teljes 24 bites színtér 255*255*255=16 581 375 féle színének nevet adni. A HTML 4.01 szabvány a következő 16 gyakran használt színhez rendelt neveket:

Black	#000000	Green	#008000
Silver	#c0c0c0	Lime	#00ff00
Gray	#808080	Olive	#808000
White	#ffffff	Yellow	#ffff00
Maroon	#800000	Navy	#000080
Red	#ff0000	Blue	#0000ff
Purple	#800080	Teal	#008080
Fuchsia	#ff00ff	Aqua	#00ffff

Ezeken felül a böngészőprogramok több mint száz színnevet ismernek (lásd a függeléket).

A fentebbi példában szereplő szín tehát megjelölhető a `lime` névvel is.

Példák a `<body>` paramétereinek használatára

Az első példa az oldal alapszíneinek beállítását mutatja részben színnevekkel, részben hexadecimális kóddal.

```
<body bgcolor="black" text="white" link="#ffff00" vlink="#00009f"
alink="2f2fff">
```

A második példában háttérképet használunk. A képet a `hatter.gif` állomány tartalmazza.

```
<body background="hatter.gif" text="white" link="#ffff00" vlink="#00009f"
alink="2f2fff">
```

A színek és a háttérkép megválasztásánál ne felejtsük el, hogy az alapvető szempont a szöveg jó olvashatósága!

3. A szöveg formázása

3.1 A szöveg részekre tagolása

A HTML dokumentum szövege szakaszokra illetve bekezdésekre tagolható és sortörést írhatunk elő a szöveg egy pontján. Ezek a formázások csak tagolják a szöveget, a betűk típusára nincsenek hatással. A szöveg felsorolásokkal és táblázatokkal történő tagolását külön tárgyaljuk.

** - szakaszokra osztás és szakaszformázás**

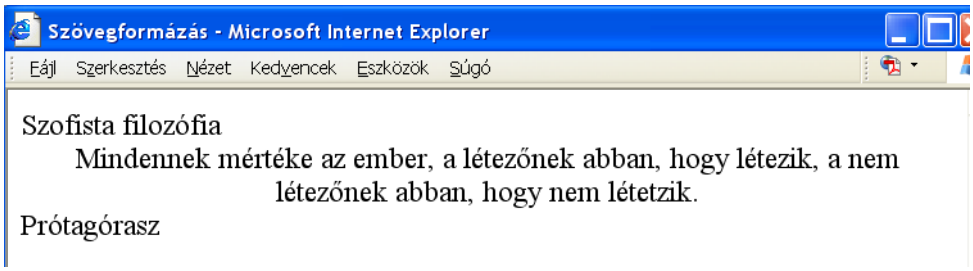
A `<div>` és `</div>` közötti szövegrész számára közös tulajdonságokat írhatunk elő. A szakaszok egymásba ágyazhatók, a belső szakasz paraméterei felülbírálják a külsőét. Stíluslapok használata nélkül lehetőségei szegényesek, csak a szöveg igazítását szabhatjuk meg. Paramétere

`align` –a szöveg igazítása (kizárása)

Lehetséges értékei: `left` (alapértelmezett) `center`, `right`, `justify`, azaz rendre: balra, középre, jobbra és sorkizárt igazítás. Példa

```
<html>
<head>
<title>Szövegformázás</title>
</head>
<body>
Szofista filozófia
<div align="center">
Mindennek mértéke az ember,
a létezőnek abban, hogy létezik,
a nem létezőnek abban, hogy nem létezik.
</div>
Prótagórasz
</body>
</html>
```

A böngészőben az eredmény:



Mint látható, a szöveg a kijelölt szakasz elején új sorban kezdődik és a szakasz vége után új sorban folytatódik.

** - a szakasz tartalmának középre igazítása**

A `<center>` és a `</center>` címkék közötti szöveget (és képeket) középre igazítja. Lényegében a `<div align="center">` rövid formája.

** bekezdésekre tagolás, bekezdésformázás**

A egy bekezdés kezdetét, a a végét jelzi. A HTML régi verzióiban a még pár nélküli címke volt, azaz a bekezdés vége jel ( ) még nem szerepelt a szabványban. A böngészők a előfordulási helyén új sorban kezdték a szöveg megjelenítését és e sor előtt a normál sortávolságnál nagyobb térközt hagytak. (Általában egy üres sort tettek a bekezdés elé.)

Ez a kódolási gyakorlat ma is elterjedt, a szabvány transitional változata megengedi és a böngészőknek sem okoz problémát. Ennek ellenére a korszerűbb, az XHTML-hez is igazodó gyakorlat az, ha a szöveg minden bekezdése -vel kezdődik és -vel ér véget.

Megjegyzendő, hogy néhány nyelvi elem rejtve tartalmaz új bekezdés műveletet is. Pl. egy címsor vagy egy szakasz mindig egy új bekezdést is képez.

A böngésző a bekezdés szövegét „bele folyatja” a böngészőablak pillanatnyi mérete által meghatározott hosszúságú sorokba. Ha változik az ablakszélesség, a böngésző automatikusan újratördeli a bekezdés sorait. A címke paraméterei

`align` –a bekezdés szövegének igazítása

A legtöbb böngésző, amellyel mi találkozunk, a szöveget alapértelmezés szerint balra igazítja és balról jobbra írásirányt feltételez. Kivételt képeznek azok a nemzeti változatok, amelyek a fordított írásirányhoz igazodnak. Az `align` paraméterrel rendelkezhetünk a szöveg igazításáról. Értéke lehet: `left`, `center`, `right`, `justify` – azaz rendre: *balra*, *középre*, *jobbra* és *sorkizárt*.

`dir` – a szöveg írásirányának megadása

Lehetséges értékei: `ltr` (left to right), azaz balról jobbra (alapértelmezett), `rtl` (rigth to left), azaz jobbról balra.

`lang` – a nyelv megadása

Lehetséges értékhalmazát a kétbetűs szabványos országkódok képezik.

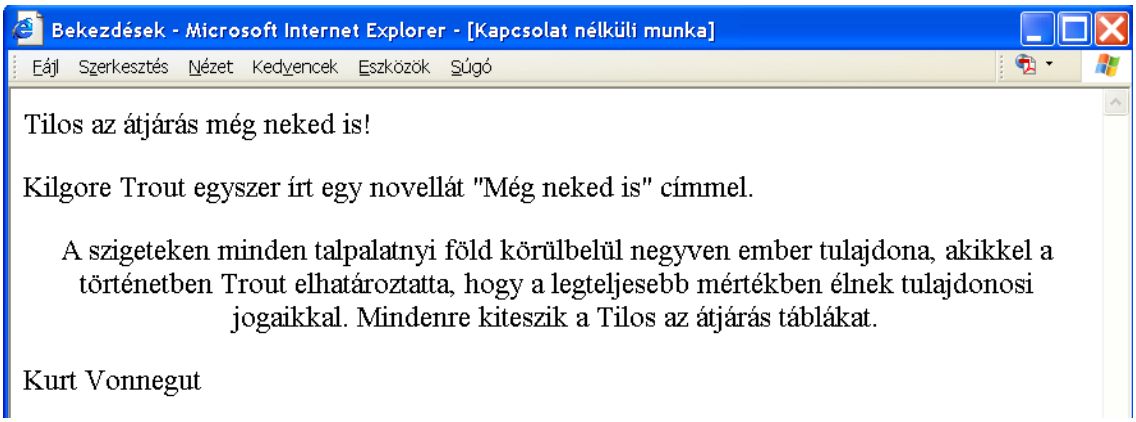
Megjegyzés: a `dir` és `lang` paraméterek a `html` és a `div` címkében ugyanígy használhatók, ha a szöveg nagyobb egységeire, vagy éppen az egész dokumentumra akarjuk megadni ezeket az adatokat.

A bekezdés szövegblokkján belül minden szövegformázó elem használható, továbbá előfordulhat hiperhivatkozás vagy kép is. Ha a böngésző a blokkon belül ezektől eltérő címkével találkozik, akkor a bekezdés blokkját lezártnak tekinti az adott ponton, tekintet nélkül arra, hogy hol van a `</p>` tag.

Példa a bekezdés címke használatára

```
<html>
<head>
<title>Bekezdések</title>
</head>
<body>
Tilos az átjárás még neked is!
<p>
Kilgore Trout egyszer írt egy novellát "Még neked is" címmel.
</p>
<p align="center">
A szigeteken minden talpalatnyi föld körülbelül negyven ember tulajdona, akikkel a történetben Trout elhatároztatta, hogy a legteljesebb mértékben élnek tulajdonosi jogaikkal. Mindenre kiteszik a Tilos az átjárás táblákat.
</p>
Kurt Vonnegut
</body>
</html>
```

A dokumentum első bekezdése a „Tilos az átjárás még neked is! szöveg, amely után egy pár nélküli `<p>` címke hatására új bekezdés kezdődik. A `<p align="center">` után a bekezdés szövege középre igazított, de a `</p>` után ismét az alapértelmezett balra igazítás lép érvénybe a Kurt Vonnegut kiírásánál. Az alábbi képernyőképen figyeljük meg a bekezdések közötti térközöket!

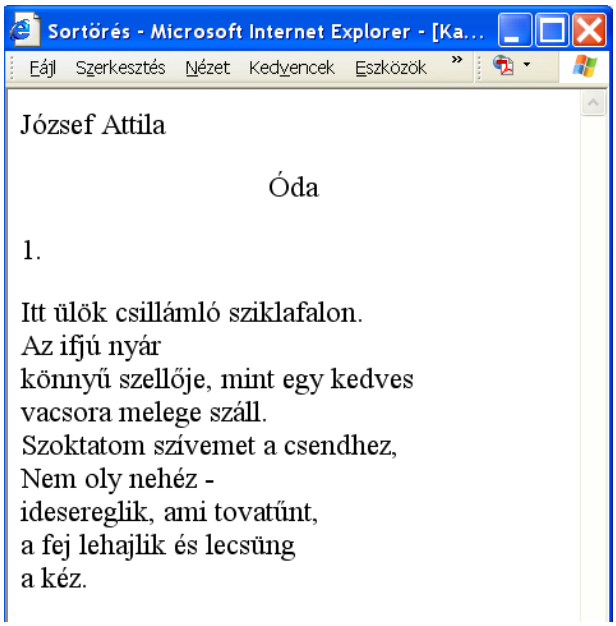


**
- sortörés**

Lezáró pár nélküli címke, ezért a lezárás jelzését önmagában tartalmazza.. A `<br />` helyétől a szöveg (illetve a szövegben lévő kép) megjelenítését a böngésző új sorban folytatja. Szemben a bekezdéstöréssel, sem a sortörés előtt, sem az után nem jelenik meg a sortávnál nagyobb térköz.

```
<html>
<head>
<title>Sortörés</title>
</head>
<body>
József Attila
<p align="center">
Óda
<p>
1.
</p>
<p>
Itt ülök csillámló sziklafalon.<BR>
Az ifjú nyár<br />
könnyű szellője, mint egy kedves<br />
vacsora melege száll.<br />
Szoktatom szívemet a csendhez,<br />
Nem oly nehéz -<br />
idesereglik, ami tovatűnt,<br />
a fej lehajlik és lecsüng<br />
a kéz.
</p>
</body>
</html>
```

Egy szűkített böngészőablakban az eredmény:



Az eredmény nem szép, de talán tanulságos. A sortörések használata mellett figyeljük meg a bekezdéscímkék működését is.

Korábban említettem, hogy milyen elemek szerepelhetnek a bekezdés blokkjában. Bekezdésben értelemszerűen nem lehet másik bekezdés. Ha mégis teszünk bele bekezdésjelet, akkor azon a ponton az előző bekezdés véget ér tekintet nélkül arra, hogy hol a lezáró `</p>` tagja. Az *Óda* cím alatt elhelyezett `<p>`-nél vége a középre igazított bekezdésnek, s mivel nem rendelkezik az igazítás módjáról, ettől kezdve az alapértelmezett balra igazítás lép érvénybe.

`clear` – a kép melletti terület üresen hagyása

A `<br />` paramétereként a kép és a szöveg viszonyának beállításához használható. Segítségével megoldható, hogy a sortörést követő szöveg ne a kép mellett, hanem mindenképpen alatta folytatódjon. Lehetséges értékei: `left`, `right`, `all`. Ha `left` értéket adunk meg, akkor a soremelés után addig lépked lefelé, amíg a bal margó mellett üres sorra jut, amelyben már nincs kép, s ott folytatja a szöveget. A `right` esetében a jobb margón, míg az `all` esetében mindkettőn figyeli, hogy az új sor képmentes legyen. Példa a balra igazított kép alá vitt szövegre:

```
<html>
<head>
<title>Sortörés</title>
</head>
<body>
<img      src="foglalkozas.jpg"
align="left" />
```
Ma a foglalkozások többségében valamilyen módon alkalmaznak számítógépet. Az irodai munkában főleg szövegszerkesztésre, vagy táblázatkezelésre használják.
```
<br clear="left" />
```
A mérnöki munkahelyek elképzelhetetlenek a tervezést és a gyártást segítő CAD illetve CAM rendszerek nélkül.
```
</body>
</html>
```

Az eredmény:

Ma a foglalkozások többségében valamilyen módon alkalmaznak számítógépet. Az irodai munkában főleg szövegszerkesztésre, vagy táblázatkezelésre használják.

A mérnöki munkahelyek elképzelhetetlenek a tervezést és a gyártást segítő CAD illetve CAM rendszerek nélkül.

3.2 Szövegtagolás megőrzése

Előfordul, hogy a HTML dokumentumokban érvényes szövegtagolási szabályok, mint például a szövegben lévő ismétlődő szóközök illetve a sorvége jelek figyelmen kívül hagyása, vagy a soroknak az ablakmérethez igazodó tördelése, nem felel meg céljainknak. Például ha egy parancs szintaxisát akarjuk megadni, nem lenne jó, ha a böngésző több sorba írná, amit az operációs rendszer számára egyetlen sorként kell beírni. Hasonló a helyzet, ha egy program meghatározott módon tagolt forráskódját szeretnénk megjeleníteni, vagy ha a dokumentumba üres sorokból álló részt akarunk beiktatni.

Az „előre formázott” azt jelenti, hogy a böngésző a forráskódban szereplő szöveget karakterről karakterre úgy jeleníti meg, ahogy azt beírtuk a szövegszerkesztőbe, tehát megtartva a többszörös szóközöket, tabulátorokat és az eredeti sortöréseket.

<pre> -előre formázott szöveg a HTML dokumentumban

A `<pre>` és `</pre>` címkék között elhelyezett szöveget az előbbiek értelmében vett „előre formázott” módon jeleníti meg, *állandó helyfoglalású* (nem arányos) *betűtípussal szedve*. Ilyen betűtípus pl. a Courier.

A tabulátor karakterek kezelése nincs szabványosítva, a legtöbb böngészőben 8 karakterenként helyezkednek el a tabulátorhelyek. A nem arányos betűtípus alkalmazásának előnye, hogy a szóközök is állandó méretűek, így a soron belül a szöveg pozícionálására használhatók. Ez *ellentmond a szövegszerkesztésnél tanultaknak,* de ne feledjük, a HTML-ben a tabulátor nem olyan kifinomult eszköz, mint a szövegszerkesztőknél, ezért ami ott tilos, az itt jobb híján elfogadható.

A `<pre>` blokkját a böngészők új bekezdésként jelenítik meg, tehát új sorba kerül a szöveg és térköz határolja el környezetétől.

Ha a sorok nem férnek el a böngésző ablakában, akkor a felhasználó a vízszintes gördítősáv használatával tudja csak elolvasni.

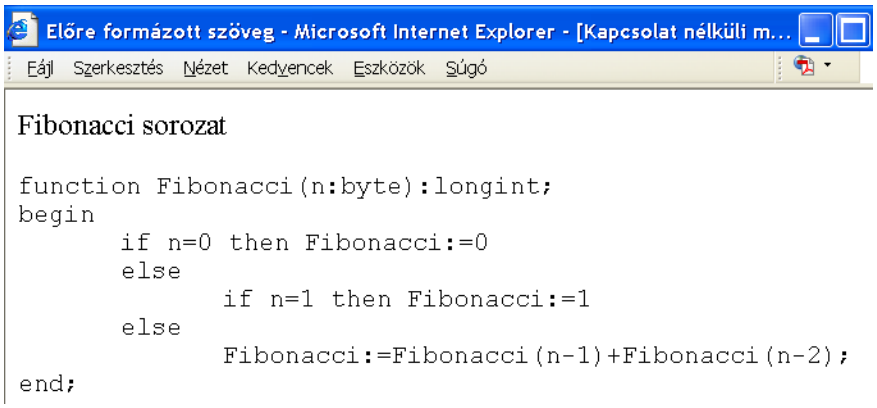
A blokkon belül szövegformázó címkék, hiperhivatkozás és kép is elhelyezhetők. A közvetlenül (`<p>`) vagy közvetve (pl. `div`) új bekezdéssel járó címkék a `<pre>` blokkján belül a szabvány szerint nem használhatók. A böngészőtől függ, hogyan értelmezi, ha mégis talál ilyent a blokkban.

Ha a szövegben a „<” vagy „>” illetve a „&” karakter is szerepel, akkor ezeket a HTML címkéktől megkülönböztetendő a következő karaktersorozatokkal kell helyettesíteni: „<” esetén `&lt;` „>” esetén `&gt;` „&” esetén `&amp;`

Példa előre formázott szöveg megjelenítésére:

```
<html>
<head>
<title>Sortörés</title>
</head>
<body>
Fibonacci sorozat
<pre>
function Fibonacci(n:byte):longint;
begin
    if n=0 then Fibonacci:=0
    else
        if n=1 then Fibonacci:=1
    else
        Fibonacci:=Fibonacci(n-1)+Fibonacci(n-2);
end;
</pre>
</body>
</html>
```

Az eredmény:



Megjegyzés a nem törhető szóköz használatáról

Ha a szövegben nem törhető szóközre van szükség – pl. a mérőszám és a mértékegység jele között -, akkor írjuk be a helyére a következő karaktersorozatot: ` ` Az egyszerű szóközökkel ellentétben a nem törhető szóközök sorozatát is megjelenítik a böngészők, így a soron belül egy szóköznél nagyobb térköz is kialakítható.

3.3 Betűformázás

Az itt tárgyalandó elemek *a szöveg megjelenési stílusát megváltoztatják, de tagolását nem*. Tehát sem bekezdés-, sem sortörést nem okoznak, és szóközt sem szúrnak be. A címkék *párosak* és a megjelenítendő szövegbe ágyazódnak be.

Típusát tekintve kétféle formázót különíthetünk el

a) A szöveg tartalma alapján írjuk elő.

b) Az elérendő szövegstílus alapján írjuk elő.

Az *a)* típusú címkék többségét a 4.0 szabvány vezette be. Többségük igazán jól csak stíluslapokkal használható. Ugyanazt a megjelenést a *b)* típusú címkékkel is el tudjuk érni, de ha a következetesen használjuk, akkor a formázás mellett az *a)* típusúak más célra – pl. magyarázó szótár, irodalomjegyzék, stb. automatikus elkészíttetésére – is felhasználhatók. Ezeknek az elemeknek a szerepe az érettségi feladatok megoldásában csekély, ezért csak kiegészítő anyagként, röviden foglalkozom néhányukkal.

A *b)* típusú címkék alkalmazásakor azt közöljük a böngészővel, hogy a kijelölt szövegrész hogyan nézzen ki. Pl. milyen legyen a betű típusa, színe, mérete stb.

Néhány tartalom-orientált formázó

Címke	Leírás	Megjelenési mód
<cite>	Hivatkozott mű címe	Dőlt
<code>	Programkód *	Állandó helyfoglalású betűvel
<dfn>	Definiálandó fogalom kiemelése	Dőlt vagy stíluslappal megadott
	Kiemelt szöveg	Dőlt
<kbd>	Billentyűzettel beírandó szöveg	Állandó helyfoglalású betűvel
<q>	Rövid idézet	Dőlt
	Erősen kiemelt, kövéren szedett	Félkövér
<var>	Változónév vagy felhasználó által megadott adat*	Állandó helyfoglalású és/vagy dőlt

* Elsősorban számítástechnikai dokumentációk számára ajánlott

Betűstílusok beállítása

Az alábbi címkék önmagukban egy-egy formai tulajdonságot állítanak be, de egymásba ágyazva kombinálhatók. Példa:

Félkövér és dőlt szedésű szöveg

- félkövér szedés

A és közötti szöveget félkövér (bold) változattal szedi. Ha ez nem lehetséges, akkor a böngésző dönt a megjelenítés módjáról. (Pl. lehet inverz színezésű vagy aláhúzott, esetleg villogó a kijelölt szövegrész.) Formai szempontból egyenértékű a <STRONG> címkével.

** - dőlt szedés**

Az és közötti szöveget dőlt (italic) változattal szedi. Ha ez nem lehetséges, akkor a böngésző dönt a megjelenítés módjáról. (Lásd <B>) Formai szempontból egyenértékű az címkével.

<strike> - áthúzott szöveg

A <strike> és </strike> közötti szöveget áthúzva írja. Rövid változata <s> iletve </s>.

<sub> - alsó index

A _{és} közötti szöveget az aktuális betűmérettel, de egy *fél sorral lejjebb* írja.

<SUP> - felső index

A ^{és} közötti szöveget az aktuális betűmérettel, de egy *fél sorral feljebb* írja.

<u> aláhúzott szöveg

Az <u> és </u> közötti szöveget aláhúzva írja.

A betűméretre és betűtípusra hatással lévő címkék

<big> - a betűméret növelése

A <big> és </big> közötti szöveget egy méretfokozattal nagyobb betűkkel szedi, mint a környezet betűmérete. Egymásbaágyazva minden <big> egy újabb méretnövelést eredményez.

<small> - a betűméret csökkentése

A <small> és </small> közötti szöveget egy méretfokozattal kisebb betűkkel szedi, mint a környezet betűmérete. Egymásba ágyazva minden <small> egy újabb méretcsökkenést eredményez.

<tt> - állandó helyfoglalású szedés

A <tt> és </tt> közötti szöveget állandó helyfoglalású betűkkel (pl. Courier) írja. Hatása hasonló a <code> illetve <kbd> címkékéhez. Ha nem kifejezetten programkódról vagy számítógépnek szóló parancsról van szó, hanem csak állandó helyfoglalású karakterekkel akarunk egy szöveget írni, akkor a <tt> használandó.

A betű típusának, méretének és színének beállítása

<basefont> - a dokumentum alapértelmezett betűméretének beállítása

Alapértelmezés szerint a böngészők a 3-as méretfokozatot (lásd alább a betűméretről szóló megjegyzést) használják a normál szöveg megjelenítéséhez. A <basefont> segítségével ez megváltoztatható.

Eredetileg csak a betűméret megváltoztatására volt alkalmas. A modern böngészők ezen kívül már a betű típusának és színének beállítására szolgáló paramétereket is értelmezni tudják.

A <basefont> a lezáró </basefont> taggal vagy nélküle egyaránt alkalmazható. Ez utóbbi esetben érvényessége a dokumentum végéig vagy a legközelebbi <basefont> címkéig terjed. Tipikus helye a dokumentum fejrészében (<head>) van. Ekkor a teljes dokumentumra érvényesek a megadott beállítások. De „elvileg" szerepelhet majd mindenütt a dokumentumban, egymást követően akár több helyen is, bár az ilyen használat aligha indokolható. Ügyelni kell arra, hogy a böngészők a </basefont> után azonnal a kiinduló 3-as betűméretű alapértelmezésre váltanak át!

Megjegyzés a betűméretről

A HTML 3.2 szabvány a betűméretek *pont* mértkegységben történő megadása helyett az úgynevezett kiterjesztett betűméret modellt vezette be, amely *a betűk méretét 1-től 7-ig* jelöli, s a böngésző határozza meg, hogy az egyes fokozathoz hány pontos betűméretet rendel. (Pl. a MS Internet Explorer a 3-as méretet 12 pontos betűkkel valósítja meg). Mint említettük, a 3-as az alapértelmezett betűméret, egy méretfokozat nagyjából 20%-kos méretbeli eltérést jelent ehhez képest. Tehát pl. a 2-es méret az alapértelmezettnél kb. 20%-kal kisebb, az 5-ös kb. 40%-kal nagyobb méretet jelent.

A <basefont> paraméterei

size – ***a betűméret***

Értéke az imént elmondottak szerint 1 és 7 közötti szám.

face – ***a betűtípus***

Az alapértelmezett betűtípust adhatjuk meg segítségével. Lehetséges értékei olyan *betűtípusok nevei*, amelyekkel a felhasználó böngészője remélhetőleg rendelkezik. Részletesebben a <font> címke kapcsán szólok erről.

color – ***a betű színe***

A paraméter értékét a színek jelöléséről korábban tanultak szerint (lásd pl. <body> címke) vagy RGB kóddal, vagy szabványos színnévvel adhatjuk meg. Példa

```
<basefont face="arial" size="2" color="navy">
```

** - egy szövegrész betűtípusának megváltoztatása**

A és a között beállítja a szöveg megjelenítéséhez a betű típusát, színét és méretét. Paraméterei hasonlóak a <basefont> címkénél ismertetett paraméterekhez.

`size` – *a betűméret*

A betűméret megadása *abszolút* vagy *relatív* módon történhet. Az első esetben a *size* értéke 1 és 7 közötti egész szám. (Lásd a *Megjegyzés a betűméretről* című részt.)

A betűméret relatív megadása azt jelenti, hogy az alapértelmezett betűmérettől való eltérést adjuk meg. Az alapértelmezett betűméret 3-as, de ha a `<basefont>`-tal ezt megváltoztattuk, akkor az így beállított méretet kell alapnak tekinteni. Méretcsökkentés esetén a negatív, növelés esetén pozitív előjelet kap a változás mérőszáma. Példa

```
<font size="+2">Két fokozattal nagyobb</font>
```

Ha az alapértelmezett betűméretet nem változtattuk, akkor a formázás eredményekén 5-ös betűmérettel írja ki a böngésző a „Két fokozattal nagyobb” szöveget.

Bármelyik módszerrel adjuk meg a betűméretet, ha a méret kisebb lenne 1-nél, akkor 1-nek, ha nagyobb lenne 7-nél, akkor 7-nek veszi a böngésző.

Nyilvánvaló, hogy a `size="+1"` azonos hatású a `<big>` címkével, a `size="-1"` a `<small>` címkével. E címkék használatában más tekintetben azonban lényeges különbség, hogy a `<big>` illetve a `<small>` többszöri egymásba ágyazásával a méretváltozások összegződnek, míg a `<font> size` paraméterénél nem, mivel annak *alapja nem a pillanatnyi szövegméret, hanem az alapértelmezett szövegméret*.

`face` – *a betűtípus*

Az alapértelmezett betűtípust adhatjuk meg segítségével. Lehetséges értékei olyan *betűtípusok nevei*, amelyekkel a felhasználó böngészője remélhetőleg rendelkezik. Mivel a böngészőtől és a gépre telepített betűtípusoktól függ, hogy a megjelölt igényt a böngésző képes-e teljesíteni, a *face* értékénél vesszővel elválasztva több betűtípust is felsorolhatunk. A felsorolás sorrendjében haladva a böngésző az első rendelkezésére álló betűtípust alkalmazza. Ha egyik megjelölt betűtípussal sem rendelkezik, akkor az alapértelmezett betűtípusát használja.[2] Példa a paraméter használatára:

```
<font face="arial, garamond, helvetia">
```

`color` – a betű színe

A paraméter értékét a színek jelöléséről korábban tanultak szerint (lásd pl. `<body>` címke) vagy RGB kóddal, vagy szabványos színnévvel adhatjuk meg. Példa:

```
<font face="sans-serif" color="#ffff00" size="+1">
```

Megjegyzés a `<font>` használatához

A `<font>` címkét kifejezetten a szöveg kisebb blokkjainak formázásához használjuk, ha a dokumentum egészében ugyanazokat a formai jellemzőket kell használni, akkor alkalmazzuk a `<basefont>` címkét. Ha csak egy méretfokozatú növelést vagy csökkentést kell végrehajtani, és a szövegrész néhány szó hosszúságú, akkor inkább a `<big>` illetve `<small>` használata javasolt.

3.4 Címsorok formázása

A HTML dokumentumokban címek formázására – ha csak lehet –, használjuk a beépített címsor stílusokat. E stílusok sorszámukkal azonosíthatók, 1-től 6-ig hierarchikus rendet alkotnak, a legfelső szint az 1-es. A címsor stílussal formázandó szövegrész jelölése: `<h`*n*`>` szöveg `</h`*n*`>`, ahol az *n* helyébe a szint sorszáma kerül. Példaként nézzünk egy kódrészletet:

[2] A MS Internet Explorer esetén az *Eszközök\Internet beállítások\Betűkészletek* alatt tekinthetjük meg a rendelkezésre álló betűkészleteket.

```
<body>
<h1>A HTML alapjai</h1>
<h2>Bevezetés</h2>
Ebben a segédletben nem közlöm a
HTML összes nyelvi elemét
</body>
```

A HTML alapjai

Bevezetés

Ebben a segédletben nem közlöm a HTML összes nyelvi elemét

Az eredmény a jobb oldalon látható. Megfigyelhető, hogy a címsor új sorban kezdődik és térköz marad el előtte is utána is – tehát *bekezdésszerűen* tördeli a böngésző. A `<Hn>` paramétere:

align – a cím igazítása

Lehetséges értékei: `left` (alapértelmezett), `center`, `right`. Példa:

```
<h1 align=”center”>A HTML alapjai</h1>
```

A címsorokat következetesen kell alkalmazni. A logikai értelemben azonos szinten álló címeket minden fejezetben azonos címsor stílussal kell formázni. A felhasznált címsor stílusok között ne legyen kihagyás. Pl. ha használjuk a `<h1>` és `<h2>` stílust és még szükség van egy alacsonyabb szintű stílusra, akkor a `<h3>` stílus legyen a következő és nem valamelyik még alacsonyabb szintű!

3.5 Az idézetek kezelése

`<blockquote>` - hosszabb idézet

A szövegben elhelyezett hosszabb, egy vagy több bekezdésnyi idézetet a `<blockquote>` és a `</blockquote>` címkék közé téve formázhatjuk.

A `<blockquote>` hatására a böngésző *külön bekezdésbe teszi a szöveget* és mind a bal, mind a jobb margóhoz képest behúzva szedi. Néhány böngésző dőlt betűket is alkalmaz. Megemlítendő, hogy eltérően kezelik a böngészők a `<blockquote>` szakaszban elhelyezett képeket. Lehet, hogy az egyik böngésző a képet a bal margóhoz, míg egy másik a behúzáshoz illeszti.

`<q>` - rövid idézet

A szövegben elhelyezett rövidebb idézeteket a `<q>` és a `</q>` közé téve formázhatjuk. Hatására a szöveg tagolása nem változik, tehát nem kerül új sorba, vagy új bekezdésbe, a böngészők általában dőlt, esetleg valamilyen állandó helyfoglalású betűtípussal szedik.

4. Listák a dokumentumban

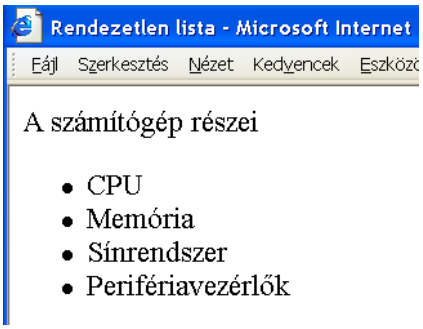
A felsorolások szövegszerkesztőben megszokott két változata a HTML dokumentumokban is megvalósítható. A szövegszerkesztőben felsorolásnak nevezett változatot itt *rendezetlen listának* (unordered list), a számozott felsorolásokat rendezett listának (ordered list) nevezik. Ez utóbbiakat akkor használjuk, ha nem közömbös a tételek felsorolási sorrendje.

4.1 Rendezetlen listák

Egy rendezetlen lista az `<ul>` címkével kezdődik és kötelezően a `</ul>` zárja le. A kettő között helyezkednek el a felsorolt tételek. Minden tétel szövege a `<li>` és a `</li>` címkék között helyezkedik el. Az `<ul>` és `</ul>` közötti tartalmat a böngésző új bekezdésként, a balmargótól behúzva jeleníti meg, így a felsorolt tételeket is behúzva helyezi el az oldalon. Minden egyes tételt felsorolás jel vezeti be. Példa:

```
<html>
<head>
<title>Rendezetlen lista</title>
</head>
<body>
A számítógép részei
<ul>
<li>CPU</li>
<li>Memória</li>
<li>Sínrendszer</li>
<li>Perifériavezérlők</li>
</ul>
</body>
</html>
```

A böngészőben látható eredmény:



Az `<ul>` és a `<li>` paraméterei:

`type` - a felsorolásjel típusának beállítása

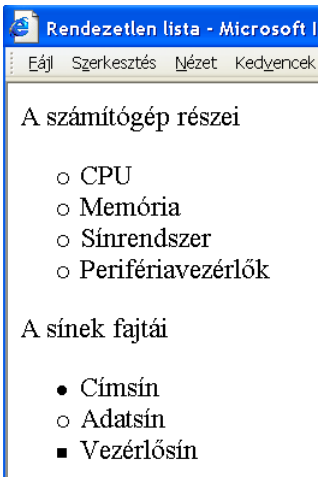
A böngészőprogramok alapértelmezésben – mint az iménti példában látható – kis fekete pöttyöt használnak felsorolásjelként. A `type` paraméterrel ez megváltoztatható. Lehetséges értékei: `disc` (alapértelmezett), `circle` (azaz kör) és `square` (azaz négyzet).

Figyelem! A MS Internet Explorer csak kisbetűs alakban fogadja el ezeket az értékeket

Ha a `type` az `<ul>` címkében helyezkedik el, akkor a beállítás a lista minden tételére érvényes a `</ul>`-ig, ha a `<li>` címkébe tesszük, akkor csak az adott tétel felsorolásjelét változtatja meg.

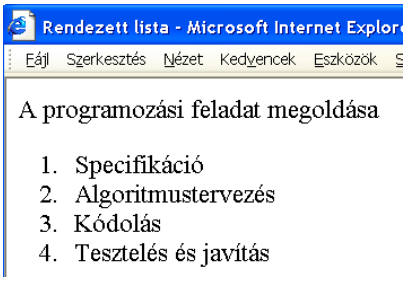
Ha az előbbi HTML kód megfelelő részét kicseréljük a következőre, akkor mindkét esetre láthatunk példát.

```
A számítógép részei
<ul type="circle">
<li>CPU</li>
<li>Memória</li>
<li>Sínrendszer</li>
<li>Perifériavezérlők</li>
</ul>
A sínek fajtái
<ul>
<li typE="disc">Címsín</li>
<li type="circle">Adatsín</li>
<li type="square">Vezérlősín</li>
</ul>
```



4.2 Rendezett listák

A rendezett listák felépítése és megjelenítése hasonló a rendezetlenekéhez, a különbség az, hogy a felsorolásjel a böngésző által automatikusan beillesztett szám. A rendezett lista az `<ol>` címkével kezdődik és kötelezően a `</ol>` zárja le. A kettő között helyezkednek el a felsorolt tételek. Minden tétel szövege a `<li>` és a `</li>` címkék között helyezkedik el. Példa:




```
A programozási feladat megoldása
<ol>
<li>Specifikáció</li>
<li>Algoritmustervezés</li>
<li>Kódolás</li>
<li>Tesztelés és javítás</li>
</ol>
```

type - a számozás jelének beállítása

Alapértelmezésben a felsorolásjel arab szám, a kezdőérték 1. A `type` paraméterrel ez módosítható. Lehetséges értékei és az értékek jelentése az alábbi táblázatban áttekinthető. A második oszlopban az egyértelműség érdekében az értékeket szóban is megnevezzük.

Érték	Szóban	Felsorolásjel	Példa
A	Nagy *a* betű	Nagybetűk	A., B., C., D.
a	Kis *a* betű	Kisbetűk	a., b., c., d.
I	Nagy *i* betű	Nagybetűkkel római számok	I., II., III., IV.
i	Kis *i* betű	Kisbetűkkel római számok	i., ii., iii., iv.
1	Egyes számjegy	Arab számok (alapértelmezett)	1., 2., 3., 4.

Ha a `type` az `<ol>` címkében helyezkedik el, akkor a beállítás a lista minden tételére érvényes a `</ol>`-ig, ha a `<li>` címkébe tesszük, akkor csak az adott tétel felsorolásjelét változtatja meg.

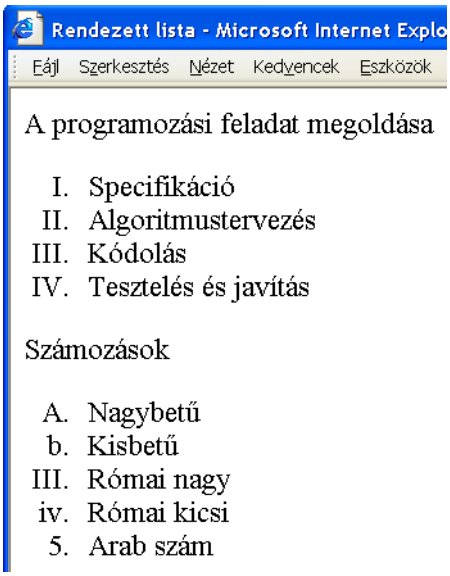
Példa:

A programozási feladat megoldása

```
<ol type="I">
<li>Specifikáció</li>
<li>Algoritmustervezés</li>
<li>Kódolás</li>
<li>Tesztelés és javítás</li>
</ol>
```

Számozások

```
<OL>
<li type="A">Nagybetű</li>
<li type="a">Kisbetű</li>
<li type="I">Római nagy</li>
<li type="i">Római kicsi</li>
<li type="1">Arab szám</li>
</ol>
```

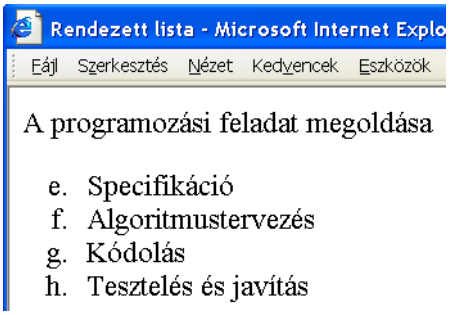


START – a kezdősorszám beállítása

Alapértelmezés szerint a kezdősorszám 1, de ez az `<OL>` `START` paraméterével megváltoztatható. Lehetséges értékei arab számjeggyel jelölve egész számok. Példa:

A programozási feladat megoldása

```
<ol type="a" start="5">
<li>Specifikáció</li>
<li>Algoritmustervezés</li>
<li>Kódolás</li>
<li>Tesztelés és javítás</li>
</ol>
```



4.3 Definíciós listák

A HTML által támogatott listák harmadik fajtáját elsősorban magyarázattal ellátott *szójegyzék* (glosszárium, ang.: glossary) készítésére használjuk. A lista tételeit a magyarázandó szavak, kifejezések alkotják, mindegyik alatt, balról behúzva helyezkedik el a magyarázata.

A definíciós lista (definition list) a `<dl>` és a `</dl>` címkék között helyezkedik el. Minden tétele az imént elmondottaknak megfelelően két részből áll. Az egyik a definiálandó kifejezés (term), amit

a <dt> és </dt> címkék fognak közre, alatta a definíciója, amit a <dd> és </dd> címkék közé kell tenni. Példa:

```
<dl>
<dt>Protokoll</dt>
<dd>Kommunikációs szabályrendszer. A sikeres kommunikációnak előfeltétele, hogy az adó és a vevő megeggyezzen olyan kérdésekben, mint a kódolás és dekódolás, az átviteli hibák kezelése, az adatátviteli csatornán a forgalom szabályozása stb.</DD>
<dt>Tűzfal</dt>
<dd>Szerepe a hálózati kapcsolat ellenőrzése, szűrése a veszélyes hálózati kapcsolatok kizárása érdekében. A tűzfal lehet önálló hálózati eszköz, amely egy egész hálózatot véd, kialakítható egy számítógépből is, de lehet egy szoftver is, amely egyetlen számítógépet véd</dd>
<dt>URL</dt>
<dd>A Uniform Resource Locator kifejezésből származó mozaikszó. A számítógép-hálózatokon elérhető fájlok helyének szabványos, operációs rendszerektől és fájlrendszerektől független leírási módja.</dd>
</dl>
```

Az eredmény:

Protokoll
: Kommunikációs szabályrendszer. A sikeres kommunikációnak előfeltétele, hogy az adó és a vevő megeggyezzen olyan kérdésekben, mint a kódolás és dekódolás, az átviteli hibák kezelése, az adatátviteli csatornán a forgalom szabályozása stb.

Tűzfal
: Szerepe a hálózati kapcsolat ellenőrzése, szűrése a veszélyes hálózati kapcsolatok kizárása érdekében. A tűzfal lehet önálló hálózati eszköz, amely egy egész hálózatot véd, de lehet egy szoftver is, amely egyetlen számítógépet véd

URL
: A Uniform Resource Locator kifejezésből származó mozaikszó. A számítógép-hálózatokon elérhető fájlok helyének szabványos, operációs rendszerektől és fájlrendszerektől független leírási módja.

4.4 Beágyazott listák

A többszintű felsorolások lényegében olyan listák, amelyekbe egy vagy több másik lista van beágyazva. A következő szint mindig egy lista tételnél épül be a listába, e tétel záró tagja ezért a beágyazott lista után helyezkedik el. Az alsóbb szint automatikusan nagyobb behúzást kap.

Beágyazott rendezetlen lista

Az alsóbb szint automatikusan más típusú felsorolás szimbólumot kap a böngészőtől. Példa:

```
A számítógép részei
<ul>
<li>CPU
<ul>
<li>Vezérlő egység</li>
<li>Aritmetikai-logikai egység</li>
</ul>
</li>
<li>Memória</li>
<li>Sínrendszer
<ul>
<li>Címsín</li>
<li>Adatsín</li>
<li>Vezérlősín</li>
</ul>
</li>
<li>Perifériavezérlők</li>
</ul>
```

A számítógép részei

- CPU
  - Vezérlő egység
  - Aritmetikai-logikai egység
- Memória
- Sínrendszer
  - Címsín
  - Adatsín
  - Vezérlősín
- Perifériavezérlők

Beágyazott rendezett lista

Az előzőtől csak annyiban különbözik, hogy minden szint az alapértelmezett arab számozást kapja. Ha ez nem felel meg, akkor a *type* paraméter segítségével beállíthatjuk a megfelelő sorszámozást. Példa:

A programozási feladat megoldása

```
<ol>
<li>Specifikáció</li>
<li>Algoritmustervezés</li>
<li>Kódolás
<ol type="a">
<li>A programozás nyelv kiválasztása</li>
<li>A fejlesztőrendszer kiválasztása</li>
<li>A program forráskódjának elkészítése</li>
</ol>
</li>
<li>Tesztelés és javítás</li>
</ol>
```

A programozási feladat megoldása

1. Specifikáció
2. Algoritmustervezés
3. Kódolás
   a. A programozás nyelv kiválasztása
   b. A fejlesztőrendszer kiválasztása
   c. A program forráskódjának elkészítése
4. Tesztelés és javítás

5. Grafikus elemek, képek a dokumentumban

5.1 Vízszintes tagolóvonal

A dokumentum erőteljes függőleges tagolását szolgálja a `<hr />` *pár nélküli elem*. Helyére a böngésző egy vízszintes vonalat szúr be a dokumentumba. A vonal új sorba kerül, pontosabban a vonal előtt és után sortörést is végez a böngésző. Ha szövegtől vett távolság nem elegendő, megnövelhetjük bekezdéstörés (`<p>`) elhelyezésével. A `<hr />` visszaállítja a bekezdés igazítását az alapértelmezett balra zártra.

A vonal megjelenése meglehetősen eltérő az egyes böngészőkben, alapértelmezésben domború vagy homorú és árnyékolt jellegű, az ablak bal szélétől a jobb széléig terjed. A `<hr />` paraméterei:

`align` – a vonal igazítása

Lehetséges értékei: `left`, `right`, `center` (alapértelmezett).

`NOSHADE` – a 3D hatás kikapcsolása

Nem kell értéket megadnunk hozzá. Hatására egy egyszerű fekete vonalat kapunk.

`size` – a vonal vastagsága

A böngészők általában 2-3 pixel vastagságú vonalat használnak alapértelmezésben. Ezt a SIZE paraméterrel megváltoztathatjuk. Az értéket pixelekben kell megadni. (Így természetesen csak egész szám jöhet szóba.) Példa:

```
<hr width=”5” />
```

`width` – a vonal hossza

Alapértelmezésben a vonal a teljes böngészőablakot átfogja. Ha ez nem felel meg, a vonal kívánt hosszát a width paraméterrel megadhatjuk. A hosszúságot pixelben, vagy a böngészőablak aktuális szélességének százalékában adhatjuk meg. Példa a két lehetőségre:

```
<hr width=”200” />
<hr width=”33%” />
```

A böngészőablak változó szélessége miatt általában nem célszerű a méretet pixelekben megadni.

5.2 Képek

Grafikus formátumok a HTML dokumentumokban

A szabvány számos grafikus formátumot ír le, de a böngészők eltérnek abban, hogy melyek használatát támogatják. Bármilyen grafikus böngészőről legyen szó, arra számíthatunk, hogy a GIF és a JPG (JPEG) formátumú képeket megjelenítik.

A GIF forátum

Ez volt az első grafikus formátum, amelyet a *CompuServe* cég kifejezetten azzal a céllal fejlesztett ki, hogy a felhasználók a gépükre telepített operációs rendszerektől és alkalmazásoktól függetlenül használható képeket küldhessenek egymásnak. Első változata a GIF87a, ennek továbbfejlesztésével jött létre a GIF89a.

A GIF jellemzői:

- veszteségmentes tömörítés (Lempel-Zev-Welch, röv.: LZW algoritmust használ)
- 8 bites színmélység (max. 256 szín)
- indexelt színkezelés (Nem rögzített a lehetséges színek halmaza, hanem a fájl tartalmazza a képen lévő színek készletét az ún. színtáblában. Az egyes képfájlban a képpontokhoz tárolt értékek a színtáblára hivatkoznak, onnan derül ki, hogy milyen színű ténylegesen a pont. Így az ábrázolható 256 szín igazodik a kép színeihez.)
- váltott soros (interlaced) megjelenítésre alkalmas (A képet nem sorfolytonosan tölti le a böngésző, hanem a kép előbb durva felbontással jelenik meg, majd ahogy letöltődik a képfájl többi része, fokozatosan egyre finomabb alakot ölt. A kis adatátviteli sebességű internet hozzáféréssel rendelkezők számára ez előnyösebb megoldás.)
- transzparens (átlátszó) megjelenítésre alkalmas (A színtábla egy színe kijelölhető átlátszó színnek. Ez azt jelenti, hogy ezt a színt képen a kép hátterének színével helyettesíti a böngésző.)
- animálható formátum (Megfelelő szoftverrel egyetlen GIF állományba több képet is tárolhatunk. Ezek lehetnek egy mozgás különböző fázisai, amelyeket a böngésző megfelelő időzítéssel egymás után ábrázol.)

A 256 szín még az indexelés módszerével is kevés fényképszerű képekhez. A GIF formátumot elsősorban egyszerű alakzatokat és kevés színt tartalmazó ábrákhoz (pl. logó, szemléltető rajz, műszaki rajz stb. alkalmazzuk.

A JPEG formátum

Ha az ábrázolandó kép formákban és színárnyalatokban gazdag (tipikusan ilyenek a fényképek, festményekről készült képek), akkor a GIF formátum nem alkalmas a kép ábrázolására. Ha növeljük a színmélységet, akkor ezzel együtt erőteljesen növekszik a képet ábrázoló adatmennyiség, azaz a képfájl mérete. Ez azonban az aránylag lassú internet kapcsolatoknál azt jelentené, hogy elfogadhatatlanul lassan jelennének meg a weblapokon a képek. Erre a problémára jó megoldást jelentett a Joint Photographic Experts Group által kifejlesztett JPEG formátum.

A JPEG jellemzői:

- 24 bites maximális színmélység
- veszteséges tömörítés (Az emberi szem érzékelési sajátosságait kihasználva a tömörítés során elhagy olyan adatokat, amelyek hiányát a képben a szem nem, vagy csak kis mértékben veszi észre. Így rendkívüli fájlméret csökkenés, akár 1:20 arányú tömörítés érhető el.)
- Változtatható a tömörítés, ezzel együtt a veszteség aránya. (Így jól összeegyeztethető a minél kisebb fájlméretre törekvés a célnak megfelelő képminőséggel.)
- az ún. *progresszív JPEG* változat a GIF-hez hasonló fokozatosan finomodó megjelenítést tesz lehetővé.

Megjegyzés a PNG formátumról

Mivel a GIF formátum szabadalmi védettség alá esik, a jogdíjfizetési kötelezettsége elkerülése érdekében, a GIF alternatívájaként használható, és már a web-technológia igényeihez is jobban alkalmazkodó formátumot dolgoztak ki. Az adatformátum a *Portable Network Graphic* nevet kapta. A fejlesztés technikai értelemben sikeres volt, egy nagyon rugalmas, sokoldalú, a GIF-nél jobb megoldást eredményezett. A W3C 1996-ban be is emelte a HTML szabványba. Sajnos a böngészők

és a szoftverfejlesztők nem fordítottak elég figyelmet a PNG (ejtsd: ping) formátum támogatására, így ma sem minden böngésző illetve grafikus szoftver támogatja használatát.

A képek jellemző előfordulásai

- A dokumentum háttérképe (lásd `<body>`)
- A szöveget illusztráló kép vagy ábra a szövegben vagy táblázatcellában elhelyezve. A dokumentumban a `<pre>` blokkok kivételével ahol szöveg lehet, ott lehet kép is.
- Hiperhivatkozás, amelyre kattintva a hivatkozott dokumentumhoz jutunk.
- Térkép, amelynek egyes pontjai hiperhivatkozásként működnek.

Az URL fogalma

Mivel a HTML dokumentumokban lévő képek önálló fájlokként töltődnek le, és a böngésző rakja össze az oldalt a megjelenítéshez, a következőkben szükségünk lesz arra, hogy megadjuk hogyan érhető el az oldalon elhelyezendő képfájl.

A nehézség két részből áll:

1. A képfájlok egy számítógép hattértárolóján helyezkednek el. Akár a web szervereket, akár a böngészőket nézzük többféle operációs rendszerrel (MS Windows változatok, UNIX változatok, Macintosh OS, stb.) és fájlrendszerrel (FAT, NTFS, EXT3 stb.) üzemelő gépeken lehetnek. Ezeknél eltérő szabályok érvényesek a fájlnevekre, könyvtárnevekre, az elérési út leírására stb. Ha az egyik rendszerhez igazodva írjuk le a fájl elérési útját, a többi nem lesz képes értelmezni azt. Például a MS operációs rendszereknél a mappák és fájlok nevében nem számít a különbség a kisbetűk és a nagyok között, pl. akár *arckep.jpg*, akár *ARCKEP.JPG* írásmóddal adjuk meg a fájlnevet, ha a fájl az adott helyen létezik, a rendszer megtalálja. Ezzel szemben a UNIX operációs rendszerek mind a könyvtárak, mind a fájlok nevében érzékeny a kis és a nagybetűk különbségére, az iménti nevek tehát két különböző fájlnevet jelentenek.
2. A fájlok nem a helyi, hanem egy hálózaton elérhető távoli gépen helyezkednek el. A hálózati adatátvitelt protokollok szabályozzák, így nem elég azt megadni, hogy hol van és mi a neve az igényelt fájlnak, hanem azt is meg kell adni, hogy milyen protokollt használva lehet azt elérni.

E problémák megoldására dolgozták ki a *Uniform Resource Locator*, röviden *URL* (egységes módszer az erőforrás helyének meghatározására) fogalmát, illetve módszerét. *Az URL* (angol kiejtése: you are ell*) lehetővé teszi, hogy operációs rendszertől függetlenül, szabványos módon hivatkozhassunk a számítógép-hálózatokon elérhető erőforrásokra*, leggyakrabban fájlokra.

Az URL általános felépítése

`<protokoll>://gépcím:<port>/<útvonal>/<fájlnév>`

protokoll
: megadja, hogy milyen típusú hálózati kapcsolatot használva lehet a szükséges adatátvitelt megvalósítani,

gépcím
: a távoli gép DNS címe vagy IP címe, amely megadja, hogy a fájl melyik számítógépen van tárolva,

port
: a távoli gép melyik hálózati portjára kell kapcsolódni az adatátvitelhez (ha nincs megadva, akkor a protokollhoz tartozó alapértelmezett portcímet kell használni (pl. http esetén a 80, ftp esetén 21 az alapértelmezett portcím,

útvonal
: könyvtárak (mappák) olyan „/” jellel elválasztott sorozata, amely leírja, hol helyezkedik el a távoli gép fájlrendszerében a szóban forgó fájl,

fájlnév
: az elérendő fájl neve az esetleges kiterjesztéssel együtt.

Példa:

```
http://info.lovassy.hu:8000/web/pld/arckep.jpg
```

Az URL a következő információkat tartalmazza: *az info.lovassy.hu* című géphez a *8000*-es porthoz kapcsolódva *http* protokoll szerint érhetjük el a fájlt, amely a webszerver gyökérkönyvtárából a *web/pld* úton haladva *arckep.jp*g neven található meg.

Az URL helyes használatához

Milyen karakterek használhatók?

Mivel még működnek olyan rendszerek, amelyek nem tudják kezelni a 8 bites karakterkészletet sem, az URL-ben csak nem szerepelhetnek a 32-nél kisebb és a 127-nél nagyobb kódhoz tartozó karakterek. (Pontosabban egy speciálisan kódolt formában szerepelhetnek, de ezzel nem foglalkozunk)

Nem használható a szóköz karakter, továbbá azok a karakterek sem, amelyeknek HTML környezetben speciális szerepük van, mint pl. <,>,\,”,TAB,{,},~

Útvonal és fájlnév, a relatív URL

A fentiekből következően, függetlenül attól, hogy a távoli gép operációs rendszere hogyan kezeli a fájlneveket, a könyvtárak és fájlok nevében a kis- és a nagybetűk között különbség van. A nevek nem tartalmazhatnak szóközöket és a magyar ábécére jellemző ékezetes karaktereket.

A távoli gép többnyire egy webszerver, amelyen a szerver szoftver biztonsági okokból a teljes fájlrendszernek csak egy részét engedi elérni http kapcsolattal. Némi pontatlansággal úgy is felfogható, hogy a webszervernek egy külön fájlrendszere van. Ennek gyökere az ún. dokumentum-gyökérkönyvtár. Az URL-ben szereplő útvonal a webszerver gyökérkönyvtárától indul és nem a távoli gép teljes fájlrendszerének gyökerétől.

Ha egy HTML dokumentumot már lekértünk a távoli rendszerről, és most e dokumentum környezetéből egy újabb fájlra akarunk hivatkozni, akkor elegendő az ún. *relatív URL*-t megadni, tehát azt, hogy hogyan lehet a megnyitott dokumentum helyétől kiindulva elérni az újabb fájlt. Például ha a megnyitott dokumentum URL-je

```
http://www.pcshop.hu/pld/nyitvatartas.html
```

akkor a *nyitvatartas.html* dokumentumban már szerepelhet egy relatív URL pl. a következő módon:

```
../pic/logo.jpg
```

Ami azt jelenti, hogy az aktuális könyvtár szülőkönyvtárából nyíló *pic* könyvtárban a *logo.jpg* a hivatkozott fájl. Az aktuális könyvtáron itt azt a könyvtárat értjük, ahonnan a *nyitvatartas.html* fájlt a rendszer korábban elérte, azaz a távoli gép */pld* könyvtára.

Protokollok

HTML dokumentumokban a következő protokollok szerepelhetnek URL-ekben:

http A HTML dokumentumok átvitelére kifejlesztett protokoll, de nem csak html típusú fájlok átvitelére alkalmas, hiszen egy HTML dokumentumban sokféle fájl lehet alkotóelem. Mint fájlátviteli protokoll bármilyen típusú fájl átvitelére alkalmas.

ftp Két távoli számítógép között fájlátvitelt megvalósító protokoll (File Transfer Protocol). Egy f*tp szerveren elhelyezett fájlra* hivatkozhatunk az **ftp** protokollmegjelölés segítségével. A böngészőprogramok rendszerint képesek ftp kliensként is működni, így fel tudják építeni az ftp kapcsolatot és el tudják érni a fájlt minden segédprogram nélkül.
Ha az URL-ben nem adjuk meg a felhasználói nevet és a jelszót, akkor a böngésző anonymous felhasználóként próbál kapcsolódni az ftp szerverhez. Ha egy

felhasználói fiókhoz akarunk kapcsolódni, akkor a felhasználói nevet és jelszót a következőképpen építhetjük be az URL-be:

```
ftp://<felh.név>:<jelszó>@<gépcím>/<útvonal>/<fájlnév>
```

Példa:

```
ftp://jozsi:AbrakaDabra@server.valahol.hu/arckep.jpg
```

Ez a megoldás azonban biztonsági szempontból helytelen, hiszen a HTML dokumentum URL-jeit bárki elolvashatja!

mailto Elektronikus levél küldése kezdeményezhető e protokoll segítségével. Az URL felépítése ilyenkor eltér az általánostól:

```
mailto:<e-mail cím>
```

Példa:

```
mailto:webmaster@www.pcshop.hu
```

file A helyi számítógépen található dokumentumra utal, tehát olyan fájlra, amely ugyanazon a gépen van, mint amelyiken a böngésző fut, amellyel a fájlt megjelenítjük. Általános alakja:

```
file://<gépcím>/<útvonal><fájl>
```

A protokoll és a gépcím általában elhagyható és az URL ilyenkor a helyi gép fájlrendszerében írja le az elérési utat. Példa MS operációs rendszer esetére:

```
file://c:/data/pic/arckep.jpg
```

gopher	Dokumentumok távoli elérésére és keresésére a World Wide Webet megelőzően használt, ma már korszerűtlen technológia protokollja
news	A USENET protokollja, USENET hírcsoportokra, cikkekre lehet hivatkozni vele.
telnet	Bejelentkezést tesz lehetővé egy távoli gépre, amennyiben azon telnet szerver működik. Biztonsági okokból ez a protokoll elavultnak és kerülendőnek tekintendő.

<img> - képek elhelyezése a dokumentumban

Az `<img>` címke segítségével képet szúrhatunk be a HTML dokumentum szövegébe. Nem okoz sem bekezdéstörést, sem sortörést. Alapértelmezésben a beszúrt kép alsó széle egy vonalban lesz a szöveg alapvonalával.

Két kötelező paramétere van. Egyik az `src`, amelyik megadja a kép URL-jét, és így logikai értelemben is nélkülözhetetlen, a másik az `alt`, amellyel az ún. alternatív szöveg adható meg, s amelyet a 4.01-es szabvány tett kötelezővé, bár a böngészők nélküle is megjelenítik a képet.

Ne felejtsük el az XHTML szerint lezárni a címkét! Az `<img>` paraméterei:

`src` - a beszúrandó kép forrása

A kép abszolút vagy relatív URL-jét adja meg. A helyes gyakorlat az, hogy a szerző a képeket egy vagy néhány e célra szolgáló könyvtárba gyűjti össze, és relatív URL-t használ az src paraméterben.

`alt` – helyettesítő szöveg

Előfordul, hogy a böngésző nem képes az előírt képet megjeleníteni, pl. azért, mert hibás az URL, vagy éppen azért mert eleve egy karakter alapú böngészőről van szó (pl. Lynx, Elinks). A böngésző ilyenkor az alternatív szöveget jeleníti meg a kép helyén. A szöveg hossza maximum1024 karakter, szóközöket és központozást is tartalmazhat. Példa:

```
<html>
<head>
<title>Képek</title>
</head>
<body>
```
Alapértelmezésben a kép alsó széle `<img src="pics\foglalkozas.jpg" alt="Férfi számítógéppel" />`a szöveg alapvonalához illeszkedik.
```
</body>
</html>
```
Az eredmény

Alapértelmezésben a kép alsó széle 

a szöveg alapvonalához illeszkedik.

A böngésző ablakából kivágott részleten az is látható, hogy a modern böngészők az alternatív szöveget egy kis súgócédulán is megjelenítik, ha az egérkurzor a kép fölé kerül.

`height` és `width` – a kép magassága és szélessége

Két okunk is lehet arra, hogy a képek méretét megadjuk a böngészőnek. Az oldal szövegét a böngésző gyorsan letölti és megjeleníti. A rajta lévő képek azonban lassan töltődnek le. Ha nem adjuk meg a kép helyén annak a méretét, akkor a böngésző nem tudja kihagyni a helyet számukra. Ahogy letöltődik egy-egy kép és kiderül a mérete, a böngésző újratördeli az oldalt, ami kellemetlen a felhasználó számára. Ha megadjuk a képek méretét, a böngésző kihagyja a helyet számukra és elmarad az ismételt újratördelés.

A másik ok az lehet, hogy a böngésző képes nagyítani és kicsinyíteni a képeket, így a méret előírásával át is méretezhetjük a képeket. Ne számítsunk azonban fényes eredményre! Sokkal jobb eredményhez juthatunk, ha magát a képfájlban lévő képet méretezzük át egy erre a célra alkalmas grafikai programmal.

A `height` és `width` paraméterrel *képpontokban* mérve adhatjuk meg az adott dimenzióban a kép megjelenítési méretét. Ha csak egyik dimenzióban adjuk meg a méretet és a másik paraméter elmarad, akkor a böngésző a két oldal arányát megtartva megváltoztatja a másikat is. Példa:

```
<img src="foglalkozas.jpg" alt="Férfi számítógéppel" height=”200” width=”210” />
```

A legtöbb böngésző támogatja a képméret megadását az ablakméret százalékában is. Az előző példát kissé átalakítva:

```
<img src="foglalkozas.jpg" alt="Férfi számítógéppel" width=”50%” />
```

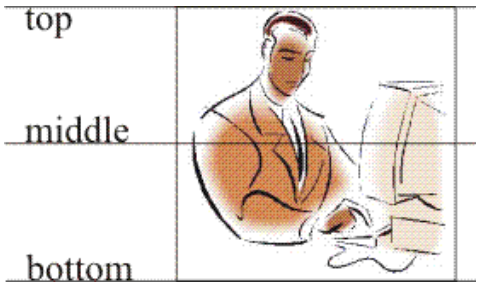
Hatására a kép szélességét az ablakszélesség felére veszi, a magasságát ehhez arányosan állítja be.

`align` – a kép igazítása

A kép és a környezetében lévő szöveg helyzetét ezzel a paraméterrel állíthatjuk be. A felhasználható paraméter értékek:

Függőleges igazítás

`top`	A kép felső széle a sor legmagasabb elemének a felső széléhez illeszkedik. Ha a szövegben nincs kép, akkor a szöveg felső széléhez igazodik.
`middle`	A kép vízszintes középvonala a szöveg alapvonalához illeszkedik
`bottom`	A kép alsó széle a szöveg alapvonalához illeszkedik.



A kép függőleges igazításának értelmezéséhez

Vízszintes igazítás és körülfuttatás

`left` A képet az ablak bal széléhez illeszti és a maradék szöveget a kép mellé írja ki. (Jobbról körülfolyatja a képet szöveggel.)

`right` A képet az ablak jobb széléhez illeszti és a maradék szöveget a kép mellé írja ki. (Balról körülfolyatja a képet szöveggel.)

```
<html>
<head>
<title>Képek</title>
</head>
<body>
<h2>Demográfiai változások</h2>
```

Időközben az általános jólét, szokásos módon az egy főre jutó nemzeti össztermék- (GDP-) egységekben kifejezve, szintén növekedett (persze óriási egyenlőtlenségekkel).

```
<p>
<img src="pics/demografia.gif" align="LEFT" />
```

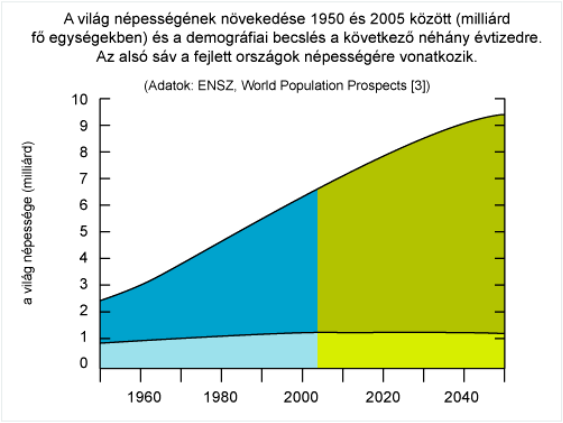
Az empirikus elemzések pedig egyértelműen azt mutatják, hogy egy országon belül a növekvő életszínvonal a lakosság számához viszonyítva aránytalanul nagyobb mértékű energiafogyasztással jár együtt. A legegyszerűbb közelítő összefüggés szerint a GDP nyolckilencszeres növekedése - például az 1000 dollár/fő/év szintről a 9000 dollár/fő/év jövedelem elérése - az olajszármazékok iránti keresletet durván tizennégyszeresére emeli. Ez a tény világosan érthető: a gazdasági növekedés motorja az ipar és az ezzel járó árumozgatás, de emellett a nagyobb lakásokban többet világítunk és fűtünk (vagy éppen hűtünk), a kerékpárt először motorra, majd autóra cseréljük, ha javul az infrastruktúra, távolabbi munkahelyekre ingázunk és így tovább.

```
</p>
</body>
</html>
```

Az eredmény

Demográfiai változások

Időközben az általános jólét, szokásos módon az egy főre jutó nemzeti össztermék- (GDP-) egységekben kifejezve, szintén növekedett (persze óriási egyenlőtlenségekkel).



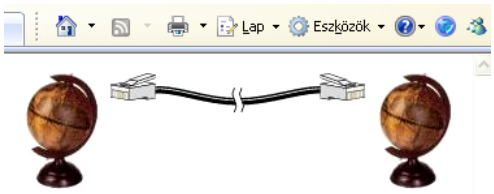
Az empirikus elemzések pedig egyértelműen azt mutatják, hogy egy országon belül a növekvő életszínvonal a lakosság számához viszonyítva aránytalanul nagyobb mértékű energiafogyasztással jár együtt. A legegyszerűbb közelítő összefüggés szerint a GDP nyolckilencszeres növekedése - például az 1000 dollár/fő/év szintről a 9000 dollár/fő/év jövedelem elérése - az olajszármazékok iránti keresletet durván tizennégyszeresére emeli. Ez a tény világosan érthető: a gazdasági növekedés motorja az ipar és az ezzel járó árumozgatás, de emellett a nagyobb lakásokban többet világítunk és fűtünk (vagy éppen hűtünk), a kerékpárt először motorra, majd autóra cseréljük, ha javul az infrastruktúra, távolabbi munkahelyekre ingázunk és így tovább.

Megjegyzések a kép igazításához

- A szabvány szerint ellenjavalt megoldás a képnek vízszintesen az ablak közepére igazítása a `<CENTER>` címkével, illetve a `<p>` címke *align* paraméterének használatával.
- Kép és szöveg változatos elrendezései hozhatók létre, ha táblázatot is használunk, de erről majd később.
- Ha nincs szükség a kép körülfolyatására, tegyük a képet a `<p>` címkével külön bekezdésbe.
- Szóba jöhet a `<br />` címke `clear` paramétere, ha egy ponttól meg akarjuk gátolni, hogy a maradék szöveg a kép mellé kerüljön!
- Egymás mellé több kép is helyezhető, ha ugyanazzal az igazítással helyezzük el a képeket.

Példa:

```
<img src="pics\globus.jpg" align="right" />
<img src="pics\kabel.jpg" align="right" />
<img src="pics\globus.jpg" align="right" />
```



`hspace` és `vspace` – térköz a kép körül

Nem szép, ha a kép és a környezetében lévő szöveg, vagy a mellette lévő másik kép egymásba érnek. Szebb és áttekinthetőbb az oldal, ha a kép és környezete között térközt hagyunk.

A `hspace` paraméterrel a kép mellett balra és jobbra, a `vspace` paraméterrel a kép fölött és alatt állíthatjuk be a térközt. Értéküket képpontban mérve kell megadni. Példa egy kódrészlettel

```
<body>
```
René Descartes (lat.: Cartesianus, 1596-1650.) francia filozófus, természetkutató és matematikus.
```
<img src="pics/Descartes.gif" align="left" />
```
Vele kezdődik az újkori filozófia antropológiai fordulata, valamint ő volt az analitikus geometria egyik megalapítója. Műveiben az egzakt természettudományok eredményeit és a matematika módszereit alkalmazta. Filozófiájában a szubjektum erős hangsúlyozása és a lehető legnagyobb bizonyosságra való törekvés a jellemző.
```
<br clear="left" />
```
René Descartes (lat.: Cartesianus, 1596-1650.) francia filozófus, természetkutató és matematikus.
```
<img src="pics/Descartes.gif" align="left" hspace="20" vspace="20" />
```
Vele kezdődik az újkori filozófia antropológiai fordulata, valamint ő volt az analitikus geometria egyik megalapítója.
Műveiben az egzakt természettudományok eredményeit és a matematika módszereit alkalmazta. Filozófiájában a szubjektum erős hangsúlyozása és a lehető legnagyobb bizonyosságra való törekvés a jellemző.
```
</body>
```

A böngészőben megjelenő eredmény:

René Descartes (lat.: Cartesianus, 1596-1650.) francia filozófus, természetkutató és matematikus. Vele kezdődik az újkori filozófia antropológiai

fordulata, valamint ő volt az analitikus geometria egyik megalapítója. Műveiben az egzakt természettudományok eredményeit és a matematika módszereit alkalmazta. Filozófiájában a szubjektum erős hangsúlyozása és a lehető legnagyobb bizonyosságra való törekvés a jellemző.

René Descartes (lat.: Cartesianus, 1596-1650.) francia filozófus, természetkutató és matematikus. Vele kezdődik az újkori filozófia antropológiai

fordulata, valamint ő volt az analitikus geometria egyik megalapítója. Műveiben az egzakt természettudományok eredményeit és a matematika módszereit alkalmazta. Filozófiájában a szubjektum erős hangsúlyozása és a lehető legnagyobb bizonyosságra való törekvés a jellemző.

A megadott térközöket mindkét irányban szimmetrikusan állítják be a böngészők. Nincs mód arra, hogy pl. a kép két oldalán különböző méretű térközöket állítsunk be. (Pontosabban ez csak stíluslappal lehetséges.)

`border` – a kép keretének vastagsága hiperhivatkozásban

Hiperhivatkozás készítéséről később lesz szó, itt csak egy vonatkozásban érintjük ezt a témát. *Ha egy képből készítünk hiperhivatkozást*, akkor a böngészők 2 képpont vastagságú színes keretbe teszik a képet, jelezve az olvasónak, hogy a kép egyben hiperhivatkozás is. Az `<img>` címke `BORDER` paraméterével képpontokban mérve megadhatjuk ennek a keretnek a vastagságát. *A zérus érték letiltja a keretezést.*

6. Hiperhivatkozás készítése

A *hiperhivatkozás* utalás egy erőforrásra. Ez az utalás egyben kapcsolatot hoz létre az adott dokumentum és a hivatkozott erőforrás között a böngésző illetve a webszerver közvetítésével. A kapcsolat révén a hivatkozott erőforrást a felhasználó elérheti, ha rákattint a hivatkozásra a böngészőben. Az erőforrás rendszerint egy másik HTML dokumentum, de lehet egy kép, egy PDF dokumentum, vagy éppen az aktuális dokumentum egy másik részlete.

A hivatkozott erőforrás (fájl) típusától függ, hogy elérése után mi történik. Ha fájl tartalmát maga a böngésző nem képes megjeleníteni, akkor elindíthat egy alkalmazást, amelyet erre a célra előre beállítottak a böngészőben, vagy felkínálja a választást a felhasználónak, hogy a letöltött fájlt háttértárra *mentse*, vagy az operációs rendszerben társított alkalmazással *megnyissa*.[3]

6.1 Az <a> hivatkozás elem és legfontosabb paraméterei

A hiperhivatkozás készítéséhez az `<a>` (*anchor*=horgony) címkét használjuk. A hivatkozás részletei az `<a>` és a `</a>` címkék között helyezkednek el. Részleteken kétféle dolog értendő:

- *paraméterek*: ezeket az általános szabály szerint a nyitó `<a>` tagban helyezzük el.
- *az a szöveg és/vagy kép*, amelyre az olvasó kattintva elindíthatja a hivatkozott erőforrás elérését.
  Annak érdekében, hogy jól észrevehető legyen, a hivatkozásban szereplő szöveg a környezettől eltérő módon jelenik meg, rendszerint aláhúzva és kék színű betűkkel. Természetesen a `<body>` elem `link`, `vlink` `alink` paramétereivel a tanultak szerint a hivatkozás színei be is állíthatók. Kép esetében böngészők keretezéssel jelzik, hogy a kép hiperhivatkozást jelöl.

Az `<a>` és `</a>` közötti szakasz tartalmazhat szöveget, sortörést, címsor formázást és képeket. A böngésző ezeket a szokott módon jeleníti meg, de a hiperhivatkozást jelző, már említett formázásokat is elvégzi.

Az `<a>` elem fontosabb paraméterei:

`href` - a hivatkozott erőforrás URL-je

Az URL fogalmát, használati módját korábban megtárgyaltuk (lásd az *5.2 Képek* című részt) Ehhez hamarosan, a könyvjelző kapcsán egy kis kiegészítést teszünk. Példa:

```
<a href="http://info.lovassy.hu/stat.html">Statisztikai adatok</a>
```

`name` – könyvjelző neve[4]

A hiperhivatkozást nemcsak egy másik dokumentumra mutathat, hanem az adott dokumentum egy bizonyos részére is. Ekkor egy *névvel* meg kell jelölnünk azt a pontot a dokumentumban, amelyre

[3] Lásd például az Internet Explorer esetében az Eszközök\Internetbeállítások\Programok ablakot.
[4] A `NAME` helyett az `ID` paraméternév is használható.

hivatkozni fogunk (nevezzük ezt *könyvjelzőnek*, az angol szakkifejezés *named anchor*), majd el kell készíteni magát a hiperhivatkozást.

A könyvjelző egy általában rövid szöveg. A dokumentumon belül ugyanazt a szöveget (nevet) nem használhatjuk fel egy másik pont megjelölésére. Az alábbi példa mutatja, hogy ilyen esetben két különböző szerepkörben használjuk az `<a>` címkét.

Az *antik.html* dokumentumban egy hely megjelölése:

```
<h2><a name="Preszókratikusok">A Szókratész előtti filozófia</a></h2>
```

Hivatkozás erre a helyre relatív URL használatával:

```
<a href="antik.html#Preszókratikusok">Preszókratikus filozófia</a>
```

Mint látható, a hivatkozásban a könyvjelző elé # jelzést kell tenni.

`target` – a megjelenés helye

Alapértelmezés szerint a hivatkozott oldal a böngésző aktív ablakában jelenik meg, lecserélve annak az aktuális tartalmát. A *target* paraméter használatával előírhatjuk, hogy hol jelenjen meg a hivatkozott tartalom. A *target* lehetséges értékeit a táblázat mutatja. (Ügyeljünk az aláhúzás jelre!)

Érték	Magyarázat
`_blank`	A hivatkozott dokumentumot új ablakban nyitja meg.
`_self`	A hivatkozott dokumentumot ugyanabban a keretben (frame) nyitja meg, amelyben a link volt.
`_parent`	A hivatkozott dokumentumot a szülő keretkészleben (frameset) nyitja meg.
`_top`	A hivatkozott dokumentumot a böngésző főablakában nyitja meg, minden ott lévő keretet (frame) lecserélve.
`name`	A hivatkozott dokumentumot a megadott nevű ablakban nyitja meg.

Példa a hivatkozott dokumentum új ablakban történő megnyitására:

```
<a href="nevsor.html" target="_blank">Névsor</a>
```

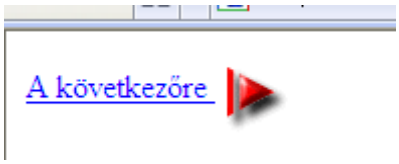
6.2 Példák hiperhivatkozásokra

Képek a hivatkozásokban

Az `<a>` és `</a>` címkék között elhelyezett képek teljes egészükben az adott hivatkozás „érzékeny” részévé válnak, tehát rájuk kattintva elindul a hivatkozott erőforrás elérési folyamata.[5] Lássunk egy példát!

```
<a href="fejezet2.html">A következőre <img align="middle" border="0"
src="pics/next.gif" /></a>
```

Eredménye:



Mind a szöveg, mind a kis háromszög alakú nyíl érzékeny elem, bármelyikre kattintva elindul a *fejezet2.html* dokumentum letöltése. Természetesen nem kötelező a kép mellé szöveget is tenni, ha a kép jelentése egyértelmű. Például

```
<A HREF="fejezet2.html"><img src="pics/next.gif" /></a>
```

Eredménye a böngészőben:

[5] A képeket gyakran ettől eltérően használják hiperhivatkozásokban. Különböző területei más és más hivatkozáshoz tartoznak, így attól függően, hogy a felhasználó hová kattint a képen, más és más dokumentumokat érhet el. Az ilyen képeket az angol szaknyelv *imagemap*-nek nevezi.

Itt jól látható a hivatkozást jelző kép keretezése, amit az előző példában a `border="no"` paraméterrel letiltottunk.

Hivatkozás HTTP protokollal

Mint az URL fogalmánál korábban már szóltunk róla, ha a hivatkozott dokumentum egy webszerveren helyezkedik el, akkor http protokollal érhetjük el. Példák a leggyakrabban előforduló esetekre:

```
<a href="http://www.info.hu/tartalom.html">Tartalomjegyzék</a>
```

A hivatkozás a *www.info.hu* nevű gépen működő webszerver gyökérkönyvtárában a tartalom.html dokumentumra mutat.

```
<a href="http://www.info.hu/doc/tartalom.html">Tartalomjegyzék</a>
```

Ugyanaz, mint az előző, de a webszerver gyökérkönyvtárából nyíló *doc* könyvtárban van a hivatkozott dokumentum.

Az alapértelmezett dokumentum

Példa:

```
<a href="http://www.info.hu">További információk</a>
```

A hivatkozás nem nevezi meg a *www.info.hu* szervertől kért dokumentumot. Ilyen esetben a webszerver az ún. nyitó oldalt küldi el válaszként, amely *többnyire* a webszerver gyökérkönyvtárában lévő *index.html* nevű dokumentum. Az alapértelmezett dokumentum neve azonban a webszerver beállításától függ. Szokásos nevek még: index.htm, default.htm, PHP szkripttel előállított oldal esetén *index.php*.

A port megjelölése

Az imént vett példákban nincs megjelölve, hogy a webszerver mely porton fogad kapcsolatokat. Alapértelmezés szerint ez a 80-as port. Ha a szóban forgó webszerver más portot használ, akkor az URL-ben ezt fel kell tüntetni. Például:

```
<a href="http://www.info.hu:8080/doc/tartalom.html">Tartalomjegyzék</a>
```

A böngészőben megjelenő eredmény:

Relatív URL használata

Az URL fogalmánál erről is volt szó, ezért csak röviden a lényeg: ha a dokumentumban relatív URL-t használunk, akkor az a magának a dokumentumnak a helyétől kiindulva adja meg az elérési utat és a fájlnevet.

Tegyük fel, hogy a *http://www.info.hu/doc/tartalom.html* dokumentum tartalmazza a következő hiperhivatkozást:

```
<a href="../fejezetek/bevezetes.html">Bevezetés</a>
```

Ami azt jelenti, hogy az aktuális (tehát a *doc*) könyvtár szülőkönyvtárából (tehát a gyökéből) nyíló *fejezetek* könyvtárban van a hivatkozott *bevezetes.html* nevű dokumentum.

A relatív URL-t úgy is felfoghatjuk, hogy a szerver az URL elejéről „hiányzó” adatot a dokumentum URL-jében szereplő adatokból pótolja. Az előző példában szereplő dokumentumra ezért relatív módon így is hivatkozhatunk:

```
<a href="/fejezetek/bevezetes.html">Bevezetés</a>
```

Az elérési út elején a / jel a gyökérkönyvtár jele. Most a hivatkozást tartalmazó dokumentum URL-jéből a *http://www.info.hu* adattal egészíti ki a szerver a relatív URL-t, hogy megkapja a dokumentum helyét.

Hiperhivatkozás a dokumentum egy megadott helyére

Mint korábban láttuk egy ilyen hivatkozás csak akkor működhet, ha a hivatkozott dokumentumban a NAME paraméter használatával meg van jelölve az a hely, amelyre hivatkozást készítünk. Nézzünk a leggyakrabban előforduló eseteket!

Az egyik eset az, amikor *a hivatkozott hely egy másik dokumentumban van*. Ekkor a helyzettől függően abszolút vagy relatív hivatkozással meg kell adni a dokumentumot és abban a hivatkozott pont nevét.

Példaként tegyük fel, hogy a http://www.info.hu/fejezetek/bevezetes.html dokumentumban egy szövegrész a következő módon van megjelölve:

```
<a name="kezdet">Bevezetés</a>
```

Tegyük fel továbbá, hogy a http://www.namiujsag.hu/tovabbi.html dokumentumban kell a hivatkozást elkészíteni. Ekkor a hivatkozás ehhez hasonló lesz:

```
<a href="http://www.info.hu/fejezetek/bevezetes.html#kezdet">A cikk
bevezetése</a>
```

A vizsgált eset egy változataként most tegyük fel, hogy a http://www.info.hu/doc/tartalom.html dokumentumban készül a hivatkozás. Ekkor relatív hivatkozást célszerű használni, például így:

```
<a href="../fejezetek/bevezetes.html#kezdet">A cikk bevezetése</a>
```

A másik eset az, amikor a hivatkozás a dokumentumon belül elhelyezkedő részre történik. Az előző esetet módosítva most magán a http://www.info.hu/fejezetek/bevezetes.html dokumentumon belül hivatkozunk a megjelölt részre. Nyilván a relatív hivatkozás logikája szerint most elegendő csak a könyvjelző nevét megadni:

```
<a href="#kezdet">A cikk bevezetése</a>
```

Ezt a megoldást gyakran alkalmazzák akkor, ha az oldal túl hosszú (ez egyébként kerülendő) és témáját tekintve több részből áll. Ilyenkor az oldal elején rendezetlen listaként készítenek egy tartalomjegyzéket az oldalról, a lista elemei pedig a megfelelő részre hivatkoznak. Példa:

```
<h2>Tartalom</h2>
<ul>
<li><a href="#bevezet">Bevezetés</a>
<li><a href="#tortenet">Az internet története</a>
<li><a href="#felepit">Az internet felépítése</a>
</ul>

<h2 name="bevezet">Bevezetés</h2>
Itt következik a bevezetés szövege. …

<h2 name="tortenet">Az internet története</h2>
Itt következik az internet történetéről szóló szöveg. …

<h2 name="felepit">Az internet felépítése</h2>
Itt következik az internet felépítéséről szóló szöveg. …
```

A példa továbbfejleszthető. Ha az egyes részek szövegének végére is készítünk hiperhivatkozást, ami a tartalomjegyzékre mutat, akkor az olvasó a rész elolvasása után könnyen visszaléphet a tartalomjegyzékhez.

Hivatkozások listája

Az előző feladatban hivatkozásokból álló listát készítettünk. Hasonló megoldásokat gyakran tartalmaznak a weblapok. Előfordul azonban, hogy az egyes hivatkozásokhoz hosszabb magyarázó

szöveg is tartozik. Ekkor természetesen nem az egész szöveget, hanem csak egy megfelelő részét foglaljuk bele a hivatkozásba. Például így:

```
<ul>
<li>A dokumentum a <a href="#bevezet">Bevezetés</a>című fejezettel kezdődik, amelyben a szerző vázolja a tárgyalt problémákat és ismerteti célkitűzéseit.
.
.
</ul>
```

Hivatkozások egyéb protokollokkal

Az alábbiakban néhány további példával szemléltetjük azoknak a protokolloknak a használatát, amelyekről már részletesen volt szó az URL fogalmával kapcsolatban (Lásd a *Képek* fejezetben *Az URL fogalma* című részt!).

Elektronikus levél írásának kezdeményezése

```
<a href="mailto:jozsi@freemail.hu">Email Józsinak</a>
```

Egy fájl letöltése FTP protokollal

```
<a href="ftp://shop.info.hu/pub/arak.xls">Árjegyzék</a>
```

Az adott fájl elérését a böngésző anonymous felhasználóként kísérli meg. Ha felhasználói fiókhoz kötődik az elérhetőség, akkor megadhatjuk a felhasználói nevet is, pl.:

```
<a href="ugyfel:ftp://shop.info.hu/pub/arak.xls">Árjegyzék</a>
```

Ha a bejelentkezési adatok közül (felhasználói név, jelszó) valamelyik hiányzik, vagy hibás, akkor a böngésző egy ablak megjelenítésével bekéri a felhasználótól.

Mivel a többi protokoll az érettségi feladatokban csekély valószínűséggel fordul elő, nem foglalkozunk többet velük.

7. Táblázatok

A táblázatokat dokumentumokban többnyire adatok rendezett, jól áttekinthető bemutatására használják. A weblaptervezők azonban hamar rájöttek arra, hogy a táblázatok felhasználhatók az oldalak elrendezésének változatosabbá tételére is. Például szegély nélküli táblázatokkal kialakíthatók többhasábos oldalak, lapszéli kommentárok. Stílusok használata nélkül *kizárólag táblázatokkal lehet az oldal tetszőleges részére szöveget, képet elhelyezni*, illetve szöveget és képet szabadon kombinálni.

A HTML szemlélete szerint *a táblázat sorokból, a sor pedig cellákból áll*. Az oszlop fogalma nem játszik szerepet a táblázat definiálásában, a táblázat oszlopai egyszerűen létrejönnek a sorok és cellák segítségével elvégzett leírásunk alapján.

7.1 Táblázat definiálása

7.1.1 A táblázat egészének tulajdonságai

A táblázat kezdetét a `<table>`, a végét a `</table>` címke jelzi. A táblázat egészére vonatkozó adatokat a `<table>` paramétereiként adhatjuk meg.

Hamarosan láthatjuk, hogy a sorok és a cellák számára hasonló paramétereket adhatunk meg, mint a táblázat egészére, ezért már most hangsúlyozni kell, hogy *a táblázat egy része számára megadott értékek felülbírálják a nagyobb egységre beállított tulajdonságokat*. Így a táblázat egésze számára beállított tulajdonságot felülbírálja a sorra beállított érték, a sor beállításait pedig a celláé.

A `<table>` paraméterei a következők:

`width` – a táblázat szélessége

Értékét megadhatjuk képpontokban számolva, vagy százalékban. Ez utóbbi esetben az érték azt adja meg, hogy a böngészőablak aktuális méretének hány százaléka legyen a a táblázat szélessége. Például a `width=400` paraméter hatására egy 400 pixel szélességű táblázatot hoz létre a böngésző, a méretbe beleértve a szegélyek méretét és a cellák közötti távolságokat is. A `width=50%` paraméterrel az ablakméret felének megfelelő szélességű lesz a táblázat.

Ha a szélességet nem adjuk meg, akkor a böngésző cellákban lévő adatok szélességének összegére méretezi a táblázatot, hogy minden adat elférjen. Ha megadjuk a szélességet, de a táblázat tartalma ebben a méretben nem fér el, akkor a `width` értékét a böngésző figyelmen kívül hagyja.

`border` – a szegély vastagsága

A táblázatot és celláit körülvevő szegély vastagságának értékét pixelekben mérve adhatjuk meg. A 0 érték hatására nem rajzol szegélyt. A *border* kulcsszó érték nélkül 1 pontos szegélyt jelent.

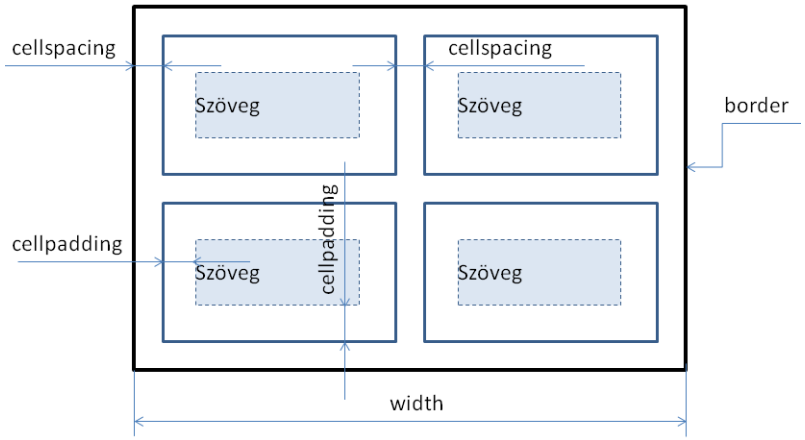
`cellspacing` – a cellák közötti távolság

Ez a paraméter határozza meg a szomszédos cellák közötti képpontokban mért távolságot, továbbá a táblázat külső éle mentén kihagyott térközt. Alapértelmezésben – tehát a *cellspacing* megadása nélkül – a böngészők 2 pixelnek veszik.

`cellpadding` – a cellamargó

A cella tartalma és a cella széle közötti pixelekben mért távolság. *Alapértelmezett értéke 1.*

A <table> paramétereinek értelmezése



`align` – a táblázat helyzetének igazítása

Alapértelmezésben a táblázatot a böngészők az ablakuk bal margójára igazítják. Az *align* paraméterrel ezt megváltoztathatjuk. Lehetséges értékei: *`left`*, *`right`*, *`center`*. Jelentésük értelemszerű.

Természetesen a táblázat igazítása befolyásolható azzal, hogy megfelelő igazítású szakaszba (lásd `<div>` címke) tesszük.

`bgcolor` – a táblázat háttérszíne

A táblázat összes cellájának háttérszínét állíthatjuk be segítségével. A többi szín paraméterhez hasonlóan RGB színkóddal vagy szabványos színnévvel adható meg a kívánt szín.

`background` – a táblázat hátterét alkotó kép

Nem szabványos paraméter, de a legtöbb elterjedt böngésző képes értelmezni. Segítségével egy képet helyezünk *a táblázat egészének* hátterébe. A paraméter értéke természetesen egy URL, amely a háttérbe kerülő képre mutat. Ha a kép nagyobb a táblázatnál, a böngésző a fölös részt levágja.

7.1.2 A táblázat sorai

A `<table>` és `</table>` címkéken belül kell leírni a táblázat sorait. Egy sor leírása a `<tr>` `</tr>` címkék között helyezkedik el. A sor tulajdonságait a `<tr>` címke következő paramétereivel állíthatjuk be:

`align` – a sor celláiban a szöveg vízszintes igazítása

Lehetséges értékei: `left`, `right`, `center`.

`valign` – a sor celláiban a szöveg függőleges igazítása

Lehetséges értékei: `top`, `middle`, `bottom`. Mivel a böngészők alapértelmezésben a cellák tartalmát függőlegesen középre helyezik, a `middle` érték magát az alapértelmezést jelöli.

`bgcolor` – a sor háttérszíne

A sor összes cellájának háttérszínét állíthatjuk be segítségével.

7.1.3 A sor cellái

A `<tr>` címkét követően írhatjuk le az adott sort alkotó cellákat. Egy cella leírása a `<td>` és `</td>` címkék között helyezkedik el. A cella tulajdonságait a `<td>` következő paramétereivel állíthatjuk be:

`width` – a cella szélessége

A `<table>` címke `width` paraméteréhez hasonlóan képpontokban vagy százalékban adhatjuk meg a cella szélességét. A százalékos érték az egész tábla szélességéhez mérten adja meg a cellaszélességet. Néhány tudnivaló:

- Ha a táblázat adott oszlopában egy cellára beállítjuk a szélességet, a böngészők e cella oszlopában a többi cellát is ugyanilyen szélességűre formázzák. Ezért a kód egyszerűsítése érdekében célszerű csak a legfelső cella szélességét előírni a `width` paraméterrel.
- Ha egy oszlopban több cellára is van `width` előírva, akkor közülük a legnagyobb értékű szabja meg az oszlop összes cellájának szélességét.
- Ha a cella megadott szélességében az adott tartalom nem fér el, akkor a böngésző a szükséges méretűre bővíti a cellát, illetve ezzel együtt az oszlopot.

`height` – a cella magasság

E paraméterrel megadhatjuk a cella minimális magasságát pixelekben mérve. Mivel a sor összes cellája ugyanolyan magasságú lehet csak, elegendő egy cella – célszerűen a sor első cellája – magasságát megadni. Ha a cella tartalma nem fér el az adott magasságú cellában, akkor a böngésző a szükséges magasságúra bővíti a cellát, s ezzel együtt a sort.

`align` – a cellában a szöveg vízszintes igazítása

Lehetséges értékei: `left`, `right`, `center`. (Ugyanaz. mint a `<tr>` címkénél.)

`valign` – a cellában a szöveg függőleges igazítása

Lehetséges értékei: `top`, `middle`, `bottom`. (Ugyanaz. mint a `<tr>` címkénél.)

`bgcolor` – a cella háttérszíne

A cella háttérszínét állíthatjuk be segítségével. (Ugyanaz. mint a `<tr>` címkénél.)

`background` – a cella hátterét alkotó kép

Nem szabványos paraméter, de a legtöbb elterjedt böngésző képes értelmezni. Segítségével egy képet helyezünk *a cella* hátterébe. A paraméter értéke természetesen egy URL, amely a háttérbe kerülő képre mutat.

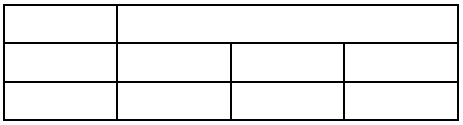
`nowrap` – sortörés tiltása a cellában

A böngésző alapértelmezésben a cella szövegét több sorba tördeli, ha nem fér el egy sorba. A NOWRAP ezt letiltja és a cella szövege egyetlen sorba kerül, hacsak a `<br>` vagy `<p>` címkékkel nem kényszerítünk ki sor- illetve bekezdéstörést.

`colspan` – oszlopátfogás (vízszintes cellaegyesítés)

Ezzel a paraméterrel adhatjuk meg, hogy a cella hány oszlopot fogjon át, azaz hány oszlop szélességre terjedjen ki. Szövegszerkesztőkben megszokott művelet az egymás melletti cellák egyesítése. A `colspan` segítségével hasonló táblázat-szerkezetet alakíthatunk ki egy HTML dokumentumban is.

Például a `<td colspan="3">` egy olyan cellát ír le, amely 3 oszlop szélességű mint teszem fel az alábbi táblázat első sorának második cellája.

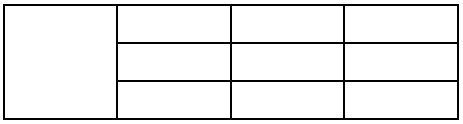


Ügyeljünk arra, hogy abban a sorban, amelyben valamely cellára COLSPAN-t alkalmazunk, a definiált cellák számát megfelelő mértékben csökkentsük. Ha a kelleténél több cella van a *collspan*-t tartalmazó cella sorában, akkor az előírtnál kevesebb oszlopot fog át a cella.

`rowspan` – sorátfogás (függőleges cellaegyesítés)

Ezzel adhatjuk meg, hogy a cella hány sort fogjon át, azaz hány cellasor magasságra terjedjen ki. Szövegszerkesztőkben megszokott művelet az egymás alatti cellák egyesítése. A `rowspan` segítségével hasonló táblázat-szerkezetet alakíthatunk ki egy HTML dokumentumban is.

Például a `<td rowspan="3">` egy olyan cellát ír le, amely 3 sor magasságú, mint például az alábbi táblázat első oszlopának első cellája.



Itt is ügyeljünk arra, hogy azokban a sorokban, amelyeket a *rowspan*-t tartalmazó cella átfog, eggyel több cella lesz. Tehát pl. az ábrán látható táblázat 2. és 3. sorában csak 3-3 cellát kell létrehozni az 1. sorból „lenyúló” cella miatt.

7.1.4 Fejléc cellák

A `<th>` címke ugyanúgy cellák létrehozására alkalmas, mint a `<tr>`, ugyanazok a paraméterei is. A különbség az, hogy a `<th>`-val létrehozott cella tartalmát a böngésző középre igazítva félkövér szedéssel jeleníti meg. Ezért az oszlopfeliratokat tartalmazó cellák létrehozására használjuk.

7.1.5 A táblázat címe

Egy táblázatnak címet is adhatunk a <caption> címkével. A `<caption>` elemet a `<table>` után kell elhelyezni a táblázat szerkezetének definiálásakor. Például

```
<table border="2">
  <caption align=”bottom”> Ide kerül a táblázat címe </caption>
  <tr>
    <td>  Itt lesznek a táblázat sorai és cellái</td>
  </tr>
</table>
```

A caption egyetlen paramétere az **`align`**, amellyel megadhatjuk, hogy a cím a táblázat fölé (*top*) vagy alá (*bottom*) kerüljön. A *top* az alapértelmezett.

7.1.6 Néhány további lehetőségről

A HTML 4.01 illetve az XHTML 1.0 szabvány bevezetett néhány olyan elemet, amellyel rendkívül összetett szerkezetű táblázatok alakíthatók ki és az oszlop fogalma is értelmezhetővé válik.

Lehetővé teszik a táblázat oszlopainak (`<colgroup>, <col>`) csoportosítását, valamint a sorok csoportosításával a táblázat fej (`<thead>`), törzs (`<tbody>`) és láb (`<tfoot>`) szakaszokra osztását. Ezek részletesebb ismertetése szükséges lenne, de túlmenne terjedelmi kereteineken.

7.2 Példák táblázatok létrehozására

1. feladat

Hozzuk létre a következő táblázatot!

Hónap	Forgalom (ezer Ft)	
	Fehér Holló Étterem	Édes Élet Bár
Január	31000	23000
Február	35000	26000
Március	41000	21000

A táblázat középre igazított, szélessége az ablak 60%-a. A fejléc cellák háttérszíne *silver*, azaz `#c0c0c0`, a szegély vastagsága 1 pixel, a cellamargó 5 képpont.

Megoldás

```
<table width=60% align="center" border cellpadding="5">
   <tr bgcolor="silver">
    <th rowspan="2">Hónap</th>
    <th colspan="2">Forgalom (ezer Ft)</th>
   </tr>
   <tr bgcolor="silver">
    <th>Fehér Holló Étterem</th>
    <th>Édes Élet Bár</th>
   </tr>
   <tr>
    <td>Január</td>
    <td align="center">31000</td>
    <td align="center">23000</td>
   </tr>
   <tr>
    <td>Február</td>
    <td align="center">35000</td>
    <td align="center">26000</td>
   </tr>
   <tr>
    <td>Március</td>
    <td align="center">41000</td>
    <td align="center">21000</td>
   </tr>
</table>
```

2. feladat

Valósítsunk meg kéthasábos elrendezést táblázat alkalmazásával! Legyen a szövegrész háttere sárga (yellow), a hasábokat 20 pontos cellák közötti térközzel válasszuk el. Minta:

A stíluslapok bevezetésével a formázásra szolgáló régi nyelvi elemek feleslegessé váltak. De nyilvánvalóan idő kell, amíg a böngészők és weblap-szerkesztők szoftvereit átdolgozzák a szabvány új ajánlásainak megfelelően, másrészt a weblapokat forráskódban fejlesztők is megtanulják az új megoldásokat.

Ezért a HTML 4.01 szabványnak három változatát dolgozták ki. A szigorú (Strict) változatból kihagyták a régebbi formázó elemeket, amelyeket stíluslapok hivatottak helyettesíteni. (Pl. FONT címke vagy az ALIGN paraméter.) A szabvány másik, ún. átmeneti (Transitional) változata még megengedi, hogy a régi, de már kerülendő nyelvi elemeket is használja egy dokumentum.

Megoldás:

```
<table border="0" cellspacing="20" bgcolor="yellow">
  <tr valign="top">
    <td width="50%">Egyik hasáb szövege ....</td>
    <td>Másik hasáb szövege ... </td>
  </tr>
</table>
```

Mivel alapértelmezés a szöveg függőlegesen középre igazítása, ezt meg kellett változtatni a `valign` paraméterrel. Hátránya ennek a többhasábos megoldásnak, hogy nekünk kell kiválasztani, hogy a szöveg mely része kerüljön egyik illetve másik hasábba.

3. feladat

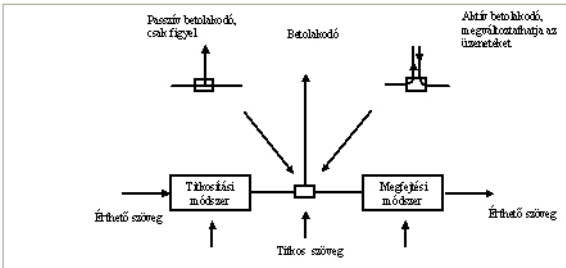
Figyeljük meg az alábbi képen látható változó hasábszámú elrendezést! Segítségképpen 1 pontos szegélyt rajzoltattunk, ami természetesen elmarad a végleges változatban. Valósítsuk meg HTML kóddal ezt az elrendezést, beleértve a szükséges szövegformázást is!

Titkosítás

A számítógép megjelenéséig a titkosító eljárások legnagyobb korlátját a kódoló emberek képességei határolták be. További korlát a titkosító módszerek gyors változtatásában jelentkezett, ez ugyanis nagyon sok ember újbóli kiképzését vonta maga után. A rejtjelezés megfejtésének problémáját kriptoanalízisnek, rejtjelezés tervezését és megfejtését együttesen titkosítástannak vagy kriptológiának nevezik. A titkosítás alapvető szabálya az, hogy a tikosítás készítőjének feltételeznie kell, hogy a megfejtő ismeri a titkosítás általános módszerét.

Más szavakkal: a feltörő ismeri az 1. ábrán látható titkosítási módszert. Ekkor jön a kulcs szerepének a fontossága. A kulcs rendszerint rövid karakterfüzér, amely a lehetséges több titkosítás közül kiválasztja az aktuálist.

Ellentétben az általános módszerrel, mely csak ritkán cserélhető, a kulcsot olyan gyakran lehet cserélni, ahogyan csak szükséges.



1. ábra A titkosítási modell

Megoldás (a szöveg illetve a kép tartalma közömbös):

```
<table border="1" cellspacing="7">
  <tr>
    <th colspan="3"><h2>Főcím</h2></th>
  </tr>
  <tr valign="top">
    <td width="33%" rowspan="2">Az első hasáb szövege ...</td>
    <td width="33%">A második hasáb szövege ...</td>
    <td >A harmadik hasáb szövege ...</td>
  </tr>
  <tr>
    <td colspan="2" align="center"><img src="filename.jpg" />
    <p><i>Képfelirat</i></p></td>
  </tr>
</table>
```

4. feladat

Figyeljük meg az alábbi oldalcímes elrendezést! Segítségképpen ismét 1 pontos szegélyt rajzoltattunk, ami maradjon el a végleges változatban! Valósítsuk meg HTML kóddal ezt az elrendezést, beleértve a szükséges szövegformázást illetve tagolást is!

Helyettesítéses rejtjelezés	A helyettesítéses rejtjelezési módszerekben az egyes betűket vagy betűcsoportokat valamilyen másik betűvel vagy betűcsoporttal helyettesítik. A hagyomány Julius Caesar-nak tulajdonítja a legréebbi rejtjelet. Ebben a módszerben az a betűt D, a b-t az E, a c-t az F, ... és a z-t C helyettesíti. Például az (angol) attack szó DWWFDN-né válik
Felcseréléses rejtjelezés	A felcseréléses rejtjelezések megváltoztatják a betűk sorrendjét. A következő példa az oszlop szerint felcserélést példázza. A rejtjelezésnek egy olyan szó vagy kifejezés a kulcsa, amelyben nincs két azonos betű. A kulcs az oszlopokat számozza oly módon, hogy az 1. oszlop alatt az e betű helyezkedik el, amelyik az ábécé kezdetéhez legközelebb esik. A titkosított szöveget oszlopokban kell kiolvasni, azzal az oszloppal kezdve, amelynek kulcsszáma a legkisebb.

Megoldás üres cellaoszlop kialakításával (a cellák szövege itt sem fontos, csak a szerkezet):

```
<table>
  <tr>
    <th width="20%" align="right">
      <h3>Első cím</h3>
    </th>
    <td>
    </td>
    <td>
      Az első címhez tartozó szöveg ...<p>
    </td>
  </tr>
  <tr>
    <th align="right">
      <h3>Második cím</h3>
    </th>
    <td>
    </td>
    <td>
      A második címhez tartozó szöveg ...
    </td>
  </tr>
</table>
```

8. Stíluslapok

Ebben a fejezetben áttekintjük a korszerű és igényes megjelenésű weblapok készítéséhez nélkülözhetetlen stíluslapok használatának alapvető tudnivalóit. Hangsúlyozni kell azonban, hogy

ez az áttekintés csak a kezdők számára leglényegesebb ismeretekre tér ki. Ha valaki a stíluslapokban rejlő lehetőségeket valóban ki akarja használni, akkor ehhez további forrásokat kell keresnie. Szerencsére az interneten bőségesen találhat ilyeneket.

Az 1. fejezetben említettük, hogy a HTML régi, tehát a 4.xx verziókat megelőző változatainak számos hátránya és hiányossága volt. Ezek közül a legfontosabbak a következők:

- A megjelenítendő tartalomba belekeverednek a formázási előírások, ezért a dokumentum forráskódja egy összetett oldal esetében már nehezen áttekinthető.
- Ha azt akarjuk, hogy egy részlet ugyanúgy nézzen ki, mint a dokumentum egy másik részlete, akkor az egyiknél alkalmazott formázási előírásokat a dokumentum másik helyén is pontosan ugyanúgy meg kell ismételni. Elég egy kis eltérés és a szándékaink szerint azonos stílusú részletek megjelenésükben különbözőek lesznek.
- A dokumentum egyes részeinek elhelyezésében és formázásában a nyomdai termékekhez vagy akárcsak a szövegszerkesztővel készült dokumentumokhoz képest a lehetőségek nagyon szerények.

Ezeknek a problémáknak a megoldása érdekében lehetővé tették, hogy a dokumentumban előforduló és előre megtervezett stílusokat a dokumentum tartalmától elkülönítve definiálja a weblap készítője. Ezt követően a dokumentum egy-egy részleténél már elegendő a megfelelő stílusra hivatkozni. Ha a dokumentum bizonyos stílusú részeinek formáján valamit módosítani akarunk, akkor most már elegendő egy helyen, a stíluslapon elvégezni a módosítást, és az automatikusan érvényesül a dokumentum teljes területén.

Ezeknek a HTML dokumentumokhoz *csatolt stílusleírásoknak* a neve az angol szakmai nyelvben *Cascading Style Sheets*, amit többnyire a *CSS* mozaikszóval helyettesítenek. A magyar szaknyelvben e fogalom jelölésére a *stíluslapok* kifejezés terjedt el. A stílusok definiálására egy új nyelvet vezettek be, amely szintaxisában és a kulcsszavaiban is jelentősen eltér a HTML-től.

Nézzünk egy példát a stíluslapok használatának szemléltetésére!

```
<html>
<head>
<title>Bevezető példa</title>
<style type = "text/css">
/* A CSS kommentár több sorra is
kiterjedhet */
h1 {
    color: darkblue;
    font-family: Arial;
    font-size: 24pt;
    text-align: center;
}
h2 {
    color: crimson;
    font-family: Arial;
    font-size: 16pt;
    font-style: italic;
    text-align: left;
}
/* A bekezdés stílusa */
p {
    font-family: Times;
    font-size: 12pt;
    text-indent: 20px;
}
</style>
<body>
<h1>A CSS alapjai</h1>

<h2>Mire jó a CSS?</h2>
```

```
<p>
```
A HTML oldalaink megjelenését befolyásoló egyszerű nyelvről van szó, mely segítségével meghatározhatjuk, hogy hogyan jelenjenek meg az egyes elemek. Többek között befolyásolhatjuk a színüket, méretüket, elhelyezkedésüket, margóikat, stb.
```
</p>
<h2>Mikor jött létre?</h2>
<p>
```
A technológia már viszonylag elég régóta létezik, a CSS szabvány leírása 1996-ban látott napvilágot. A szabvány azóta több kiadást ért meg, illetve 1998-ban napvilágot látott a CSS 2 szabvány leírása, amit 2006-ban kissé módosítottak. A CSS 3 kidolgozása pedig folyamatban van.
```
</p>
<h2>Miért jó?</h2>
<p>
```
A weblap szövegében jobban elkülönül a megjelenítendő tartalom és a formázásra vonatkozó előírások. Sokkal egyszerűbb a tervezés és a tervek módosítása.
```
</p>
</body>
</html>
```

Az újdonság a `<head>` szakaszon belül, a kiemelt részben látható. A `<style>` címkével írtuk le az egyes dokumentumrészek stílusának jellemzőit. Példaként tegyük fel, hogy a 2-es szintű címek betűszínét meg akarjuk változtatni. Ekkor csak a *crimson* helyett kell más színt megadnunk és minden `h2` stílusú cím az új színnel jelenik meg. Egy példa az új lehetőségekre: a HTML nem tette lehetővé, hogy egy bekezdés első sorát a többihez képest behúzzuk. Most ezt is megtehetjük a *text-indent* tulajdonság beállításával.

Megjegyzés a stíluslapban

A stíluslapba a /* jelzés után magyarázó szöveget iktathatunk be. A kommentár több soros is lehet, végét a */ karakterpár jelzi. Fontos, hogy jól értsük: a stíluslap nyelve más, ezért annak a területén a HTML kommentár érvénytelen szöveget jelent még akkor is, ha a stíluslap a HTML dokumentumba van beágyazva (ld. később)!

8.1 A stílusdefiníciók lehetséges elhelyezései

8.1.1 A dokumentumba beágyazott stíluslap

Ha csak egyetlen weblap stíluselemeit akarjuk definiálni, akkor célszerű megoldás lehet a beágyazott stíluslap alkalmazása. Az iménti példánk ezt a megoldást mutatja. A stílusok leírását a <head> szakaszban helyezzük el, így a stílusdefiníciókat a böngészők nem jelenítik meg.

`<style>` a stílusdefiníciós szakasz

Beágyazott stíluslap esetén a stílusokat a `<style>` és a `</style>` címkék között adjuk meg. A nyitó címke kötelező paramétere a type, amely a dokumentumrész MIME típusát adja meg, ami esetünkben mindig *text/css*. A nyitó címke tehát:

```
<style type="text/css">
```

8.1.2 Külső fájlban elhelyezett stíluslap

Ha nem csak egy, hanem több HTML dokumentumban is szeretnénk felhasználni ugyanazokat a stílusokat, akkor mindenképpen külön fájlban célszerű a stílusokat definiálni. *A definíciókat tartalmazó fájl csak egyszerű szövegfájl lehet, kötelezően css kiterjesztéssel!*

A stíluslap állomány tartalma

A fájl csak stílusdefiníciókat tartalmaz, ugyanolyan módon, mintha beágyazott stíluslapot készítenénk. Természetesen a `<style>` és a `</style>` címkék nem kerülhetnek a fájlba! Például, ha a bevezető feladatot egy *bevpld.css* nevű fájllal oldanánk meg, akkor a *bevpld.css* tartalma a következő lenne:

```
h1 {
    color: darkblue;
    font-family: Arial;
    font-size: 24pt;
    text-align: center;
}
h2 {
    color: crimson;
    font-family: Arial;
    font-size: 16pt;
    font-style: italic;
    text-align: left;
}
p {
    font-family: Times;
    font-size: 12pt;
    text-indent: 20px;
}
```

A stíluslap-állomány és a HTML dokumentum összekapcsolása, a <link> címke

A <head> szakaszban elhelyezett, pár nélküli `<link>` címkével kapcsoljuk a dokumentumunkhoz a stíluslapot tartalmazó fájt. A `<link>` kötelező paraméterei:

rel=”stylesheet”

A *rel* paraméter határozza meg, hogy a HTML dokumentum és a csatolt dokumentum milyen viszonyban van. CSS esetében ennek a paraméternek az értéke mindig *stylesheet*.

type=”text/css”

A `<style>` címkénél leírtakkal azonos módon kell használni!

href – a stíluslap-állomány megadása

A hiperhivatkozásnál tanultakhoz hasonlóan ezzel a paraméterrel adjuk meg az URL-t a stíluslapot tartalmazó állományhoz.

A bevezető példa HTML dokumentumában a fejrész most így alakul:

```
<head>
<title>Bevezető példa</title>
<link type ="text/css" rel="stylesheet" href="bevpld.css" />
</head>
```

A dokumentum további részei természetesen nem változnak.

8.1.3 HTML elemhez rendelt (inline) stílus

Ha a dokumentumban egy bizonyos stílust csak egyetlen helyen akarunk alkalmazni, akkor bizonyos mértékig joggal ítélhetjük feleslegesnek külön stílusdefiníció készítését. Ilyen esetekben a megfelelő HTML elemhez kapcsolhatunk stílusdefiníciót a következő módon: a HTML nyitó tagjában elhelyezzük a *style*= szöveget, majd egy nyitó és egy záró *idézőjel* (vagy aposztróf) *között*, pontosvesszővel elválasztva felsoroljuk a formázási szabályokat.

Az első példa egy `<h1>` stílusú szöveg stílusának módosítását végzi el.

```
<h1 style=”color: darkblue; font-family: Arial; font-size: 24pt; text-align:
center; ”> A CSS alapjai</h1>
```

A következő példa egy táblázat egy sorát formázza a `<tr>` elemhez rendelt stílusdefinícióval.

```
<tr style="color: blue; background-color: silver; text-align: center; font-size:
16pt;">
```

8.2 Stílusdefiníciók

8.2.1 A definíciók felépítése

Egy stílus definíciója a legegyszerűbb esetben a következőképpen néz ki:

```
kiválasztó {tulajdonság: érték;}
```

Ahol

- kiválasztó: meghatározza, hogy mi az, amit formázni kell
- tulajdonság: meghatározza, hogy a kiválasztott elem mely tulajdonságát akarjuk megszabni
- érték: előírja, hogy milyen legyen az adott tulajdonság.

Példa:

```
td {font-family: Arial;}
```

Ez a szabály előírja, hogy a dokumentumban minden `<td>` elem szövege Arial betűtípusú legyen.

Fontos, hogy minden egyes tulajdonság értékének megadását pontosvesszővel kell lezárni! Ha a definíció több tulajdonságot tartalmaz, akkor ezeket legalább egy szóközzel, tabulátorral vagy újsor karakterrel elválasztva kell felsorolni.

Példa a kódolási konvencióknak megfelelő, jól olvasható írásmódra:

```
body {
    font-family: Arial;
    font-size: 12pt;
    color: black;
    background-color: navy;
}
```

Több elem azonos tulajdonságait csoportosan definiálhatjuk, s ha szükséges, akkor párhuzamosan az eltérő tulajdonságokat is megadhatjuk. Például az alábbi példában a címsorok mindegyikének betűtípusát és színét közösen adjuk meg, a mellette a h1 stílus betűméretét külön definiáljuk.

```
h1, h2, h3, h4, h5 {
    font-family: Arial;
    color: darkblue;
}
h1  {
    font-size: 22pt;
}
```

8.2.2 Kiválasztók létrehozása és alkalmazása

Az univerzális kiválasztó

Ha bizonyos formai jegyeket az egész dokumentumra akarunk érvényesíteni, akkor az egyetemes kiválasztónak nevezett * karaktert kell alkalmaznunk. Természetesen a * kiválasztóra a dokumentum törzsében nem kell hivatkoznunk, hatása automatikus. Példa:

```
* {
    font-family: Arial;
    color: darkblue;
}
```

Amikor a kiválasztó egy HTML címke

A korábbiakban ilyen eseteket láttunk. Ezt a megoldást akkor alkalmazzuk, ha egy bizonyos HTML címkével jelölt részletnek a dokumentum minden helyén ugyanúgy kell megjelennie. Ekkor a kiválasztó maga a HTML címke neve.

A leszármazás alapú kiválasztók

Az ilyen típusú kiválasztók hatása attól függ, hogy egy elem leszármazottja-e egy másiknak. *A leszármazási kapcsolat itt azt jelenti, hogy a dokumentumban az egyik elemmel megnyitott szerkezet (ez az „ős”) tartalmazza a másikat (ez a másik a „leszármazott”).* Nézzünk két példát!

Az első példában az feladat, hogy minden szakaszon (ld. címke) belül megjelenő <h2> stílusú szöveg színét *crimson*-ra állítsa. Ehhez a stílusdefiníció a következő:

```
div h2 {
    color: crimson;
}
```

Mint látható, a leszármazási kapcsolatot jelölésében, a befoglaló szerkezet neve van elől. Tegyük fel, hogy a HTML dokumentumban a következő részlet szerepel:

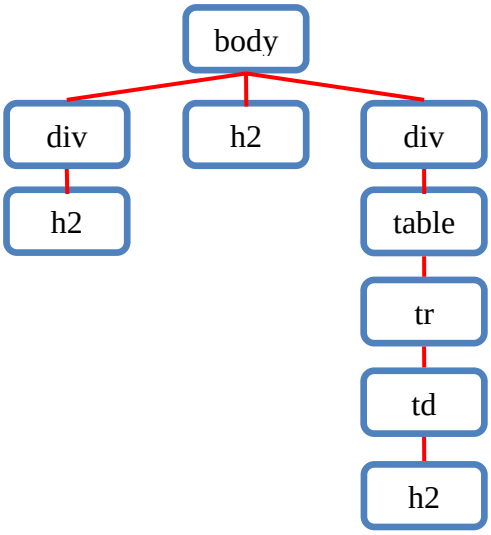
```
<div>
    <h2>A leszármazási kapcsolat</h2>
</div>
<h2>Univerzális kiválasztó</h2>
```

Mivel „A leszármazási kapcsolat” szöveg közvetlenül a szerkezeten belüli <h2> stílusú szöveg, ezért a böngésző *crimson* színnel jeleníti meg. Ezzel szemben az „Univerzális kiválasztó” szövegre a stílusdefiníció nem lesz hatással.

Vajon mi a helyzet akkor, ha a <h2> stílusú rész nem közvetlenül van a szerkezetben? Tegyük fel, hogy az előbbi dokumentumból származik a következő részlet is!

```
<div>
<table>
<tr>
<td>
<h2> A stílusok definiálása </h2>
</td>
</tr>
</table>
</div>
```

Az ábra a dokumentum elemeinek leszármazási kapcsolatait ábrázolja.



A táblázatos szerkezet leszármazási ágát nézve látható, hogy a <h2> elem csak többszörös közvetítéssel leszármazottja a elemnek. A böngésző azonban ilyen esetekben a közvetett leszármazást is elfogadja, tehát „A stílusok definiálása szöveg” is *crimson* színnel jelenik meg.

Feladat: értelmezzük az alábbi stílusdefiníciót!

```
div * {
    color: silver;
}
```

Közvetlen leszármazott (gyermek) kiválasztók[6]

Ezek a kiválasztók csak akkor érvényesülnek, amikor a leszármazási kapcsolat közvetlen. Tegyük fel, hogy az előző példa stílusdefinícióját a következőre módosítjuk:

```
div > h2 {
    color: crimson;
}
```

Most csak „A leszármazási kapcsolat” szöveg jelenne meg crimson színnel.

Osztály (class) alapú kiválasztók

Ezek a teszik lehetővé a stílusok legrugalmasabb alkalmazását, mert a stílus már nem kötődik egy meghatározott HTML címkéhez. Definiáljuk HTML elemek egy halmazának (más szóval: osztályának) tulajdonságait, de majd csak a dokumentum egy adott helyén mondjuk meg, hogy egy bizonyos HTML címke bele tartozik-e a stílusjegyek adott halmazába, vagy sem.

Tegyük fel például, hogy a dokumentum egy területén a háttér világos, és itt fekete szövegszínt akarunk, a `<strong>` stílussal kiemelt részek színét pedig pirossal szedjük. Egy másik területen a háttér sötét, és itt a szöveget fehérrel, a kiemelt részeket pedig sárgával szeretnénk megjeleníteni. A megoldás lehet pl. a következő:

```
<html>
<head>
<title>Osztály típusú kiválasztók</title>
<style type="text/css">
body {
    background-color: lightgray;
}
.darkbgr {
    background-color: black;
    color: white;
}
.lightbgr {
    background-color: white;
    color: black;
}
.darkstr {
    background-color: black;
    color: yellow;
}
.lightstr {
    background-color: white;
    color: red;
}
</style>
</head>
<body>
<h1 class="darkbgr">Példa osztály alapú stíluskiválasztásra</h1>
<p class="lightbgr">Tegyük fel például, hogy a dokumentum egy területén a háttér világos, és itt fekete szövegszínt akarunk, a <strong class="lightstr">strong</strong> stílussal kiemelt részek színét pedig pirossal szedjük.</p>
<p class="darkbgr">Egy másik területen a háttér sötét, és itt a szöveget fehérrel, a <strong class="darkstr">kiemelt</strong>részeket pedig sárgával szeretnénk megjeleníteni.</p>
<h1 class="lightstr">És még ezt is lehet!</h1>
</body>
</html>
```

Figyeljük meg, hogy a stílusdefinícióban az osztálynév előtt most *pontot kell tennünk*, de a dokumentum törzsben a *class* kulcsszó után *a név már a pont nélkül* szerepel!

[6] A CSS 3 szabványban definiált megoldás. Még nem minden böngésző támogatja.

A <h1 class="darkbgr"> a böngésző számára azt jelenti, hogy a <h1> címke hatókörébe eső szövegre a *darkbgr* korábban definiált stílusjellemzőit is alkalmaznia kell. Remélhetőleg a példa további részei értelmezhetők az olvasó számára.

Figyeljük meg az utolsó szövegsor formázását! Itt a <h1> elemet a *lightstr* osztályba tettük, noha ezt a stílusosztályt a feladat szerint nem erre szántuk. Az ilyen hibák elkerülése érdekében korlátozhatjuk egy stílusosztály alkalmazhatóságának körét azzal, hogy egy meghatározott címkéhez kötjük. Egy ilyen címkéhez kötött osztályt hiába próbálunk a definiálttól eltérő címkére alkalmazni, annak nem lenne hatása.

Példaként a törzset változatlanul hagyva módosítsuk az előbbi dokumentumban a stílusdefiníciók megfelelő részét az alábbi módon:

```
p.darkbgr {
    background-color: black;
    color: white;
}
p.lightbgr {
    background-color: white;
    color: black;
}
strong.darkstr {
    background-color: black;
    color: yellow;
}
strong.lightstr {
    background-color: white;
    color: red;
}
```

Mivel a *darkbgr*, *lightbgr* stílusokat a <p>, a *darkstr*, *lightstr* stílusokat a címkéhez kötöttük, más címkével hiába próbáljuk használni e stílusosztályokat, az nem fog működni. Jegyezzük meg, hogy a címkéhez kötött osztályokat szintaktikai szempontból ugyanúgy használjuk, mint a kötetleneket, ezért nem kellett az előbbi példában a dokumentum törzsén változtatni. Tehát például marad a kiemelt részlet is.

A stílustévesztés problémája rávilágít arra. hogy *beszédes neveket* célszerű használni. Gondoljunk arra, hogy még egy kisebb, de formailag igényes honlap esetében is várhatóan több tucat stílust kell használnunk. A név utaljon arra, hogy milyen célra, azaz milyen dokumentumrészek formázására fogjuk használni az adott stílust.

Lehetőség van arra is, hogy egy dokumentumrészre egyszerre több stílus osztály alkalmazását is előírjuk. Ekkor a *class* kulcsszó után szóközökkel elválasztva kell felsorolni az alkalmazandó stílusokat! Például legyen a két osztály a következő:

```
.kiemelt {
    color: red;
}

.definicio {
    font-family: Courier;
}
```

Majd a dokumentumtörzsben

```
<p class="kiemelt definicio"> Stíluson formai tulajdonságok egy halmazát értünk.</p>
```

Eredményeként a szöveg piros színnel, Courier betűtípussal (ha a böngésző számára ez a betűtípus elérhető) jelenik meg.

Azonosító (id) alapú kiválasztók

Ha az oldalhoz JavaScript kód is tartozik, vagy ha nagyon hosszú az oldal és célszerű lenne egyes részeinek gyors eléréséhez hivatkozásokat készíteni, akkor az osztály alapú kiválasztókkal szemben

előnyösebb lehet a hozzájuk formailag nagyon hasonló ún. *azonosító alapú* kiválasztókat használni. A stílusdefinícióban az eltérés annyi, hogy pont helyett kettőskeresztet (#) kell használni. A dokumentumtörzsben pedig a *class* kulcsszót az *id* helyettesíti. Nézzünk példákat!

```
#banner {
    background: black;
    height: 300px;
    width: 720px;
}
div#sponsor {
    background: darkblue;
    color:
}
```

A dokumentumtörzsben alkalmazva:

```
<div id=”sponsor”> <p>Honlapunk támogatója az International Peanut Research </p></div>
```

Paraméter alapú kiválasztók

Ebben az esetben a HTML címke valamilyen paraméteréhez kapcsoljuk hozzá egy stílus alkalmazását. A paramétert, ha szükséges, akkor értékével együtt, szögletes zárójelek közé kell tenni! Nézzünk néhány példát ízelítőül!

```
h1[align]  {
    text-background: black;
    color: white;
}
h2[align=”center”]   {
    text-background: gray;
    color: darkblue;
}
a[href=”http://www.lovassy.hu”] {
    color:red;
    font-weight:bold;
}
a[href$=”.pdf”]   {
    background-image: url(doc_icon.png) no-repeat;
    padding-left: 15px;
}
```

Az első példában a h1 címsorok számára definiált stílust a böngésző akkor alkalmazza, ha a `<h1>` címke paraméterlistájában előfordul az *align* paraméter, tekintet nélkül annak értékére.

A második példában a h2 címsorokhoz definiált stílust akkor alkalmazza, ha a `<h2>` paraméterlistájában az *align* paraméter pontosan a”*center*” értékkel szerepel.

A harmadik példában megadott stílust csak azokra a hiperhivatkozásokra érvényesíti, amelyekben a href értéke *”http://www.lovassy.hu”*.

A negyedik példában annyi az újdonság. hogy paraméter értékét leíró szöveg végére írunk elő feltételt. Ehhez a $ jelzést használjuk fel. A stílust minden olyan linkre alkalmazza, amelyben a hivatkozott fájlnév vége *.pdf*.

Pszeudo elemek

A hagyományos HTML címkék segítségével a dokumentum finomabb részleteire még csak hivatkozni sem tudunk. Ilyen részlet például a szöveg első sora, vagy első betűje. A CSS ezeknek a részleteknek az automatikus formázására bevezetett néhány ún. pszeudo elemet. Közülük nézzünk most a példa kedvéért hármat!

`::first-letter`	A stílus a HTML címkével kijelölt szöveg első betűjére vonatkozik
`::first-line`	A stílus a HTML címkével kijelölt szöveg első sorára vonatkozik
`::selection`	A felhasználó által egérrel kijelölt szöveg stílusa (csak CSS 3)

Megjegyzés: a CSS 3 szabvány szerint a név elé két kettőspontot kell tenni, ahogy fentebb bemutattuk, a régebbi szabványok egy darab kettőspontot írnak elő (pl. `:first-letter`). Itt is igaz: ma még kevés böngésző képes a CSS 3 megvalósítására. Az alábbi példa megírásának idején sem *Google Chrome*, sem *Internet Explorer 9* esetében nem működne a dupla kettőspontos jelöléssel, míg a *Safari* még a *::selection* értelmezésére is képes.

```
<html>
<head>
<title>Pszeudo elemek</title>
<style type="text/css">
* {
    font-family: Arial;
    font-size: 12pt;
}
p:first-letter {
    font-size: 300%;
    background-color: crimson;
    color: black;
}
</style>
</head>
<body>
<p>
A hagyományos HTML címkék segítségével a dokumentum finomabb részleteire még csak hivatkozni sem tudunk. Ilyen részlet például a szöveg első sora, vagy első betűje. A CSS ezeknek a részleteknek az automatikus formázására bevezetett néhány ún. pszeudo elemet.
</p>
</body>
</html>
```

Pszeudo osztályok

Elsősorban dinamikus, események hatására változó elemek formázására szolgálnak. A mai böngészőkkel elsősorban a hiperhivatkozások (ld. `<a>` címke) kezelésére használható eszközök. Néhány pszeudo osztályt felsorolunk példaként:

`:link`	A még meg nem látogatott dokumentum hiperhivatkozásának stílusa
`:visited`	A már meglátogatott dokumentum hiperhivatkozásának stílusa
`:active`	Annak az elemnek a stílusa, amelyet a felhasználó éppen egérrel kiválaszt
`:hover`	Annak az elemnek a stílusa, amely felett a felhasználó egere éppen áthalad.

Egy példa az alkalmazásukra:

```
<html>
<head>
<title>Pszeudo osztályok</title>
<style type="text/css">
a:link {
    background: blue;
    color: white;
}
a:visited {
    background: white;
    color: black;
}
a:hover {
    background: black;
    color: white;
}
a:active {
    background:red;
    color: white;
}
```

```
</style>
</head>
<body>
<p><a href="css08.html">Osztályok</a></p>
<p><a href="css09.html">Pszeudo elemek</a></p>
</body>
</html>
```

Megjegyzendő, hogy az ismertetett pszeudo osztályokat jelenleg az Internet Explorer 9 és a Google Chrome részlegesen támogatja, a Safari teljes egészében.

További érdekes lehetőségeket rejt az, hogy a *:hover* osztályt a mai böngészők már nem csak az `<a>` címkével, hanem más osztályokhoz kapcsolva is értelmezni tudják. Így pl. nagyobb területeket (szövegrészeket, rövid híreket, nagy méretű gombokat lehet kiemelni amikor a kurzor fölé kerül és egyben a kiemelés mellé hiperhivatkozást is lehet hozzá készíteni, ha szükséges. Például:

```
#highlited:hover {
    color: red;
    background-color: yellow;
    font-style: italic;
}
```

Majd pl. egy tetszőleges szövegrész kiemelése: `<span id="highlited">szöveg</span>`

8.3 A tulajdonságok értékének megadása

8.3.1 Szöveges értékek

Ügyeljünk arra, hogy *tulajdonság értékét, amennyiben az a CSS nyelvben egy foglalt szó, ne tegyük sem aposztrófok (’szöveg’), sem idézőjelek („szöveg”) közé*, mert ebben az esetben nem értékként, hanem közönséges szövegként kezeli a böngésző, ami az adott helyzetben értelmezhetetlen. Ebben tehát a CSS nyelv pontosan ellentéte az XHTML nyelvnek, amelyben a paraméterek értékét kötelező idézőjelek közé tenni.

Ennek megfelelően a következő stílusdefiníció hatástalan:

```
body {
       background-color: „navy”;
}
```

Ha a tulajdonság értéke nem foglalt szó, hanem a helyzettől függően alakuló szöveg, mint például egy URL, akkor idézőjelek közé tehető. Ha egy érték – pl. egy név – önmagában szóközt tartalmaz, akkor idézőjelek közé kell tenni! Példák:

```
background-image: url(“images/background_small.gif”);
font-family: “Times New Roman”, Times, serif;
```

8.3.2 Számok

A CSS mind egész, mind valós szám konstansokat elfogad mennyiségek leírásához, de nem fogad el kifejezéseket.

A **valós számok** kezdődhetnek + vagy – előjellel, a törtrész kezdetét pedig tizedes ponttal kell jelölni. Az előjelek értelmezése függ attól, hogy milyen tulajdonságról van szó. Például a

```
margin-left: -1.5em
```

eredménye az lesz, hogy a dokumentumnak az az eleme, amelyre ez a stílus elem vonatkozik, az aktuális betűméret másfélszeresével *balra* léptetve helyezkedik el.

Az **egész számok** ugyancsak kezdődhetnek előjellel, de tudnunk kell, hogy melyek azok a tulajdonságok, amelyek értéke nem lehet negatív.

8.3.3 Méretek

Hosszúsági adatokat kétféleképpen adhatunk meg: relatív vagy abszolút méretként. Az abszolút méret független a felhasználói környezettől, mint például a monitor képátlójától, felbontásától vagy az ablak méretétől. Ezzel szemben a relatív méretmegadás figyelembe veheti ezeket.

Abszolút kiterjedést jelentő adatoknál a következő mértékegység jelölések használhatók:

Jelölés	Leírás
in	Hüvelyk (inch)
cm	Centiméter
mm	Milliméter
pt	Pont (1 pont =1/72 hüvelyk)
pc	Pica (1 pica = 12 pont)

Példák:

```
font-size: 12pt
margin: 1in
```

Relatív hosszúságok megadásánál a viszonyítási alap jelzésére a következő jelölések használhatók:

Jelölés	Leírás
em	A legközelebbi betűmérethez viszonyított
ex	Az aktuális betűtípus kis x karakterének magasságához viszonyított
px	Pixelrelatív, tehát a megjelenítő eszköz (pl. monitor) elemi pontméretéhez viszonyított

Példa:

```
div {
    font-size: 12px;
    height: 2em;
}
```

A szabály szakasz számára 12 pixel méretű betűket ír elő. Mivel a magasság értékét a legközelebbi betűméret kétszeresére veszi (2em), és ez a betűméret éppen a most definiált 12 pixel, a div elem magassága 24 pixel lesz.

Az *ex* viszonyítási alap kevéssé megbízható, mert a böngészők többnyire 0,5em értékkel helyettesítik.

A *px* esetében figyelembe kell venni, hogy a pixelméret függ a monitor aktuális felbontásától. Ha 800x600 a felbontás, akkor az 1px nagyobb távolságot jelent, mintha a felbontás 1024x768. Képek méreteinek megadásánál ez általában nem jelent problémát, mert a képek méreteit eleve pixelekben szokás kalkulálni már a kép elkészítésénél. Jóval több problémát okozhat a méretek pixelekben történő megadása, ha táblázat méretének, vagy pozíciónak a megadására használjuk, s különösen akkor, ha az oldalt ki is nyomtatják.

A **százalékban kifejezett méretek** természetesen ugyancsak relatív mennyiségek. Jelölésére a mérőszámot követő % jelet használjuk. A százalékos méretmegadás esetében nagyon pontosan kell azt tudni, hogy mi az alapméret, amelynek valamennyi százalékát akarjuk megadni méretként. Nézzük a következő példát!

```
div {
    width: 100%;
    color: white;
    background: black;
}
```

A százalékos méret itt a `<div>` elemnél a rendelkezésre álló szélesség 100%-a. Ha a körülményekről mást nem tudunk, akkor ez a `<body>` elem szélességét jelenti. De ha a `<body>`

elemnek zérusnál nagyobb margója van, akkor a szakasz tartalma sem terjedhet túl ezen a margón.

8.3.4 Színek

Színek megadása névvel

A színeknek szabványos színnévvel történő megadása nemcsak egyszerűsége miatt előnyös, hanem azért is, mert a jobban áttekinthetővé teszi a forráskód szintjén azt, hogy milyen színeket használunk, és melyiket milyen célra.

A CSS 3 szabvány 147 színnevet tartalmaz. Az elterjedt böngészők az összes színnevet támogatják. A szabványos színnevek és definíciójuk az interneten könnyen megtalálhatók, a W3C honlapján a teljes CSS szabványok részeként is elérhetők. Példa:

```
body {
    background-color: gray;
}
```

Decimális RGB színkódok

Megadhatjuk a kívánt színárnyalatot, úgy is, hogy a *vörös*, a *zöld* és a *kék* színösszetevők értékét tízes számrendszerbeli számokkal adjuk meg a fenti sorrendnek megfelelően felsorolva. Mint az olvasó számára ismert, az egyes összetevők maximális erőssége 255. A pontos szintaxis kiolvasható az alábbi példából:

```
body {
    background-color: rgb(128, 128, 128);
}
```

A CSS szabvány szerint a szín megadható úgy is, ha megadjuk, hogy az egyes összetevők a teljes erősség hány százalékával vesznek részt a szín kialakításában. Példa:

```
body {
    background-color: rgb(50%, 50%, 50%);
}
```

Megjegyzendő, hogy a CSS 3 tartalmaz lehetőséget a szín átlátszóságának megadására is, de ezt még a böngészők nem támogatják.

Hexadecimális RGB színkódok

Ismert, hogy HTML-ben egy szín RGB összetevőinek értékét 16-os számrendszerben kell megadni, minden összetevőt két számjeggyel. A hat számjegy között nincs tagoló jel és az első számjegy előtt kettőskereszt jelzi, hogy hexadecimális számról van szó. Ezt a színkódot a CSS is elfogadja. Példák:

```
body {
    background-color: #808080;
}
div {
    color: #000000;
    background-color: #FF0000;
    border: thin solid #FFA500;
}
```

Web-biztos színek kódolása

A World Wide Web elterjedésének kezdetén a számítógépek a maiakhoz képest kevés színt tudtak megjeleníteni, de már voltak közöttük jelentős különbségek ezen a téren. A weblapok tervezői számára bizonytalan volt, hogy milyen színeket alkalmazhatnak, ha azt szeretnék, hogy minden felhasználójuk számára ugyanúgy nézzenek ki a weblapjaik. Megegyeztek abban, hogy legalább 8 bites színmélység elvárható a felhasználó számítógépétől és böngészőjétől. Az így rendelkezésre álló 256 színt két csoportra osztották. A weblapok számára olyan színeket tartottak fenn, amelyek hexadecimális RGB kódjaiban csak a következő értékek szerepelhetnek: 00, 33, 66, 99, CC, FF. Ezzel *összesen 216 féle színárnyalat* volt kódolható. A maradék 40 színt fenntartották az operációs

rendszer részére. A 216 árnyalatra ettől kezdve a tervezők úgy gondoltak, hogy minden böngésző képes ezeket helyesen megjeleníteni. Ezért nevezték ezt a palettát *web-biztosnak* (web-safe). Ma már a megjelenítő eszközök 24 bites színmélységgel dolgoznak és természetesen így a böngészők számára sem jelent problémát akár több millió színárnyalat megjelenítése sem. A web-biztos színek fogalma tehát elveszette jelentőségét[7], de a CSS szabvány még ismeri ezt a fogalmat, sőt tartalmaz egy rövid kódolási lehetőséget is, amely kihasználja, hogy egy RGB összetevő hexadecimális kódjában a második karakter ugyanaz, mint az első, és így 6 helyett elegendő 3 karakter a teljes szín kódolásához. Példa:

```
p {
    color: #000;
    background-color: #F00;
}
```

8.3.5 URL megadása

A stíluslapokkal kapcsolatban kétféle erőforrás (fájl) megadása jöhet szóba: stíluslap vagy kép. A szintaxis kiolvasható az alábbi példákból:

```
background-image: url(„images/background.gif”);
list-style-image: url(“pepper.gif”);
background-image: url('../images/backgrounds/palms.jpg’);
```

Az útvonal a HTML-ben megismert logika szerint lehet abszolút, vagy relatív *a stíluslapot tartalmazó helyhez képest*. (Amennyiben a HTML dokumentumba beágyazott a stíluslap, akkor a HTML dokumentum helyéhez képest, különben a *.css fájl helyéhez képest.)

8.4 Formázás stíluslapokkal

A stíluslapok használatának részletes bemutatására nincs módunk. Az alábbiakban csak egy-egy pillantást vetünk a fontosabb lehetőségekre. Szerencsére annak, aki az eddigieket megértette, az interneten elérhető ismertetők és oktatási anyagok segítségével nem lesz túl nehéz az alaposabb tudás megszerzésére.

Jelmagyarázatként: a CSS tulajdonságok alábbi táblázataiban félkövér dőlt szedés jelzi, hogy nem körülírással, hanem foglalt szóval van dolgunk, amit betű pontossággal ugyanúgy kell beírnunk.

8.4.1 Karakterformázás

Alkalmazható tulajdonságok

Tulajdonság	Szerep	Értékek
`font`	A betűtípus valamennyi tulajdonságának állítása egyetlen deklarációban felsorolva	• megadható a következő tulajdonságok értéke: *font-style, font-variant, font-weight, font-size/line-height, font-family,* • rendszer *betűtípusok* esetén: ***caption***, ***icon***, ***menu***, ***message-box***, ***small-caption***, ***status-bar***
`font-family`	A betűtípus beállítása	egy vagy több betűtípus neve
`font-size`	A betű méretét állítja be	• az előző méret 1,2-szerese a méretkülönbség: ***xx-small***, ***x-small***, ***small***, ***medium***, ***large***, ***x-large***, ***xx-large*** • az előzőhöz *képest* egy fokozattal változik a méret: ***smaller***, ***larger*** • százalékos *méret*: *a szülő elem betűméretének százalékában* • abszolút *méret*: *mértékegységek: px, pt, em*
`font-size-adjust`	A betűméretből eredő betűmagasság és a kis x karakter magasságának aránya (ha az értéket növeljük, akkor a betűméret csökkentése kevésbé rontja az olvashatóságot).	**none**, *egy szám (például 0.58)*

[7] Kivételt képeznek azok az esetek, amelyekben a weblap tervezőjének tekintettel kell lenni olyan ipari monitorokra, régebbi PDA-kra és mobiltelefonokra, amelyek 8 bites színmélységre képesek csak.

`font-stretch`	Változtatja a betűszélességet keskenyebbre vagy szélesebbre.	***normal***, ***wider***, ***narrower***, ***ultra-condensed***, ***extra-condensed***, ***condensed***, ***semi-condensed***, ***semi-expanded***, ***expanded***, ***extra-expanded***, ***ultra-expanded***
`font-style`	A font stílusát állítja be	***normal***, ***italic***, ***oblique***
`font-variant`	A szöveget vagy kisméretű nagybetűkkel („kis kapitális”) vagy normálisan jeleníti meg	***normal***, ***small-caps***
`font-weight`	Megszabja, hogy a betű a szövegben mennyire erőteljes, azaz mennyire vastag és sötét, a megjelenése.	***normal***, ***bold***, ***bolder***, ***lighter***, ***100, 200, 300, 400, 500, 600, 700, 800, 900***

Példák az alkalmazásra

```
.articlehead {
    font: italic small-caps bold 1.5em/1.2em sans-serif;
{
.normaltext   {
    font-family: Arial, “Microsoft Sans Serif”, Tahoma, Helvetica, sans-serif;
    font-style: normal;
    font-size: 12pt;
}
.examples {
    font-family: “Courier New”, Courier, monospace;
    font-size: 80%;
}
p.def {
    font-family: serif;
    font-style: italic;
    font-weight: 300;
    font-size: larger;
}
.extra {
    font-family: serif;
    font-variant: small-caps;
    font-weight: bold;
    font-size: medium;
}
```

Megjegyzés a választható betűtípusokról

Mivel operációs rendszerenként, sőt számítógépenként eltérő a böngésző számára rendelkezésre álló betűtípusok készlete. Ezért megegyezés van arról, hogy bizonyos betűtípusokat minden böngésző képes megjeleníteni megközelítőleg egyformán. Ezeket a betűkészleteket *generikus betűtípusoknak* nevezik. Ha a tervező generikus típust ad meg, akkor garantált, hogy a szöveg böngészőtől és operációs rendszertől függetlenül jó közelítéssel egyformán jelenik meg. A CSS a következő generikus betűtípus neveket rögzíti:

- *serif* (megvalósítja pl. a Times illetve a Times New Roman típus)
- *sans-serif* (megvalósítja pl. a Helvetica és az Arial típus)
- *monospace* (megvalósítja pl. a Courier vagy a Courier New állandó betűszélességű típus)
- *cursive*
- *fantasy*

A font-weight esetében tudni kell, hogy a számszerűen megadott súlyokkal jelzett vastagsági különbségeket a legtöbb általánosan használt betűkészlet esetében a böngésző nem képes megjeleníteni, mert csak a legmagasabb professzionális minőségre tervezett betűkészletekben található a „bold” (a magyar szaknyelveben: félkövér) mellett nyolc különböző mértékben kövér betűváltozat. Ha erre mégis szükség van, akkor a weblapnak külön le kell töltenie a szükséges betűkészletet. A *bolder* és *lighter* értékek a szülő környezetben érvényeshez képest jelentenek vastagabb illetve világosabb betűket.

A font tulajdonság segítségével egyetlen deklarációban megadhatjuk a betűk majd minden tulajdonságát. Legalább a betűméretetet kötelező megadni! A betűmérethez egy / jellel hozzákapcsolva megadhatjuk a sormagasságot is, de ez más nem kötelező. A fenti példákban az `.articlehead` stílusosztály mutatja a szintaxist. Ennek az osztálynak a jellemzői tehát a következők: dőlt, kis kapitális, félkövér, a betűméret a környező szöveg betűméretének másfélszerese, a sormagasság pedig 1,2-szeres, a betűtípus generikus, sans-serif.

8.4.2 Szövegformázás

Az alkalmazható fontosabb tulajdonságok

Tulajdonság	Szerep	Értékek
`color`	A szöveg színét állítja be	szín megadása a tanult módon
`letter-spacing`	A karakterek közötti távolság beállítása	***normal***, vagy a távolság megadása (negatív is lehet)
`word-spacing`	A szavak közötti távolság (a szóköz méretének) beállítása	***normal***, vagy a távolság megadása (negatív is lehet)
`line-height`	A sor magasságának beállítása	***normal***, *vagy* a távolság megadása
`text-align`	A szöveg igazítását szabályozza	***left***, ***right***, ***center***, ***justify***
`text-decoration`	A szöveg speciális alakítása alá- illetve áthúzással, villogással.	***none***, ***underline***, ***overline***, ***line-through***, ***blink*** (több is megadható egyszerre)
`text-indent`	Az első sor behúzását szabályozza	a távolság megadása (lehet negatív is)
`text-shadow`	A szöveg árnyékhatását határozza meg	***none***, ***color***, ***length***
`text-transform`	A karakterek átalakítását végzi nagybetűssé, kisbetűssé, illetve a szavak első betűjét nagybetűssé	***none***, ***uppercase***, ***lowercase***, ***capitalize***

`white-space`	Meghatározza, hogyan kezelje a böngésző a szóközöket, tabulátorokat. (Alapértelmezett a „normal” viselkedés.)	• ***normal*** (egynél több szóközt nem vesz figyelembe és átlépi a szövegben lévő nem HTML sortöréseket)
		• ***pre*** (megtartja a szóközöket, mint a HTML `<pre>` címkéjéhez hasonlóan)
		• ***nowrap*** (nem alkalmaz automatikus sortörést ha betelik a sor, ettől kezdve csak egy
		 törheti el a sort)

Példák az alkalmazásra

```
p {
    color: black;
    letter-spacing: 10px;
    word-spacing: 10px;
    line-height: 3em;
    text-indent: 25%;
    text-align: justify;
}
.condensed {
    letter-spacing: -2px;
    word-spacing: -5px;
    text-indent: -25px;
    text-decoration: underline;

}
h1, h2, h3, h4, h5 {
    color: darkblue;
    text-align: right;
    text-transform: capitalize;
}
#special {
    text-decoration: underline overline;
    text-transform: uppercase;
    white-space: nowrap;
}
```

8.4.3 Hátterek

Alkalmazható tulajdonságok

Tulajdonság	Szerep	Értékek
`background`	Az összes háttér tulajdonság beállítása egyetlen deklarációban felsorolva az értékeiket.	Megadható a következő tulajdonságok értéke (legalább egy): *background-color, background-image, background-repeat, background-attachment, background-position*
`background-attachment`	Beállítja, hogy a kép gördüljön-e az oldal tartalmával együtt, ha a felhasználó a gördítősávot használja, vagy sem. Alapértelmezés: *scroll*	• ***scroll*** (együtt gördül) • ***fixed*** (helyben marad)
`background-color`	Az elem háttérszínét állítja be. Alapértelmezés: transparent, azaz nem használ háttérszínt.	szín megadása a tanult módon *vagy* ***transparent***
`background-image`	A megadott képet háttérképként kezeli. (Alapértelmezésben az adott elem teljes rendelkezésre álló hátterét kitölti a képet ismételve.)	A képfájlhoz url megadása vagy ***none***
`background-position`	A háttérkép kiindulási helyzetét állítja be. A pozíciót két adattal kell megadni!	Lehetséges értékpárok (a páron belül a sorrend felcserélhető): ***top left***, ***top center***, ***top right***, ***center left***, ***center center***, ***center right***, ***bottom left***, ***bottom center***, ***bottom right***, vagy • mindkét adat %-ban • csak egyetlen %-os adatot adunk meg, ekkor a másikat 50%-nak veszi • keverjük a fenti kulcsszavak egyikét egy %-os adattal vagy a két koordinátát valamilyen CSS hosszúság egységben mérve (pl. px) adjuk meg
`background-repeat`	A háttérkép ismétlődését szabályozza. Alapértelmezés: *repeat*, azaz a képet mindkét dimenzióban ismétli. (Ld. background-image tulajdonság!)	• ***repeat*** (képet mindkét dimenzióban ismétli) • ***repeat-x*** (a képet csak az x tengely mentén, azaz vízszintesen) ismétli) • ***repeat-y*** (a képet csak az y tengely mentén, azaz függőlegesen) ismétli) • ***no-repeat*** (a kép csak egy példányban jelenik meg)

Példák az alkalmazásra

```
h1 {
    background-color: aquamarin;
}
div#super {
    background-color: #00ff00;
}
div#navigation a:hover {
    background-color: red;
}
```

Az utóbbi példa értelmezése: a „navigation” azonosítójú `<div>` elem területén a hiperhivatkozások háttérszíne legyen piros, ha az egérkurzort a felhasználó a hivatkozás fölé viszi.

```
body {
```

```
    background-image: url('../images/backgr.gif');
    background-repeat: repeat-y;
}
div#article {
    background-image: url('artic.jpg');
    background-repeat: no-repeat;
    background-position: center center;
    background-attachment: fixed;
}
div#head {
    background-image: url('headbkgr.jpg');
    background-repeat: no-repeat;
    background-position: 20% 10%;
}
div#footer {
    background-image: url('footbkgr.jpg');
    background-repeat: no-repeat;
    background-position: 20px 20px;
}
div {
    background: transparent url('bkgr.jpg') no-repeat scroll top center;
}
```

Megjegyzendő, hogy a *transparent* és a *scroll* alapértelmezett, pozícióban pedig a *top left* pár az alapértelmezett, így a szabály a következőre egyszerűsíthető: `background: url('bkgr.jpg') no-repeat center;`

```
div#content {
    background: lightyellow url('cont.gif') no-repeat fixed center center;
}
```

8.4.4 A weblaptervezés alapelveiről

A weblapoknál gyakori, hogy egy oldalon a közlendő tartalom elkülönülő részekből (pl. logo, hirdetés, rövid hírek, navigáló gombok, stb.) áll, amelyeket különböző szempontok (pl. esztétikai, pszichológiai, üzleti) szerint a felület különböző részein kell elhelyezni. Tisztán HTML kódolású dokumentumok esetében egy összetettebb *elhelyezési tervet* (layout) csak táblázattal lehet megvalósítani. A táblázat egy-egy celláját ekkor a tartalom egy-egy elemének *tároló dobozaként* használjuk. Ez a módszer egyrészt esztétikai szempontból csak szegényes lehetőségeket biztosít, másrészt nehezen áttekinthető és nagy méretű forráskódot eredményez. A stíluslapok segítségével minden tekintetben sokkal jobb eszközt kapunk a kezünkbe.

A szerepe

A CSS alapú kódolás a dokumentum elkülönítendő részeit önálló szakaszként kezeli, s a `<div>` címkét *általános célú tárolóként* használja, amelynek minden tulajdonságát a hozzá kapcsolt, stíluslapon definiált stílus adja meg. Egy példa kizárólag a technika szemléltetésére:

`<div id="copyright">` © Lánczos és fia Kft. `</div>`

Egy `<div>` szakaszon belül lehet több további `<div>` szakasz is, ily módon a `<div>` mint tároló a tartalmi elemek *csoportosítására* is alkalmas, nem csak a szétválasztására.

Természetesen nem helyes, ha minden más HTML címkét a `<div>` címkével próbálunk helyettesíteni. Láttuk, hogy stílust sokféle módon kapcsolhatunk a címkékhez, így a CSS előnyeit mindenütt kiaknázhatjuk.

Vannak azonban olyan HTML címkék, amelyek használatát kerülnünk kell, ha stíluslapokat használunk, mivel bizonyos címkék még a korábbi szemléletmódot képviselik, ezért csökkentik a stíluslapok alkalmazásának előnyeit. Pl. ha egy részlet megjelenését nem teljesen a hozzá kapcsolt stílus szabja meg, akkor kénytelenek vagyunk a szövegben a problémát okozó HTML címkét is megkeresni, megváltoztatni, ahányszor csak a megjelenés módosítására szükség van. Pl. ezért kerülendő a `<center>` és a `<font>` címke, és néhány címke esetében az *align* paraméter. Hasonló

okokból kerülendő a szövegbe ágyazott <b> és <i> címke. Helyettük használjunk megfelelő szövegstílust a *font-style* és *font-weight* tulajdonságokkal! Ennek megvalósítási módját hamarosan megvizsgáljuk (ld.). Ha valamilyen ok miatt mégsem stílussal formázzuk, akkor a <b> helyett használjuk inkább címkét, amit a böngészők félkövér formázással valósítanak meg, ugyanakkor a HTML is inkább a szöveget beszédesen tagoló mintsem formázó címkének tekinti. Hasonló okokból az <i> helyett ajánlott az használata.

A szerepe

Ha a szöveg egy tetszőleges részére helyben, tehát HTML elemhez rendelt (inline) módon akarunk egy stílust érvényesíteni, akkor a címkét használhatjuk általános célú tárolóként, mert nincs önálló formázó hatása a szövegre. Példa:

```
<span style=”font-weight: bold font-style: italic;”>Félkövér és dőlt szöveg</span>
```

Természetesen a címkét nem csak inline módon alkalmazhatunk, hanem külön stíluslapon definiált stílust is rendelhetünk hozzá. Példa:

```
<html>
<head>
<title>A span alkalmazása</title>
<style type = "text/css">
#highlited {
    color: red;
    font-style: italic;
}
</style>
<body>
<p>
A HTML oldalaink megjelenését befolyásoló egyszerű nyelvről van szó, mely <span
id="highlited">segítségével meghatározhatjuk, hogy hogyan jelenjenek meg az egyes elemek</span>. Többek
között befolyásolhatjuk a színüket, méretüket, elhelyezkedésüket, margóikat, stb.
</p>
</body>
</html>
```

Röviden összegezve az elmondottakat: *ha CSS stílusokkal formázzuk a dokumentumot, akkor kerüljük el a HTML minden olyan elemének használatát, amelyet csak a dokumentum megjelenésének kialakítására használnánk.*

8.4.5 A CSS doboz modellje

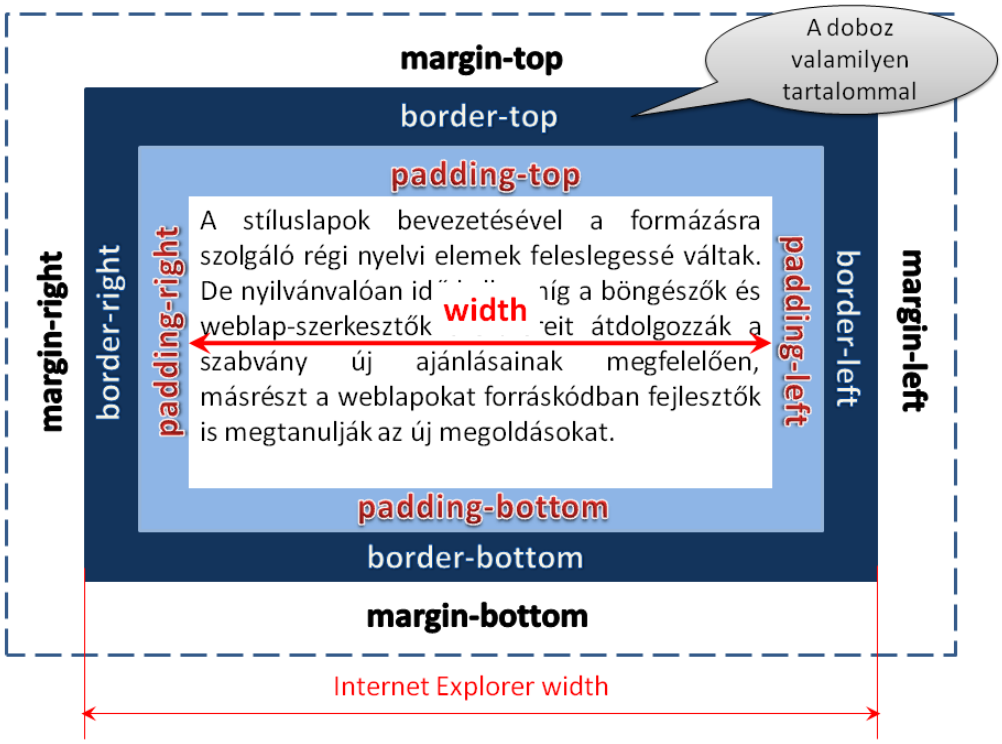
A CSS szemléletében minden dolog, ami elkülönül a környezetétől, egy „dobozban”, azaz a felület egy téglalap alakú részében helyezkedik el. Ez az egyszerű, de nagyon hatékony elképzelés lehetővé teszi, hogy a weblapok tartalmi elemeit el tudjuk helyezni az oldalon továbbá méretezni és formázni tudjuk a megjelenítésükre szolgáló területet.

A kiterjedés méretei mellett minden doboznak három alapvető tulajdonsága van:

- margó: a doboz körüli térközt jelenti
- szegély: vonal, amely a doboz szélén húzódik és elhatárolja azt a környezetétől
- belső margó: a doboz széle és a megjelenített tartalom közötti térköz

Az alábbi ábra a doboz modellt mutatja a később tárgyalandó tulajdonságnevekkel együtt. Megjegyzendő, hogy az Internet Explorer a 6-os verziót megelőzően a doboz szélességét a szabványtól eltérő módon értelmezték, ezért ezt külön is feltüntettük az ábrán. Az Internet Explorer 6 és újabb verziói a megfelelő !DOCTYPE deklarációval átkapcsolhatók a szabványt követő üzemmódba. Lapozz vissza a 2.1.2 részhez, onnan kimásolható!

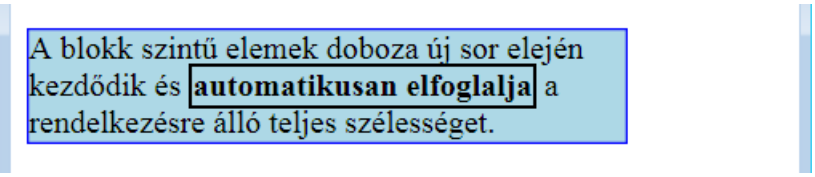
A doboz modell



Stíluslapokkal dolgozva különösen fontos, hogy lássuk a különbséget az ún. *blokk* szintű, illetve az ún. *soron belüli* (inline) címkék között, mert ez a különbség meghatározza, hogy az egyes dobozok hogyan jelennek meg. A blokk szintű elemek, mint a `<h1>`, `<p>`, `<div>`, doboza új sor elején kezdődik és automatikusan elfoglalja a rendelkezésre álló teljes szélességet. Ezt a szélességet az az elem határozza meg, amely a szóban forgó blokk szintű elemet tartalmazza. Ezzel szemben az olyan soron belüli elemek, mint a `<strong>`, `<em>`, `<img>`vagy a nem ajánlott `<b>`, `<i>` doboza pontosan a nyitó címkénél kezdődik és éppen a záró címkéig terjed. Példaként készítsük el a következő dokumentumot:

```
<html>
<head>
<title>Doboz modell</title>
<style type="text/css">
div {
    margin: 0;
    width:300px;
    border: dashed solid black;
}
p {
    border: thin solid blue;
    background-color: lightblue;
}
strong {
    border: 2px solid black;
}
</style>
</head>
<body>
<div>
<p>A blokk szintű elemek doboza új sor elején kezdődik és <strong>automatikusan elfoglalja</strong> a rendelkezésre álló teljes szélességet. </p>
</div>
</body>
</html>
```

A <p> elemet tartalmazó szakasz szélességét 300 pixelre korlátoztuk. A böngészőben megjelenő eredmény:



Fontos tudnivaló, hogy a böngészők a dobozok egyes tulajdonságaihoz (pl. belső margó) a szabványban nem rögzített alapértelmezett értékeket kapcsolnak, s ezek ráadásul böngészőnként eltérőek lehetnek.

8.4.6 Margók

Margó tulajdonságok

Tulajdonság	Szerep	Értékek
`margin`	Az összes margó tulajdonság beállítása egyetlen deklarációban felsorolva az értékeiket.	Megadható a következő tulajdonságok értéke: margin-top margin-right margin-bottom margin-left
`margin-bottom`	Az elem alsó margóját állítja be.	***auto***, vagy a távolság megadása abszolút illetve százalékos mennyiségként (negatív érték is lehet)
`margin-left`	Az elem bal margóját állítja be.	
`margin-right`	Az elem jobb margóját állítja be.	
`margin-top`	Az elem felső margóját állítja be.	

Példák az alkalmazásra

```
p {
    margin-right: 0;
    margin-left: 20px;
}
.head {
    margin-top: 10%;
    margin-bottom: 5%;
}
```

Összevont margó deklaráció esetén a paraméterek sorrendje a következő:

```
    margin: <margin-top> <margin-right> <margin-bottom> <margin-left>;
```

Példa:

```
margin: 10px 0 5px 20px;
```

Az alábbi megoldások ugyancsak szabályosak:

```
margin: <margin-top> <margin-right/margin-left> <margin-bottom>;
margin: <margin-top/margin-bottom> <margin-right/margin-left>;
margin <margin-top/margin-right/margin-bottom/margin-left>;
```

A következő példában a jobb és a bal margó a rendelkezésre álló hely 2%-a, míg az alsó margó negatív. Az alatta lévő eset érdekessége az, hogy az alsó margót a böngésző alapértelmezésén hagyja.

```
margin: 10px 2% -10px;
margin: 11px 2% -15px auto;
```

Blokkok igazítása a margó *auto* értékével

A `margin-left: auto;` és a `margin-right: auto;` szabályok használhatók blokk szintű elemek (pl. <h1>, , <p>) *vízszintes* igazítására. Ha mindkettőt alkalmazzuk, akkor az elemet a rendelkezésre álló térköz közepére helyezi. Érvényes ez az egész dokumentum *vízszintesen* középre igazítására is a böngésző ablakában például rögzített szélességű oldal esetén.

Margó „elnyelés”

Ha két elem margója érintkezik egymással, akkor a két margó térköze nem adódik össze, hanem a nagyobb margó „elnyeli” a kisebbet.

8.4.7 A doboz méretei

A dobozok méreteit a következő tulajdonságok szabályozzák

Tulajdonság	Szerep	Értékek
`height`	A doboz magasságát állítja be.	***auto***, vagy a távolság megadása abszolút illetve relatív (%, em) mennyiségként
`width`	A doboz szélességét állítja be.	
`line-height`	Megszabja, hogy milyen magas legyen egy szövegsor. (Ld. korábban.)	***normal***, *vagy* a távolság megadása
`max-height`	Megszabja, hogy mekkora lehet legfeljebb a doboz magassága.	a távolság megadása abszolút illetve százalékos mennyiségként, alapértelmezett érték: ***0***
`min-height`	Megszabja, hogy mekkora legyen legalább a doboz magassága.	
`max-width`	Megszabja, hogy mekkora lehet legfeljebb a doboz szélessége.	
`min-width`	Megszabja, hogy mekkora legyen legalább a doboz szélessége.	

Példák a használatra

```
width:200px;
height:30em;
width: 25%;
height: auto;
```

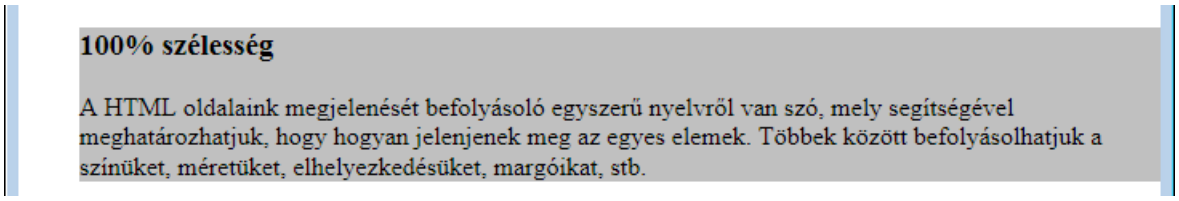
Megjegyzendő, hogy néhány régebbi böngésző verzió (Mozilla, IE a 6-os verzió előtt) a *height* tulajdonságot nem egészen a szabványnak megfelelően valósítja meg.

A *height* és *width* tulajdonságok alapértelmezése az *auto* érték. Az *auto* értelmezése függ attól, hogy milyen HTML elemre alkalmazzuk a *height* illetve a *width* tulajdonságot. A `<div>` és más blokk jellegű elemre alkalmazva az *auto* szélesség hatására az elem vízszintesen kiterjed a teljes rendelkezésre álló távolságra, az *auto* magasság hatására az elem olyan magas lesz, amilyen magasságot a tartalom megjelenítése igényel.

A méret százalékos megadásánál ügyelni kell arra is, hogy a doboz modellnek megfelelően a margó, a szegély és a belső margó mérete nem számítandó a *width* illetve a *height* mérethez. Példa:

```
div {
    margin: 30px;
    width: 100%;
    background-color: silver;
}
```

Ezt a stílust alkalmazva a következő eredményt kapjuk:



Mint látható, a jobb oldali margónak nem maradt hely, a tartalom számára teljes maradék távolságot lefoglalta a `width:100%;` előírás alapján.

A tartalom elhelyezésének tervezésekor gyakran van szükség arra, hogy az elem mérete ne csökkenhessen egy határérték alá, vagy éppen arra, hogy ne növekedhessen egy határérték fölé. Ezekben az esetekben használjuk a táblázatban látható maximum illetve minimum méret tulajdonságokat. Példa:

```
p {
    margin: 10px;
    background-color: silver;
    width: auto;
    min-width: 300px;
}
```

Alkalmazzuk ezt a stílust egy szöveg megjelenítésére, majd csökkentsük a böngésző ablakának szélességét! Figyeljük meg, hogy a 300 pixel szélesség elérésekor megjelenik a vízszintes gördítősáv, ami jelzi, hogy a <p> elem mérete nem csökken tovább, csak az ablakban már nem fér el.

8.4.8 Szegélyek

Szegély tulajdonságok

Egy szegélynek három alapvető tulajdonsága van: *vastagság* (width), *stílus* (style) és *szín* (color). A doboznak mind a négy szegélye külön-külön formázható, de mód van több tulajdonság összevont deklarációjára is.

Tulajdonság	Szerep	Értékek
`border`	A négy keret tulajdonság összevont deklarációja	Megadható a következő tulajdonságok értéke: *border-width, border-style, border-color*
`border-left`	A bal keret tulajdonságainak beállítása egyetlen deklarációban	Megadható a következő tulajdonságok értéke: *border-width, border-style, border-color*
`border-right`	A jobb keret valamennyi tulajdonságának beállítása egyetlen deklarációban	
`border-top`	A felső keret tulajdonságainak beállítása egyetlen deklarációban	
`border-bottom`	Az alsó keretre vonatkozó tulajdonságok egyidejű állítása egyetlen delkarációban	
`border-color`	A négy oldal keret színét állítja be, egytől négy színt tartalmazhat	Megadható a következő tulajdonságok értéke: *border-left-color, border-right- color, border-top- color, border-bottom- color*
`border-left-color`	A bal keret színét állítja be	Szín megadása a tanult módon
`border-right-color`	A jobb keret színét állítja be	
`border-top-color`	A felső keret színét állítja be	
`border-bottom-color`	Az alsó keret színét állítja be	
`border-style`	A négy keret stílusát állítja be, egytől négy stílust tartalmazhat	Megadható a következő tulajdonságok értéke: *border-left-style, border-right- style, border-top- style, border-bottom- style*
`border-left-style`	A bal keret stílusát állítja be	Megadható értékek: ***none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset*** alapértelmezett: ***none***
`border-right-style`	A jobb keret stílusát állítja be	
`border-top-style`	A felső keret stílusát állítja be	
`border-bottom-style`	Az alsó keret stílusát állítja be	
`border-width`	A négy keret szélességének beállítása egyetlen deklarációban, egytől négy értéket tartalmazhat	Megadható a következő tulajdonságok értéke: *border-left-width, border-right-width, border-top-width, border-bottom-width*
`border-left-width`	A bal keret szélességét állítja be	Megadható értékek: ***thin***, ***medium***, ***thick***, vagy a távolság megadása abszolút illetve százalékos mennyiségként, alapértelmezett érték: ***medium***
`border-right-width`	A jobb keret szélességét állítja be	
`border-top-width`	A felső keret szélességét állítja be	
`border-bottom-width`	Az alsó keret szélességét állítja be	

Példák a használatra

```
p#example0 {border-top-width: thin}

p#example1 {
    border-top-width: 1px;
    border-right-width: 4px;
    border-bottom-width: 2px;
    border-left-width: 3px;
}
```

Ugyanez a beállítás összevont deklarációval:

```
p#example2 {
    border-width: 1px 4px 2px 3px;
}
```

Ilyen összevon deklaráció esetén az adatok értelmezése a következő:

- 4 paraméter esetén a sorrend: felső, bal, jobb, alsó
- 3 paraméter esetén a sorrend: *felső*, *bal* és *jobb* egyenlő, *alsó*
- 2 paraméter esetén a sorrend: *felső* és *alsó* egyenlő, *jobb* és *bal* egyenlő
- 1 paraméter esetén: mind a négy egyenlő a megadott értékkel

Az alábbi példában a bal és a jobb oldali szegély vastagsága egyaránt 4em.

```
border-width: 2em, 4em, 3em;
```

A vonalstílusokat érdemes egyenként kipróbálni. Az összevont vonalstílus (border-style) szintaxisa ugyanazt a logikát követi, mint amit a vastagság beállításánál már láttunk. Például a

```
border-style: solid dashed dotted;
```

szabály szerint a felső szegély folytonos, a bal és a jobb oldali szaggatott, az alsó pontozott vonal.

Ha a szegély színét nem adjuk meg, akkor a böngésző szövegszínnel azonos színűnek tekinti. Az összevont szegélyszín (border-color) szintaxisa is a vonalvastagságnál ismertetett szabályt követi a paraméterek sorrendjének értelmezésében.

Ha a lehetséges négy közül egy bizonyos szegély összes tulajdonságát egy összevont szabályban akarjuk megadni, akkor a paraméterek sorrendje: *vastagság*, *stílus*, *szín*. Például

```
border-left: 2px solid darkblue;
```

A paraméterek száma háromnál kevesebb is lehet. A hiányzó adatot itt is az elem helyén érvényes alapértelmezés, vagy végső soron a böngésző alapértelemezése pótolja. Például a

```
border-left: solid darkblue;
```

szabály esetében a szegélyvastagság *medium* lesz, mert ez a szabály alapértelmezett értéke. Ügyeljünk arra, hogy a vonalstílus alapértelmezése *none*, tehát ha elhagyjuk a vonalstílus értékét a szabályból, és azt külön sem állítjuk be, akkor a szegély nem jelenik meg a böngészőben.

Végül nézzük a *border* tulajdonságot, amely szegély összes tulajdonságának az összevont deklarálására alkalmas. A paraméterek és a sorrendjükre vonatkozó szabály ugyanaz, mint amit a *border-left* példa kapcsán az előbb áttekintettünk. Tehát a paraméterek: *vastagság*, *stílus* és *szín*. A különbség annyi, hogy most az adatok mind a négy szegélyvonalra érvényesek. A *border* szabályt tehát jellemzően akkor alkalmazzuk, ha mind a négy oldal szegélyét egyformára kell beállítani. Példa:

```
p#example2 { border: 1px solid black; }
```

CSS 3 új szegély tulajdonságok

A CSS 3 szabvány hivatalosan még nem jelent meg, de az újabb böngésző verziók (közte az Internet Explorer 9, Google Chrome) már támogatják pl. azokat a szegély tulajdonságokat, amelyek az igényes megjelenésű weblapok készítéséhez szükségesek, és amelyeknek egyes böngészőknél

(pl. Mozilla) már voltak előzményei. Az alábbiakban a lekerekített sarkú szegélyezés eszközeit mutatjuk be.

Tulajdonság	Szerep	Értékek
`border-radius`	A szegély mind a négy sarkán a lekerekítés sugarának megadása összevont deklarációban	Megadható a következő tulajdonságok értéke: *border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius, border-top-left-radius*
`border-top-left-radius`	A bal felső sarok sugarának beállítása	A sugár, mit távolság megadása
`border-top-right-radius`	A jobb felső sarok sugarának beállítása	
`border-bottom-right-radius`	A jobb alsó sarok sugarának beállítása	
`border-bottom-left-radius`	A bal alsó sarok sugarának beállítása	

Példa:

```
#rounded {
    border: thin solid blue;
    border-top-left-radius: 2em;
    border-top-right-radius: 2em;
    width: 250px;
    height: 100px;
    background-color: lightsteelblue;
    padding: 2em;
    text-align: justify;
}
```

A stílust egy `<div id=”rounded">` szakaszban alkalmazva az eredmény:

A CSS 3 szabvány hivatalosan még nem jelent meg, de az újabb böngésző verziók már támogatják pl. azokat a szegély tulajdonságokat, amelyek az igényes megjelenésű weblapok készítéséhez szükségesek,

8.4.8 Belső margók

Tulajdonság	Leírás	Értékek
`padding`	Az összes oldal belső margóját állítja be egy összevont deklarációban	Megadható a következő tulajdonságok értéke: *padding-top, padding-right, padding-bottom, padding-left*
`padding-bottom`	Az alsó oldal belső margójának beállítása	A távolság megadása abszolút illetve százalékos mennyiségként
`padding-left`	A bal oldal margójának beállítása	
`padding-right`	A jobb oldal margójának beállítása	
`padding-top`	A felső oldal margójának beállítása	

Példák a használatra:

```
p {
    padding-top: 10px;
    padding-right: 0;
    padding-bottom: 1.2em;
    padding-left: 5%;
}
```

Ugyanezeket a beállításokat összevontan is megadhatjuk a *padding* tulajdonsággal.

```
p {padding: 10px 0 1.2em 5%;}
```

Látható, hogy szintaxisban itt is a szegélyeknél bemutatott logika érvényesül: a doboz felső részétől jobbra haladva megyünk körbe a paraméterek megadásával. Érvényes ez a hasonlóság a négynél kevesebb paraméter értelmezésére is. Például a

```
padding: 10px 5px 10px;
```

szabály a jobb és a bal oldali belső margót 5-5 pixelre állítja.

8.4.9 Listák

A lista logikai szerkezetét továbbra is az `<ul>` vagy `<ol>`, valamint a `<li>` címkék segítségével írjuk le, de a formai tulajdonságokat stílusokkal határozzuk meg, mellőzve az említett HTML elemek formai paramétereit.

Tulajdonság	Leírás	Értékek
`list-style`	Az összes lista tulajdonság állítása egyetlen deklarációban	Megadható a következő tulajdonságok értéke: *list-style-type, list-style-position, list-style-image*
`list-style-type`	A lista tételei előtt szereplő szimbólum típusát határozza meg	Megadható értékek: ***none, disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-alpha, upper-alpha, lower-greek, lower-latin, upper-latin, armenian, georgian***
`list-style-image`	Lista szimbólumnak képet állít be	***none*** vagy *url a képfájlhoz*
`list-style-position`	A lista szimbólumainak helyzetét jelöli ki	***inside***, ***outside***

A `list-style-type` egyes értékeinek magyarázata

Érték	A szimbólum	CSS 1
none	Nincs szimbólum	*
disc	Kitöltött kör	*
circle	Kör	*
square	Négyzet	*
decimal	Decimális szám	*
decimal-leading-zero	Szám, az elején nullákkal kiegészítve (01, 02, stb.)	
lower-roman	Kisbetűs latin szám (i, ii, iii, iv, v, stb.)	*
upper-roman	Nagybetűs latin szám (I, II, III, IV, V, stb.)	*
lower-alpha	Kisbetű (a, b, c, d, e, stb.)	*
upper-alpha	Nagybetű (A, B, C, D, E, stb.)	*
lower-greek	Kis görög betű (alpha, beta, gamma, stb.)	
lower-latin	Latin kisbetű (a, b, c, d, e, stb.)	
upper-latin	Latin nagybetű (A, B, C, D, E, stb.)	
armenian	Tradicionális örmény szám	
georgian	Tradicionális gregorianus szám (an, ban, gan, etc.)	

Egyes böngészők (pl. Internet Explorer a 6-os verzióig) csak a CSS 1 szabvány szerinti értékeket ismerik.

A `list-style-position` tulajdonság szabályozza, hogy a szimbólum a felsorolt tételek területén belül, vagy azon kívül (az előtt) helyezkedik-e el.

Példák a listák formázására stílusok segítségével:

```
li {
    list-style-type: square;
}
```

Ehhez például a HTML dokumentum törzse lehet a következő:

```
<ul>
  <li>Margó</li>
  <li>Szegély</li>
  <li>Belső margó</li>
</ul>
```

Mivel a lista stílus öröklődik a következő szabály ugyanezzel a hatással jár volna:

```
ul {
    list-style-type: square;
}
```

Az alábbi példában a felsorolás szimbólum kép

```
#article ul {
    font-family: sans-serif;
    list-style-image: url(../images/gomb.gif);
}
```

```
<ul id=”article”>
  <li>Margó</li>
  <li>Szegély</li>
  <li>Belső margó</li>
</ul>
```

A következő példa az összevont deklarációt mutatja, majd a felsorolás szimbólum letiltását láthatjuk.

```
    list-style: disc outside url(‘download.gif’);
    list-style-type: none;
```

8.4.10 Táblázatok

A HTML 4.01 szabvány a `<table>` elemnek a megjelenést szabályozó paramétereit nem minősítette kerülendőnek, ezért e paramétereket használhatjuk továbbra is. Ugyanakkor a stíluslapok a táblázatok formázásában is ugyanazokat az előnyöket biztosítják, amelyeket a többi HTML elem esetében láttuk, és ha egyszer rászántuk magunkat a stíluslapok használatára, akkor ésszerű dokumentumaink táblázatait is így megvalósítani. Hangsúlyozni kell, hogy a táblázat szerkezetét a tanult HTML címkék segítségével kell definiálni, a CSS csak a formai tulajdonságok alakításánál jön szóba!

A korábbi dokumentum elemeknél áttekintett CSS tulajdonságok – természetesen a megfelelő helyen alkalmazva – a táblázatok formázásában használhatók. Nemcsak a szövegformázásra gondolunk itt, hanem pl. a *border* tulajdonságra a szegélyek formázásához, a *padding* tulajdonságokra a *cellpadding* helyett, a *width* vagy a *color* tulajdonságokra is. Az alábbi táblázatban néhány olyan CSS tulajdonság látható, amelyek kifejezetten a táblázatokra szabottak:

Tulajdonság	Szerep	Érték
`caption-side`	Megadja, hogy a tábla melyik oldalán (felül vagy alul) jelenjen meg a cím (caption)	Megadható értékek: ***top***, ***bottom***
`table-layout`	Ha a táblázat számára előzetesen lefoglalt helyen a tartalom nem fér el, akkor a böngészők gyakran megnövelik a táblázat méretét. Ezzel a tulajdonsággal kikényszeríthető, hogy a böngésző az általunk megadott méreteket alkalmazza.	Megadható értékek: ***auto***, ***fixed*** alapértelmezett: *auto*
`border-collapse`	Két szegély között a térköz megjelenítése (separate) vagy eltávolítása (collapse).	Megadható értékek: ***collapse***, ***separate*** alapértelmezett: *separate*
`border-spacing`	A táblázat cellái között megjelenő térközt szabályozza mindkét irányban.	Megadható értékek: *vízszintes távolság* és *függőleges távolság* alapértelmezett: 0 0
`empty-cells`	Meghatározza, hogy az üres cellák szegélyvonal látszódjon-e vagy sem.	Megadható értékek: ***show***, ***hide*** alapértelmezett: *show*
`vertical-align`	A cella tartalmának függőleges igazítása	Megadható értékek: ***top***, ***baseline***, ***middle***, ***bottom***

Szegélyezés

Alapértelmezés szerint a táblázatok szegély nélkül jelennek meg. Ha csak külső szegélyt akarunk, akkor a szegély tulajdonságot csak a `<table>` címkéhez kell kapcsolni. Példa:

```
table { border: 2pt solid darkblue; }
```

Természetesen az összes tanult szegély tulajdonságot használhatjuk, s így akár egyenként is formázhatjuk a négy szegélyvonalat. Ha csak a cellák szegélyeit (minden cella körbe szegve) akarjuk láttatni, akkor a szabály lehet pl. a következő:

```
table * { border: 2pt solid darkblue; }
```

Ha a táblázat- és a cellaszegélyeket együtt akarjuk, akkor a szabály:

```
table, table * { border: 2pt solid darkblue; }
```

Ha a külső szegélyt eltérő stílussal akarjuk formázni, akkor különítsük el a szabályokat! A következő példában szemléltetjük a *border-collapse* hatását is. Ha két táblázatcella érintkezik, akkor alapértelmezés szerint mindkettő látszik, tehát 0 cellatérköz esetén is dupla szegélyeket látunk. Ezzel a tulajdonsággal megadhatjuk, hogy a böngésző mindkettőt megjelenítse, vagy vonja össze azokat. Ez utóbbi esetben azt a szegélyt jeleníti meg a böngésző, amely vizuálisan a legjobban elkülönítő hatású. Próbáljuk ki az alábbi stílusok hatását a *border-collapse* nélkül is!

```
table {
    border: 3pt solid blue;
    border-collapse: collapse;
}
td, th {
    border: 1pt solid blue;

}
```

A bal oldali kép a *border-collapse* nélkül mutatja a stílusok hatását.

Hónap	Forgalom (ezer Ft)	
	Fehér Holló Étterem	Édes Élet Bár
Január	31000	23000
Február	35000	26000
Március	41000	21000

Hónap	Forgalom (ezer Ft)	
	Fehér Holló Étterem	Édes Élet Bár
Január	31000	23000
Február	35000	26000
Március	41000	21000

A *border-spacing* tulajdonsággal szabályozható a szegélyek távolsága. Például

```
    border-spacing: 10px 5px;
```

Sajnos az Internet Explorernek még a 7-es verziója sem ismeri ezt. Ha tekintettel akarunk lenni erre, akkor kénytelenek vagyunk mégiscsak a HTML eszközeit használni (pl. `<table cellspacing="10">`).

Függőleges és vízszintes igazítás

A feladatot a *text-align* és a *vertical-align* tulajdonságokkal oldhatjuk meg. A text-align a korábban tanult módon vízszintesen igazítja a szöveget, illetve a cella tartalmát. Ha egy felsőbb szintű elemhez definiáljuk, akkor az alsóbb szintűek öröklik a beállítást. Így például a

```
    table { text-align: center; }
```

az egész táblázatra érvényesíti a középre igazítást. Természetesen egy alsóbb szintű címkéhez ettől még kapcsolhatunk más igazítást, ha a tábla szintű beállítás valahol nem alkalmas. Például

```
    th { text-align: left; }
```

vagy

```
#thisright { text-align: right;}
```

majd a táblázatban a megfelelő celláknál:

```
<td id=”thisright”> Valami cellatartalom </td>
```

A *vertical-align* tulajdonság nem öröklődik automatikusan az alsóbb szintű elemekre. Csak arra az elemre érvényes, amelyhez deklaráltuk. Ez azt jelenti, hogy minden cellánál külön meg kell adni a stílusát? Általános értelemben igen, de a stílus definíciók strukturálásával és a kiválasztók célszerű használatával radikálisan csökkenthető az ismétlődő stíluskijelölés terhe. Például

```
.total {
    margin-left: auto; /* Igazítsuk középre a táblázatot! */
    margin-right: auto;
    vertical-align: top;
    border: 3pt solid blue;
}
.total th {
    vertical-align: middle;
    border: 1pt solid blue;
    height: 3em;
}
.total td {
    vertical-align: bottom;
    border: 1pt solid blue;
    height: 3em;
}
```

Ezt követően már csak egy helyen, a `<table>` címkében kell feltüntetni a stílusosztályt: `<table class="total">`. Ezt követően a fejlécre és a cellákra már automatikusan a megfelelő stílusváltozatot alkalmazza a böngésző.

A *.total* stílusból azt is látjuk, hogy a bal- és jobbmargó automatikusra állítása itt is a teljes blokk középre igazítását eredményezi.

Egy táblázatformázási feladat

A feladat részletes megfogalmazása még hiányzik!

Az első öt hónap forgalmi adatai

Hónap	Forgalom (ezer Ft)	
	Fehér Holló Étterem	Édes Élet Bár
Január	31000	23000
Február	35000	26000
Március	41000	21000
Április	45000	27000
Május	48000	26000

Egy lehetséges megoldás
Stílusok:

```
/* A táblázatra mint egészre vonatkozóan */
table.forgalom {
    margin-left: auto;
    margin-right: auto;
    width: 550px;      /* A teljes táblázat szélessége */
    border: 3px solid blue;
    border-collapse: collapse;
    font-family: Arial, sans-serif;
    font-size: 12pt;
    text-align: center;
}
table.forgalom * {
    vertical-align: middle;
    height: 2em;
}
```

```
table.forgalom th, td {
    border: 1px solid lightblue;
}
/*** A táblázatcím formázásához***/
table.forgalom caption {
    font-size: 16pt;
    font-weight: bold;
    color: mediumblue;
}
/*** A fejléc formázásához ***/
table.forgalom th {
    background-color: lavender;
    color: mediumblue;
}
table.forgalom .th-honap {
    width: 100px;            /* Mivel ez az első cella az 1. oszlopban, ez lesz az oszlopszélesség is */
    border-bottom: 3px solid blue;
}
table.forgalom .th-etterem {
    width: 250px;            /* Mivel ez az első cella a 2. oszlopban, ez lesz az oszlopszélesség is */
    border-bottom: 3px solid blue;
}
table.forgalom .th-bar {
    border-bottom: 3px solid blue; /* A 3. oszlop szélessége a maradék lesz */
}
/*** A táblázattörzs celláinak formázásához ***/
table.forgalom td {
    padding: 0 0 0 10px;
    color: dimgray;
}
table.forgalom .td-honap {
    text-align: left;
}
/* A táblázattörzs sorainak formázásához */
table.forgalom .even {
    background-color: beige;
}
table.forgalom .odd {
    background-color: antiquewhite;
}
```

A HTML törzsben ehhez a következő tartozik:

```
<table class="forgalom">
  <caption>
  Az első öt hónap forgalmi adatai
  </caption>
   <tr>
    <th class="th-honap" rowspan="2">Hónap</th>
    <th colspan="2">Forgalom (ezer Ft)</th>
   </tr>
   <tr>
    <th class="th-etterem">Fehér Holló Étterem</th>
    <th class="th-bar">Édes Élet Bár</th>
   </tr>
   <tr class="odd">
    <td class="td-honap">Január</td>
    <td>31000</td>
    <td>23000</td>
   </tr>
   <tr class="even">
    <td class="td-honap">Február</td>
    <td>35000</td>
    <td>26000</td>
```

```
   </tr>
   <tr class="odd">
    <td class="td-honap">Március</td>
    <td>41000</td>
    <td>21000</td>
   </tr>
   <tr class="even">
    <td class="td-honap">Április</td>
    <td>45000</td>
    <td>27000</td>
   </tr>
   <tr class="odd">
    <td class="td-honap">Május</td>
    <td>48000</td>
    <td>26000</td>
   </tr>
</table>
```

A túlbonyolítás lehetősége miatt *nagyon lényeges annak az alapos átgondolása, hogy az egyes formai tulajdonságokat a stílusok fájának melyik pontján állítjuk be*. Végül megjegyzendő, hogy a 7.1.4 pontban megemlített lehetőségek sokkal hatékonyabbá teszik a táblázatok stílusokkal történő formázását is.

8.5 Elemek pozícionálása

Korábban már szóltunk arról, hogy a tisztán HTML alapú weblap tervezés csak a táblázatokkal lehetett megoldani azt, hogy az oldal egyes tartalmi elemei, mint pl. menük, cikkek, hirdetések, meghatározott helyre kerüljenek. A CSS merőben új lehetőségeket nyújt az elemek pozícionálásához. Aki stíluslapokat használ, az felejtse el a táblázatos módszert! A táblázatokat használjuk arra, amire kitalálták: adatok áttekinthető közzétételére.

Az alábbi táblázat a pozícionáláshoz alkalmazható legfontosabb tulajdonságokat foglalja össze

Tulajdonság	Szerep	Érték
`top`	Az elem tetejének helyzetét állítja be	***auto***, vagy a távolság megadása abszolút illetve relatív (%, em) mennyiségként alapértelmezett: *auto*
`right`	Az elem jobb szélének helyzetét állítja be	
`bottom`	Az elem aljának helyzetét állítja be	
`left`	Az elem bal szélének helyzetét állítja be	
`position`	Beállítja az adott elem pozícionálásának módját.	Megadható értékek: ***static, relative, absolute, fixed*** alapértelmezett*: stati*c
`float`	Lehetővé teszi, hogy az egyes dobozokat a normál elhelyezéstől eltérve egymás mellé tegyük („körülfolyatás”).	Megadható értékek: ***left, right, none*** alapértelmezett: *none*
`clear`	Szabályozni lehet segítségével, hogy az elem körül hogyan helyezkedjenek el a környező elemek.	Megadható értékek: ***left, right, none***
`display`	Segítségével megváltoztatható egy elem soron belüli vagy blokk jellege.	Megadható értékek: ***block, inline, none***
`visibility`	Szabályozza a doboz tartalmának láthatóságát. *hidden* esetén a doboz és tartalma nem látható, de a helye megőrződik	Megadható értékek: ***visible, hidden*** alapértelmezett: *visible*

8.5.1 Pozícionálási mód

Alapértelmezés szerint a HTML dokumentum blokk szintű elemei az oldal tetejétől az alja felé *egymás alá* „áramlanak” be a böngésző ablakába. A soron belüli (inline) elemek pedig balról jobbra „folyva” töltik ki a helyet. (Lásd a doboz-modellről szóló részt!) A *position* tulajdonsággal ez a viselkedés módosítható.

Ha a *position* értéke *static*, akkor lényegében az előbb leírt normál viselkedés (normal flow) szerint kerül helyére az adott elem doboza. Ha abszolút pozícionálást, azaz *absolute* értéket állítunk be,

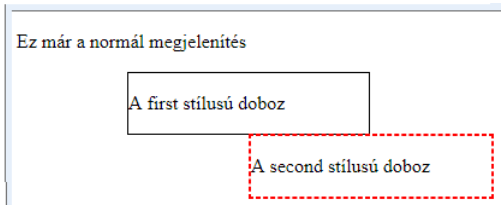
akkor a dobozt a bal felső sarok koordinátáinak megadásával a dokumentum egy meghatározott helyére tehetjük. Tehát az abszolút pozícionálással kivesszük az elemet a normál megjelenítési módból, de ez azt is jelenti, hogy a rákövetkező elem a normál megjelenítést követve oda kerül, ahol az abszolút helyzetű elem helye lett volna. Fontos, hogy a pozíció koordinátái mindig a befoglaló elem (bal felső sarkának) pozíciójától számítandók! Példa:

```
.first {
    position: absolute;
    left: 100px;
    top: 50px;
    width: 200px;
    border: 1px solid black;
}
.second {
    position: absolute;
    left: 100px;
    top: 50px;
    width: 200px;
    border: 2px dashed red;
}
```

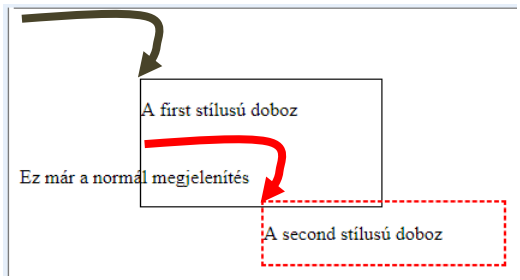
Alkalmazzuk e stílusokat az alábbi módon a HTML dokumentum törzsében!

```
<div class="first">
  <p>A first stílusú doboz</p>
  <div class="second">
    <p>A second stílusú doboz</p>
  </div>
</div>
<p>Ez már a normál megjelenítés</p>
```

Láthatjuk, hogy a böngésző a harmadikként szereplő bekezdést normál megjelenítés szerint az elejére tette, a *second* stílusú doboz helyzetét pedig a *first* dobozának bal felső sarkától mérte:



Ha az elemet relatív módon pozícionáljuk, azaz a *position* érékét *relative*-ra állítjuk, akkor ahhoz a helyhez viszonyítva adhatjuk meg a helyzetét, ahol a normál megjelenítés szerint a helye lenne. Tehát mintegy odébb tehetjük az elemet, miközben az benne marad a normál megjelenítési folyamatban. Például, ha az előbbi példában az *absolute* helyére mindenütt a *relative* kerül, akkor az eredmény:



A *first* stílusú az első, de helyéről eltolva. Alatta a *second* stílusú a második, de helyéről ugyancsak eltolva. A helye azonban üresen maradt, s a harmadik bekezdés pontosan ezen üres hely alá került.

Végül a negyedik lehetőség az, hogy az elemet a böngésző ablakában egy bizonyos helyre rögzítjük. Ezt a `position: fixed;` szabállyal tehetjük meg. A doboz akkor is a helyén marad, ha a felhasználó a gördítősávot használja.

8.5.2 A helyzet megadása

A *top, right, bottom, left* tulajdonságok az elem helyzetének előírására szolgálnak. Ahogy az imént láttuk, működésük a *position* tulajdonságtól függ. A `top: 0; left: 0;` beállítás az elemet a befoglaló elem bal felső sarkában helyezi el. A dobozt tároló `<div>` elem helyzetének meghatározásához természetesen elegendő egyik sarkának helyzetét megadni, továbbá szükség szerint az elem szélességét és/vagy magasságát. A kiterjedés megadása nélkül általában a megjelenítendő tartalom szabja meg az elem méretét. A *top* és *bottom* értéke együtt meghatározza a magasságot, a *right* és *left* értéke a szélességet.

8.5.3 Elemek egymás mellett

Körülfolyatás

A *float* tulajdonság lehetővé teszi, hogy az adott elemet kivegyük a normál megjelenítési folyamatból és a befoglaló elem bal vagy jobb széléhez illesszük. A befoglaló elem tartalma ekkor az elem körül helyezkednek majd el, ha ehhez van elegendő hely.

A tulajdonság lehetséges értékeinek jelentése:

Érték	Értelmezése
left	Az elem a befoglaló elem bal széléhez (margójához) illeszkedik, a befoglaló elem tartalmak körülötte (jobbra) helyezkednek el
right	Az elem a befoglaló elem jobb széléhez (margójához) illeszkedik, a befoglaló elem tartalmak körülötte (balra) helyezkednek el
none	Az elem nem lesz körülfolyatva, marad normál megjelenítésben

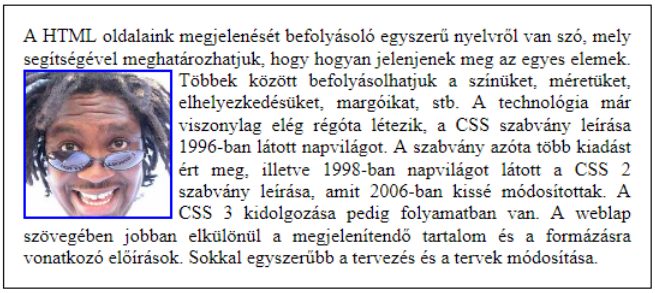
Ha a *float* tulajdonságot használjuk, akkor az elem szélességét is célszerű beállítani, mert enélkül doboza a rendelkezésre álló szélességet 100%-ban kihasználja, s így nem marad hely mellette más elemek számára.

A *float* együtt jár azzal, hogy a függőleges irányú margóknál (tehát alatta és felette) nincs „margóelnyelés” (ld. 8.4.6), mindkét margó megmarad.

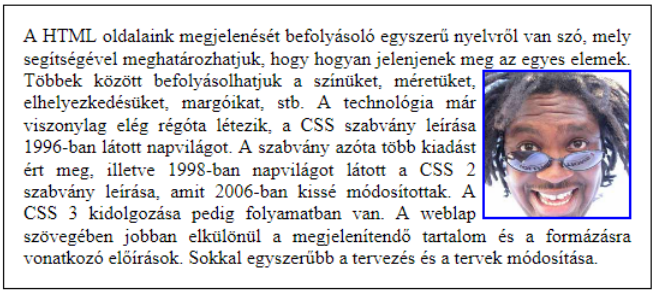
A következő példában szöveg blokkba teszünk egy képet. Stílusok:

```
img {
    float: left;
    border: 2px solid blue;
    width: 120px;
    height: 120px;
    margin-right: 5px;
}
p {
    width: 40%;
    border: 1px solid black;
    margin: 10px;
    padding: 1em;
    text-align: justify;
}
```

Az eredmény az `<img>` címkének a szövegben elfoglalt helyétől függően lehet például a következő:



Ha a körülfolyatást a `float: right;` szabállyal végezzük, akkor az eredmény:



Több *float* tulajdonságú elem is egymás mellé helyezhető, amennyiben van ehhez elég széles hely.

Előfordulhat, hogy a *float* tulajdonságú elem mellett nem akarunk megjeleníteni semmi mást. Ekkor a *clear* tulajdonsággal letiltható az adott oldalon a megjelenítés. A *clear* értékeinek magyarázata:

Érték	Értelmezése
left	Az elem bal oldalán tiltja tartalom megjelenítését. (A másik elem *float: left* beállítását hatástalanítja)
right	Az elem jobb oldalán tiltja tartalom megjelenítését. (A másik elem *float: right* beállítását hatástalanítja)
both	Az elem mindkét oldalán tiltja tartalom megjelenítését
none	Nincs tiltás.

A *clear* működésének pontos megértéséhez vegyük a következő általános példát! Ha egy „**A**” elemnek nincs `clear: left;` tulajdonsága, akkor egy „**B**” elem, amelynek `float:left;` tulajdonsága van, az „A” elem mellé kerül, annak a jobb oldalára. Ha azonban „**A**” elemnek megvan a `clear: left;` tulajdonsága, akkor „**B**” elem hiába rendelkezik a `float:left;` tulajdonsággal, ezt az előtte levő „**A**” elem *clear* tulajdonsága *kikapcsolja*, ezért nem mellé, hanem alá kerül a „**B**”.

A blokk vagy soron belüli jelleg módosítása

Tudjuk, hogy a blokk szintű elemek dobozai mindig új sorban kezdődnek és egymás alá kerülnek a normál megjelentés szabályai szerint. Ezzel szemben a soron belüli elemek dobozait a böngésző balról jobbra haladva egymás mellé helyezi.

Ha egy elemnek az alapértelmezett kezelését meg akarjuk változtatni, akkor a *display* tulajdonságot kell használnunk. Erre többnyire akkor van szükség, ha kisebb blokk szintű elemeket egymás mellé, vagy soron belüli elemeket egymás alá akarunk helyezni. Az alábbi példában listából egyszerű *vízszintes menüt* készítünk a display segítségével.

```
ul#navbar {
    padding-left: 0;
    list-style-type: none;   /* Ne rakjon ki listaszimbólumokat */
    font-family: arial, sans-serif;
}
/* Kerüljenek egymás mellé a lista elemei és legyen az egész elem kattintható! */
ul#navbar li {
    display:inline;
}
```

```
/* A hiperhivatkozás jellemzői */
ul#navbar a {
    border: 1px dashed steelblue;
    border-bottom: none;
    padding: 5px 15px 5px 15px;
    margin-right: 5px;
    background-color: midnightbue;
    color: midnightblue;
    text-decoration: none; /* A link aláhúzásának kikapcsolása */
}
ul#navbar a:hover {          /* Ha a kurzor a link fölé kerül, váltson színt és betűstílust*/
    background-color: antiquewhite;
    font-style: italic;
}
```

A HTML törzsben lehet például a következő:

```
<ul id="navbar">
<li><a href="lapok/kiallitas.html">Kiállítás</a></li>
<li><a href="lapok/mozi.html">Mozi</a></li>
<li><a href="lapok/szinhaz.html">Színház</a></li>
<li><a href="lapok/koncertek.html">Koncertek</a></li>
</ul>
```

8.6 Példák elhelyezési tervek megvalósítására

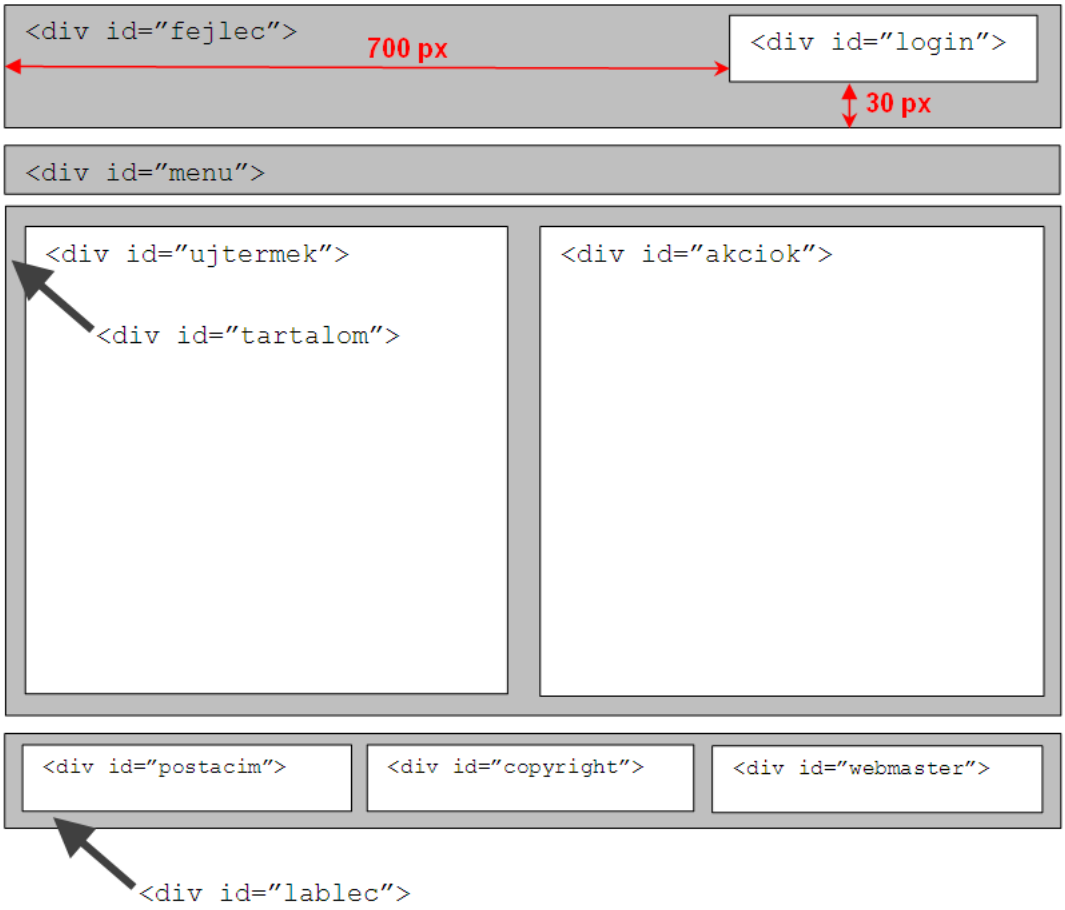
9.6.1 Az elhelyezési tervek típusai

Tervezési módszerek

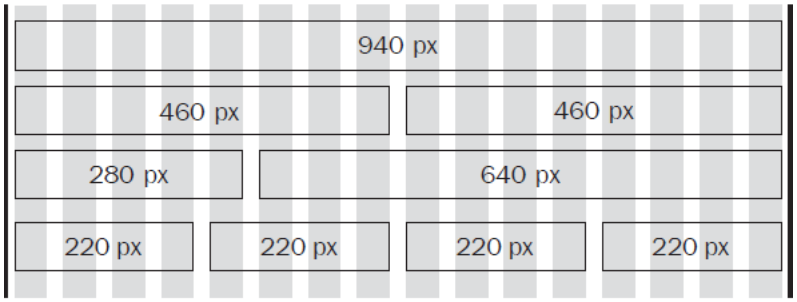
A stíluslapok kettős szerepet játszanak a weblapok készítésében. Az egyik szerep a tartalom elemeinek formázása, hogy azok szépek és jól érthetők legyenek. Eddig elsősorban a CSS ezen képességeivel foglalkoztunk. A másik szerep azonban legalább ennyire fontos: jól áttekinthető és vonzó módon rendezhetjük el a böngésző ablakában a tartalom különböző részeit. Amikor a weblapot készítünk, először nyilván azt kell eldönteni, hogy mi mindent akarunk a weblapon közzétenni, és milyen közönség számára. A következő kérdés az, hogy hogyan kellene mindezt a látogató számára megjeleníteni. E kérdés megválaszolása az *elhelyezési terv* (lay out) kialakításával kezdődik. Ez tartalmazza a felület tervrajzát, rajta a vizuálisan is elkülönülő területek a méreteikkel, színeikkel, stb. Olyan ez, mint egy építész számára az épület alaprajza. A HTML és a CSS eszközeivel – többek között a *float*, az abszolút és relatív pozícionálás, külső illetve belső margók, szegélyek stb. – a weblapot kódoló informatikus ezt a tervet valósítja meg.

A weblapok szerkezeti felépítésében olyan szokások alakultak ki, amelyeket nekünk is érdemes figyelembe venni. Például a nyitó oldal tetején, a *fejlécen* (header) kiemelt megjelenítéssel látható a webhely neve, ami lehet egy cég vagy szervezet neve, esetleg logóval kiegészítve. Gyakran mindez egy a teljes oldalt átfogó, képet vagy animációt is tartalmazó szalagon (banner) helyezkedik el. Az oldalt az ugyancsak teljes szélességű *lábléc* (footer) zárja, amely megnevezi a szerzői jogok tulajdonosát (copyright), esetleg a kapcsolatfelvételhez szükséges adatokat is tartalmaz. A kettő között helyezzük el azokat a *tartalmakat*, amelyekért érdemes az oldalt meglátogatni. Ezt a részt is kisebb-nagyobb részekre oszthatjuk. A leggyakoribb megoldás a papíralapú sajtóból átvett *hasábok* alkalmazása. A képernyő méretét többszörösen meghaladó oldalak a látogató számára nehezen áttekinthetők, a keresett információ megtalálása hosszú időt igényelhet és nélkülözhetetlen a kényelmetlen föl-le görgetés. Ezért a tartalmat szinte mindig több oldalra szétbontva célszerű közölni. Ebből következik, hogy az oldalakon – a nyitóoldalon mindenképpen – olyan *navigáló* elemeket kell elhelyezni, amelyekkel a kívánt oldal könnyen elérhető. Ezek lényegében hiperhivatkozások, amelyek például vízszintes vagy függőleges menük, nyomógombok formájában jelennek meg.

A terven célszerű szerepeltetni a blokkokat, amelyek az oldal dobozait hordozzák, feltüntetve méreteiket, pozíciójukat, a hozzájuk kapcsolódó alapvető stílusok neveit. Egy terv „kezdeményre” mutat példát az alábbi ábra:



Jó módszer egy megfelelő osztású rácson megrajzolni az egyes dobozokat. Ehhez felhasználhatunk egy grafikai szoftvert, amelyben a rács megjeleníthető és a dobozok könnyen áthelyezhetők, átméretezhetők. Alább egy 960 pixel széles oldal tervének egy részlete látható.



Változó szélességű (liquid) tervek

A változó szélességű tervek a meghatározó mérteket és pozíciókat nem pixelekben, hanem relatív módon, elsősorban a rendelkezésre álló térköz százalékában adják meg. Ha a felhasználó növeli, vagy csökkenti a böngészőablak méretét, illetve kisebb vagy nagyobb felbontású képernyőn nézi az oldalt, akkor az egyes dobozok terjedelme változik. Hátránya azonban az ilyen típusú weblapoknak, hogy nehéz úgy megtervezni az oldalt, hogy erősen eltérő ablakméretek illetve felbontások esetén is a tervező szándékainak megfelelően nézzen ki. A *min-width* és *max-width* tulajdonságokkal korlátozhatjuk a méretváltozást egy olyan tartományra, amelyben az elemek szépek és a tartalom sem torzul.

Az ilyen tervezésnél alapvetően a normál megjelenítés szabályait hagyjuk működni, de – ahol szükséges – a *float* tulajdonsággal a dobozokat egymás mellé rakjuk, és ilyen módon hasábokat alakítunk ki különböző tartalmak számára. Alább (ld. 8.6.2) példát is láthatunk egy három hasábos vázlatos terv megvalósítására.

Rögzített szélességű (fixed) tervek

A tervezők többsége szívesebben dolgozik állandó szélességű oldalakkal, mivel a látvány és a viselkedés pontosabban tervezhető. Kevésbé kell aggódni a monitorok változó felbontása vagy a változó ablakméret miatt, mert a terv eleve azzal számol, hogy ha az oldal nem fér el, akkor a látogatónak görgetnie kell.

Természetesen az oldalt úgy kell tervezni, hogy a látogatók többségének a képernyőjén a teljes méretű böngésző ablakban elférjen az oldal, s így ne legyen szükség oldal irányú görgetésre. Ma azzal számolhatunk, hogy a képernyők többsége alkalmas az *1024x768* képpontos megjelenítésre. Az 1024 képpontos szélességből le kell vonnunk a böngészők ablakának keretét. Így kb.960-980 képpontnyi hellyel lehet tervezni. Általánosan elterjedt a *960 képpont* szélesség alkalmazása.

A terven az alapvető méreteket mindenképpen pixelekben adjuk meg. Az oldalt az ablakon belül, illetve a belső elemeket a külsőkön belül középre a bal és a jobb margó *auto* értékűre állításával igazítjuk középre.

Sajnos akár változó, akár rögzített szélességű tervet készítünk, számítanunk kell arra, hogy még az elterjedt böngészők sem egységesen jelenítik meg az oldalainkat. Ezért többféle böngészővel illetve azok több verziójával kell tesztelni és hiba esetén utána kell nézni, hogyan lehet az adott böngésző hibájának következményeit kiküszöbölni.

8.6.2 Változó szélességű, három hasábos elhelyezési terv

Az alábbi tervben egy vízszintes menünek terveztünk helyet. A három oszlop egyforma széles, de más-más stílusnevet rendelünk hozzájuk, így később az arányuk könnyen módosítható. A méretek kalkulálásakor tartsuk szem előtt a doboz modell ábráját!

A terv megvalósításából a HTML rész

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitionalt//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" />
<html>
<head>
<title>3 oszlopos változó szélességű lap</title>
```

```
<link type="text/css" rel="stylesheet" href="layout-liquid_3col.css" />
</head>
<body>
  <div class="column123">
    <div id="header">Fejléc</div>
    <div id="navigation">Navigáció</div>
    <div class="column1">Hasáb I.</div>
    <div class="column2">Hasáb II.</div>
    <div class="column3">Hasáb III.</div>
    <div id="footer">Lábléc</div>
  </div>
</body>
</html>
```

A stíluslapot tartalmazó *layout-liquid_3col.css* fájl tartalma:

```
#header {
    background-color: #cccccc;
    height: 120px;
    padding: 10px;
}
#navigation {
    background-color: #dfdfdf;
    height 40px;
    padding: 10px;
}
#footer {
    background-color: #cccccc;
    height 40px;
    padding: 10px;
    clear: both;
    border-top: 20px solid white;
}
.column1, .column2, .column3 {
    float: left;
    width: 28%;
    margin-left: 2.5%;
    background-color: #cccccc;
    padding: 1%;
    margin-top: 20px;
    height: 175px;
    min-width: 210px;
    max-width:400px;
}
.column123 {
    min-width:750px;
}
```

8.6.3 Rögzített szélességű három hasábos elhelyezési terv

A fenti háromhasábos tervnek egy 960 pixel szélesre tervezett és kissé módosított változatát valósíthatjuk meg például az alábbi módon.

A HTML rész:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitionalt//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" />
<head>
<title>3 column layout</title>
<link type="text/css" rel="stylesheet" href="layout-fixed_3col.css" />
</head>
<body>
  <div id="frame">
    <div id="page">
      <div id="header">Fejléc</div>
      <div id="navigation">Navigáció</div>
```

```
      <div class="column1">Hasáb I.</div>
      <div class="column2">Hasáb II.</div>
      <div class="column3">Hasáb III.</div>
      <div id="footer">Lábléc</div>
    </div>
  </div>
</body>
</html>
```

A változatosság kedvéért háttérképet is alkalmazunk. A *frame* blokk természetesen nem HTML keretet jelent, a név csak arra utal, hogy ez fogja össze az oldal egészét. A stíluslap:

```
body {
    margin 0px;
    background-color: antiquewhite;
    font-family: arial, verdana, sans-serif;
    text-align: center;
}
#frame {
    margin-left: auto;
    margin-right: auto;
    text-align: left;
    width: 960px;
}
#page {
    padding: 0px, 10px, 10px, 10px;
    background-image: url("../pic/bkgrd.jpg");
}
#header {
    background-color: darksalmon;
    height: 120px;
    padding: 10px;
}
#navigation {
    background-color: limegreen;
    height 40px;
    padding: 10px;
}
#footer {
    background-color: darksalmon;
    height 40px;
    padding: 10px;
    clear: both;
    border-top: 20px solid white;
}
.column1, .column2, .column3 {
    float: left;
    width: 286px;
    background-color: palegoldenrod;
    padding: 10px;
    margin-top: 20px;
    height: 200px;
}
.column1, .column2 {
    margin-right: 21px;
}
```

8.6.4 Navigációs elemek

A menük a weblapok legfontosabb elemei közé tartoznak, egyben interaktív elemekként lehetőséget adnak arra is, hogy az oldalt változatosabbá, érdekesebbé tegyük segítségükkel. Ennek megfelelően nagyon sokféle, szép és látványos megoldást láthatunk az interneten böngészve, de bőségesen kaphatunk tippeket, kész megoldásokat is, ha egy kis időt szánunk a keresésre az interneten. Az alábbiakban csak néhány egyszerűbb technikát mutatunk be.

Függőleges, nyomógomb jellegű menük

A menü kiválasztható pontjai mindig hiperhivatkozások, amelyek az adott oldal valamelyik könyvjelzővel megjelölt részére, vagy egy másik oldalra mutatnak. A menüt ezért célszerű hivatkozások listájának tekinteni. Ha igényeink nem túl nagyok, akkor ezzel meg is elégedhetünk. Ha elegánsabb megoldást szeretnénk, akkor megpróbálhatjuk a lista elemeit színezéssel, szegélyezéssel nyomógombszerűen kialakítani. Ennek a feltételei:

- A lista elemeinek felsorolás szimbólumai ne jelenjenek meg, mert itt zavaróak. Megoldás: a `list-style-type: none;` szabállyal tiltsuk le a szimbólum kiírását.
- Ne csak a lista elemének szövege legyen kattintásra „érzékeny”, hanem az egész doboza. Megoldás: a `display: block;` szabály az `<a>` elemet blokk jellegűvé alakítja, s ezzel a felsorolás teljes `<li>` eleme kattintásra érzékennyé válik, nemcsak a benne lévő link szövege.
- A hivatkozás szövegét alapértelmezés szerint a böngésző aláhúzással is kiemeli. Ez most szintén zavaró, le kellene tiltani. Megoldás: az `<a>` elemre állítsuk be a `text-decoration: none;` szabályt.

Ne feledkezzünk el arról sem, hogy a *:hover* pszeudo osztály által adott lehetőségről, amit felhasználhatunk arra, hogy a kurzor alatt lévő menüpont megjelenési módját megváltoztassuk.

Az első példa a szegélyekkel és a betű stílusával játszva mozgás és 3D élményét kelti. A stílusok:

```
#navbar {
    list-style-type: none;
    padding-left: 0px;
    margin-left: 0;
    font-family: arial, sans-serif;
}
#navbar a {
    display: block;       /* Kattintható legyen az egész blokk egyben */
    width: 10em;
    height: 1.25em;
    line-height: 1.25em;
    zoom: 1;              /* Egy Internet Explorer hiba ellen*/
    border-left: 25px solid steelblue;
    border-top: 1px solid steelblue;
    border-right: 1px solid steelblue;
    border-bottom: 1px solid steelblue;
    border-top-right-radius: 8px;
    border-bottom-right-radius: 5px;
    background-color: midnightblue;
    color: white;
    padding-left: 20px;
    padding-top: 5px;
    padding-bottom: 5px;
    text-decoration: none;
}
#navbar a:hover {
    background-color: steelblue;
    border-right: 3px solid midnightblue;
    font-style: italic;
}
```

A HTML dokumentum törzsében a menü helyén lehet pl. a következő:

```
<ul id="navbar">
<li><a href="lapok/kiallitas.html">Kiállítás</a></li>
<li><a href="lapok/mozi.html">Mozi</a></li>
<li><a href="lapok/szinhaz.html">Színház</a></li>
<li><a href="lapok/koncertek.html">Koncertek</a></li>
</ul>
```

Megjegyzés: az Internet Explorer régebbi verzióiban az általunk használt display beállítást a böngésző nem teljesen a leírt módon hajtja végre, ezért érdemes a HTML dokumentum <head> szakaszába az alábbi feltételes kódot beépíteni:

```
<!--[if IE 5]>
    <style>
    #navholder a {
        height: 1em;
        float: left;
        clear: both;
        width: 100%;
    }
</style>
<![endif]-->

<!--[if IE 6]>
    <style>
    #navholder a { height: 1em; }
</style>
<![endif]-->
```

A következő menü arra mutat példát, hogyan használjunk felsorolás szimbólumként képet, illetve a kép váltogatásával hogyan tehetjük érdekessé és mozgalmassá a menüt.

```
#navholder {
    float: left;
    width: 200px;
    font-family: Geneva, Arial, Helvetica, sans-serif;
    font-size: 0.8em;
    list-style-type: none;
}
#navholder ul {
    margin: 0px;
    padding: 0px;
}
#navholder li {
    margin: 0px;
}
#navholder a {
    display: block;
    background-image: url(../pic/bullet1.gif);
    background-repeat: no-repeat;
    background-position: 0% 50%;
    background-color: lightskyblue;
    padding: 4px 4px 4px 25px;
    border-right: 1px solid #000000;
    border-left: 1px solid #666666;
    border-bottom: 1px solid #000000;
}
#navholder a:link, #navholder a:visited {
    color: #FFFFFF;
    text-decoration: none;
}
#navholder a:hover, #navholder a:active {
    color: #000000;
    background-image: url(../pic/bullet2.gif);
    background-repeat: no-repeat;
    background-position: 0% 50%;
    text-decoration: none;
}
#sellink a:link, #sellink a:visited, #sellink a:hover, #sellink a:active {
    color: #000000;
    background-image: url(../pic/bullet1.gif);
    background-repeat: no-repeat;
    background-position: 0% 50%;
```

```
    text-decoration: none;
}
```
A HTML törzsben az előbbihez képest csak az <ul> stílusazonosítója változik a következő módon:
`<ul id="navholder">`

Vízszintes, nyomógomb jellegű menük és füles lapozók

Legyen az első feladat az előbbi *navholder* stílusú menü vízszintes változatának elkészítése. A stílusokon alig kell változtatni, és a kód átvizsgálása után egyszerűsíteni is lehetne.

```
#navholder {
    float: left;
    width: 100%;
    font-family: Geneva, Arial, Helvetica, sans-serif;
    font-size: 0.8em;
    list-style-type: none;
}
#navholder ul {
    margin: 0px;
    padding: 0px;
}
#navholder li {
    float: left;
    white-space: nowrap;
    margin: 0 0 1em 0;
    padding: 0;
}
#navholder a {
    background-image: url(../pic/bullet1.gif);
    background-repeat: no-repeat;
    background-position: 0% 50%;
    background-color: lightskyblue;
    padding: 4px 50px 4px 25px;
    border-right: 1px solid #000000;
    border-left: 1px solid #666666;
    border-bottom: 1px solid #000000;
}
#navholder a:link, #navholder a:visited {
    color: #FFFFFF;
    text-decoration: none;
}
#navholder a:hover, #navholder a:active {
    color: #000000;
    background-image: url(../pic/bullet2.gif);
    background-repeat: no-repeat;
    background-position: 0% 50%;
    text-decoration: none;
}
#sellink a:link, #sellink a:visited, #sellink a:hover, #sellink a:active {
    color: #000000;
    background-image: url(../pic/bullet1.gif);
    background-repeat: no-repeat;
    background-position: 0% 50%;
    text-decoration: none;
}
```
A HTML kódban ismét csak az `<ul>` stílusának azonosítóját kell cserélni az előző példához képest.

A 8.5.3 pont végén ismertetett vízszintes menüből kis módosítással ún. füles (tabbed) lapozót készíthetünk. Két példa:

Arra azonban ügyelni kell, hogy a kerekített sarkú szegélyeket csak a legújabb böngészők tudják megjeleníteni! Ezt ki lehet kerülni, ha a füleket képekkel helyettesítjük. Ezt a technikát azonban itt már nem tárgyaljuk. (Lásd *http://www.alistapart.com/articles/slidingdoors*. Továbbá hasznos eszközöket kaphatunk menük alkalmazásához a következő helyen: *http://www.exploding-boy.com/2005/12/15/free-css-navigation-designs*)

A) Függelék (Web-biztos színek és neveik)

aliceblue 240,248,255 F0F8FF	darkslategray 47,79,79 2F4F4F	lightsalmon 255,160,122 FFA07A	palevioletred 219,112,147 DB7093
antiquewhite 250,235,215 FAEBD7	darkturquoise 0,206,209 00CED1	lightseagreen 32,178,170 20B2AA	papayawhip 255,239,213 FFEFD5
aqua 0,255,255 00FFFF	darkviolet 148,0,211 9400D3	lightskyblue 135,206,250 87CEFA	peachpuff 255,239,213 FFEFD5
aquamarine 127,255,212 7FFFD4	deeppink 255,20,147 FF1493	lightslategray 119,136,153 778899	peru 205,133,63 CD853F
azure 240,255,255 F0FFFF	deepskyblue 0,191,255 00BFFF	lightsteelblue 176,196,222 B0C4DE	pink 255,192,203 FFC0CB
beige 245,245,220 F5F5DC	dimgray 105,105,105 69,69,69	lightyellow 255,255,224 FFFFE0	plum 221,160,221 DDA0DD
bisque 255,228,196 FFE4C4	dodgerblue 30,144,255 1E90FF	lime 0,255,0 00FF00	powderblue 176,224,230 B0E0E6
black 0,0,0 000000	firebrick 178,34,34 B22222	limegreen 50,205,50 32CD32	purple 128,0,128 800080
blanchedalmond 255,255,205 FFFFCD	floralwhite 255,250,240 FFFAF0	linen 250,240,230 FAF0E6	red 225,0,0 FF0000
blue 0,0,255 0000FF	forestgreen 34,139,34 228B22	magenta 255,0,255 FF00FF	rosybrown 188,143,143 BC8F8F
blueviolet 138,43,226 8A2BE2	fuchsia 255,0,255 FF00FF	maroon 128,0,0 800000	royalblue 65,105,225 4169E1
brown 165,42,42 A52A2A	gainsboro 220,220,220 DCDCDC	mediumaquamarine 102,205,170 66CDAA	saddlebrown 139,69,19 8B4513
burlywood 222,184,135 DEB887	ghostwhite 248,248,255 F8F8FF	mediumblue 0,0,205 0000CD	salmon 250,128,114 FA8072
cadetblue 95,158,160 5F9EA0	gold 255,215,0 FFD700	mediumorchid 186,85,211 BA55D3	sandybrown 244,164,96 F4A460
chartreuse 127,255,0 7FFF00	goldenrod 218,165,32 DAA520	mediumpurple 147,112,219 9370DB	seagreen 46,139,87 2E8B57
chocolate 210,105,30 D2691E	gray 128,128,128 808080	mediumseagreen 60,179,113 3CB371	seashell 255,245,238 FFF5EE
coral 255,127,80 FF7F50	green 0,128,0 008000	mediumslateblue 123,104,238 7B68EE	sienna 160,82,45 A0522D
cornflowerblue 100,149,237 6495ED	greenyellow 173,255,47 ADFF2F	mediumspringgreen 0,250,154 00FA9A	silver 192,192,192 C0C0C0
cornsilk 255,248,220 FFF8DC	honeydew 240,255,240 F0FFF0	mediumturquoise 72,209,204 48D1CC	skyblue 135,206,235 87CEEB
crimson 220,20,60 DC143C	hotpink 255,105,180 FF69B4	mediumvioletred 199,21,133 C71385	slateblue 106,90,205 6A5ACD

cyan 0,255,255 00FFFF	indianred 205,92,92 CD5C5C	midnightblue 25,25,112 191970	slategray 112,128,144 708090
darkblue 0,0,139 00008B	indigo 75,0,130 4B0082	mintcream 245,255,250 F5FFFA	snow 255,250,250 FFFAFA
darkcyan 0,139,139 008B8B	ivory 255,240,240 FFF0F0	mistyrose 255,228,225 FFE4E1	springgreen 0,255,127 00FF7F
darkgoldenrod 184,134,11 B8860B	khaki 240,230,140 F0D58C	moccasin 255,228,181 FFE4B5	steelblue 70,130,180 46,82,B4
darkgray 169,169,169 A9A9A9	lavender 230,230,250 E6E6FA	navajowhite 255,222,173 FFDEAD	tan 210,180,140 D2B48C
darkgreen 0,100,0 006400	lavenderblush 255,240,245 FFF0F5	navy 0,0,128 000080	teal 0,128,128 008080
darkkhaki 189,183,107 BDB76B	lawngreen 124,252,0 7CFC00	oldlace 253,245,230 FDF5E6	thistle 216,191,216 D8BFD8
darkmagenta 139,0,139 8B008B	lemonchiffon 255,250,205 FFFACD	olive 128,128,0 808000	tomato 253,99,71 FF6347
darkolivegreen 85,107,47 556B2F	lighblue 173,216,230 ADD8E6	olivedrab 107,142,35 6B8E23	turquoise 64,224,208 40E0D0
darkorange 255,140,0 FF8C00	lightcoral 240,128,128 F08080	orange 255,165,0 FFA500	violet 238,130,238 EE82EE
darkorchid 153,50,204 9932CC	lightcyan 224,255,255 E0FFFF	orangered 255,69,0 FF4500	wheat 245,222,179 F5DEB3
darkred 139,0,0 8B0000	lightgoldenrodyellow 250,250,210 FAFAD2	orchid 218,112,214 DA70D6	white 255,255,255 FFFFFF
darksalmon 233,150,122 E9967A	lightgreen 144,238,144 90EE90	palegoldenrod 238,232,170 EEE8AA	whitesmoke 245,245,245 F5F5F5
darkseagreen 143,188,143 8FBC8F	lightgrey 211,211,211 D3D3D3	palegreen 152,251,152 98FB98	yellow 255,255,0 FFFF00
darkslateblue 72,61,139 483D8B	lightpink 255,182,193 FFB6C1	paleturquoise 175,238,238 AFEEEE	yellowgreen 154,205,50 9ACD32

B) Függelék (Speciális karakterek kódolása)

Egy HTML dokumentumban szükség lehet olyan karakterek megjelenítésére is, amely különlegesnek számít, mert a billentyűzeten fel sem lelhető, vagy eltér az oldal alapértelmezett kódlapjának karakterkészletétől. A HTML/XHTML szabványok lehetővé teszik, hogy meghatározott karaktereket speciális kóddal jelöljünk. A böngésző majd e kód alapján rajzolja ki a megfelelő jelet.

Az ilyen karaktereket kétféle módon kódolhatjuk:

- *számjegyekkel*, vagy
- *névvel*

A számjegyekkel történő kódolás szintaxisa: `&#nnnn;`
ahol az `nnnn` helyében decimális számjegyek egy sorozata állhat. Például a `&#216;` kód a **Ø** karaktert jelöli.

A névvel történő kódolás szintaxisa: `&név;`
ahol a `név` helyébe a karakter szabványos nevét kell beírni. Például a `&Oslash;` kód ugyancsak a **Ø** karaktert jelöli.
Mindkét kódolási módnál fontos a lezáró *pontosvessző*!

Az alábbiakban csoportosítva közöljük a szabványos karakterkódokat.

A HTML/XHTML nyelveben speciális jelentésű karaterek

Jel	Számkód	Név
&	&#38;	&amp;
>	&#62;	&gt;
<	&#60;	&lt;
"	&#34;	&quot;

Kiemelésre, hangsúlyok jelölésére használt karakterek

Jel	Számkód	Név
´	&#180;	&acute;
¸	&#184;	&cedil;
ˆ	&#710;	&circ;
¯	&#175;	&macr;
·	&#183;	&middot;
˜	&#732;	&tilde;
¨	&#168;	&uml;

Nemzeti ábécék speciális betűi

Jel	Számkód	Név	Jel	Számkód	Név	Jel	Számkód	Név
Á	&#193;	&Aacute;	ð	&#240;	&eth;	Õ	&#213;	&Otilde;
á	&#225;	&aacute;	Ë	&#203;	&Euml;	õ	&#245;	&otilde;
Â	&#194;	&Acirc;	ë	&#235;	&euml;	Ö	&#214;	&Ouml;
â	&#226;	&acirc;	Í	&#205;	&Iacute;	ö	&#246;	&ouml;
Æ	&#198;	&AElig;	í	&#237;	&iacute;	Š	&#352;	&Scaron;
æ	&#230;	&aelig;	Î	&#206;	&Icirc;	š	&#353;	&scaron;
À	&#192;	&Agrave;	î	&#238;	&icirc;	ß	&#223;	&szlig;
à	&#224;	&agrave;	Ì	&#204;	&Igrave;	Þ	&#222;	&THORN;
Å	&#197;	&Aring;	ì	&#236;	&igrave;	þ	&#254;	&thorn;
å	&#229;	&aring;	Ï	&#207;	&Iuml;	Ú	&#218;	&Uacute;
Ã	&#195;	&Atilde;	ï	&#239;	&iuml;	ú	&#250;	&uacute;
ã	&#227;	&atilde;	Ñ	&#209;	&Ntilde;	Û	&#219;	&Ucirc;
Ä	&#196;	&Auml;	ñ	&#241;	&ntilde;	û	&#251;	&ucirc;
ä	&#228;	&auml;	Ó	&#211;	&Oacute;	Ù	&#217;	&Ugrave;

Jel	Számkód	Név	Jel	Számkód	Név	Jel	Számkód	Név
Ç	Ç	Ç	ó	ó	ó	ù	ù	ù
ç	ç	ç	Ô	Ô	Ô	Ü	Ü	Ü
É	É	É	ô	ô	ô	ü	ü	ü
é	é	é	Œ	Œ	Œ	Ý	Ý	Ý
Ê	Ê	Ê	œ	œ	œ	ý	ý	ý
ê	ê	ê	Ò	Ò	Ò	ÿ	ÿ	ÿ
È	È	È	ò	ò	ò	Ÿ	Ÿ	Ÿ
è	è	è	Ø	Ø	Ø			
Ð	Ð	Ð	ø	ø	ø			

Elválasztó és központozó jelek

(néhányuk szerepe tipográfiai ismeretek nélkül nehezen érthető)

Jel	Számkód	Név	Magyarázat
¢	¢	¢	cent
¤	¤	¤	pénznem, valuta
€	€	€	euro
£	£	£	Font
¥	¥	¥	yen és yuan
¦	¦	¦	
•	•	•	
©	©	©	copyright
†	†	†	
‡	‡	‡	
⁄	⁄	⁄	
…	…	…	
¡	¡	¡	
ℑ	ℑ	ℑ	
¿	¿	¿	
	‎	‎	balról jobbra jel
—	—	—	
–	–	–	
¬	¬	¬	
‾	‾	‾	felülvonás
ª	ª	ª	
º	º	º	
¶	¶	¶	bekezdésjel
‰	‰	‰	
′	′	′	aposztróf
″	″	″	idézőjel
ℜ	ℜ	ℜ	
®	®	®	registered trade mark
	‏	‏	jobbról balra jel
§	§	§	
			ún. lágy elválasztó
¹	¹	¹	
™	™	™	trade mark
℘	℘	℘	

Jel	Számkód	Név	Magyarázat
„	„	„	
«	«	«	
“	“	“	
‹	‹	‹	
‘	‘	‘	
»	»	»	
”	”	”	
›	›	›	
’	’	’	
‚	‚	‚	
			em térköz
			en térköz
			nem törhető szóköz
			keskeny szóköz
	‍	‍	nulla térközzel csatkozó
	‌	‌	térköz karakterek közel tartására

Matematikai és technikai jelölésekhez

Jel	Számkód	Név	Jel	Számkód	Név
°	°	°	⊇	⊇	⊇
÷	÷	÷	∴	∴	∴
½	½	½	Α	Α	Α
¼	¼	¼	α	α	α
¾	¾	¾	Β	Β	Β
≥	≥	≥	β	β	β
≤	≤	≤	Χ	Χ	Χ
−	−	−	χ	χ	χ
²	²	²	Δ	Δ	Δ
³	³	³	δ	δ	δ
×	×	×	Ε	Ε	Ε
ℵ	ℵ	ℵ	ε	ε	ε
∧	∧	∧	Η	Η	Η
∠	∠	∠	η	η	η
≈	≈	≈	Γ	Γ	Γ
∩	∩	∩	γ	γ	γ
≅	≅	≅	Ι	Ι	Ι
∪	∪	∪	ι	ι	ι
∅	∅	∅	Κ	Κ	Κ
≡	≡	≡	κ	κ	κ
∃	∃	∃	Λ	Λ	Λ
ƒ	ƒ	ƒ	λ	λ	λ
∀	∀	∀	Μ	Μ	Μ
∞	∞	∞	μ	μ	μ
∫	∫	∫	Ν	Ν	Ν
∈	∈	∈	ν	ν	ν
⟨	〈	⟨	Ω	Ω	Ω

Jel	Számkód	Név	Jel	Számkód	Név
⌈	⌈	⌈	ω	ω	ω
⌊	⌊	⌊	Ο	Ο	Ο
∗	∗	∗	ο	ο	ο
µ	µ	µ	Φ	Φ	Φ
∇	∇	∇	φ	φ	φ
≠	≠	≠	Π	Π	Π
∋	∋	∋	π	π	π
∉	∉	∉	ϖ	ϖ	ϖ
⊄	⊄	⊄	Ψ	Ψ	Ψ
⊕	⊕	⊕	ψ	ψ	ψ
∨	∨	∨	Ρ	Ρ	Ρ
⊗	⊗	⊗	ρ	ρ	ρ
∂	∂	∂	Σ	Σ	Σ
⊥	⊥	⊥	σ	σ	σ
±	±	±	ς	ς	ς
∏	∏	∏	Τ	Τ	Τ
∝	∝	∝	τ	τ	τ
√	√	√	Θ	Θ	Θ
⟩	〉	⟩	θ	θ	θ
⌉	⌉	⌉	ϑ	ϑ	ϑ
⌋	⌋	⌋	ϒ	ϒ	ϒ
·	⋅	⋅	Υ	Υ	Υ
∼	∼	∼	υ	υ	υ
⊂	⊂	⊂	Ξ	Ξ	Ξ
⊆	⊆	⊆	ξ	ξ	ξ
∑	∑	∑	Ζ	Ζ	Ζ
⊃	⊃	⊃	ζ	ζ	ζ

Figurák és nyilak

Jel	Számkód	Név
↵	↵	↵
↓	↓	↓
⇓	⇓	⇓
↔	↔	↔
⇔	⇔	⇔
←	←	←
⇐	⇐	⇐
→	→	→
⇒	⇒	⇒
↑	↑	↑
⇑	⇑	⇑
♣	♣	♣
♦	♦	♦
♥	♥	♥
♠	♠	♠
◊	◊	◊

C) Függelék (Beágyazott tartalmak: az iframe címke)

Ha egy hosszabb dokumentumot akarunk az oldalon elhelyezni, de korlátozott területen, önálló görgetősávval ellátva, akkor segíthet a beágyazott keret *(inline frame)*, azaz `<iframe>` elem használata. Még gyakoribb alkalmazási helyzet az, amikor egy másik webhelyről akarunk egy oldalt vagy egy videót beágyazni a saját dokumentumunkba. Külön előnye, hogy a beágyazott keret tartalma a befogadó dokumentum újratöltése nélkül cserélődik, ha pl. abban egy linkre kattint a felhasználó. A beágyazott keret a HTML dokumentumban bárhol elhelyezkedhet és a képhez hasonlóan körülfolyatható szöveggel, beállíthatunk szegélyeket és margókat a keret körül.

Az `<iframe>` egyetlen kötelező paramétere az src, amelynek értéke annak a lapnak az URL-je, amelyet az `<iframe>` segítségével be akarunk ágyazni a dokumentumba. Természetesen többnyire érdemes a height illetve a width paraméterekkel a keret szükséges méretét is meghatározni.

Az `<iframe>` paraméterei

`name` – a keret neve

Segítségével nevet adhatunk a keretnek. Ez akkor hasznos, ha meg akarjuk határozni, hogy egy bizonyos tartalom melyik keretben jelenjen meg. Különösen fontos akkor, amikor a dokumentumunkban olyan hivatkozásokat (linkeket) akarunk készíteni, amelyekkel különböző tartalmakat meghatározott keretekben akarunk megjeleníteni. Ilyenkor a kereteknek nevet adunk a name paraméterrel, majd a hivatkozás (link) target paraméterében a nevével azonosítjuk azt a keretet, amelyben a hivatkozott tartalomnak be kell töltődni. Példa:

```
name="pictures"
```

`align` – a keret igazítása

Megadja, hogyan jelenjen meg a kereten kívüli szöveg. Lehetséges értékek

Érték	Hatása
left	A kerete a bal margóhoz igazított, a szöveg jobbra helyezkedhet el, körbefolyhatja.
right	A keret a jobb margóhoz igazított, a szöveg balra helyezkedhet el, körbefolyhatja.
top	A keret teteje egy sorban lesz a körülötte lévő szöveggel.
middle	A keret közepe egy sorban lesz a körülötte lévő szöveggel.
bottom	A keret alja egy sorban lesz a körülötte lévő szöveggel. (Alapértelmezett)

`height` és `width` – a keret magassága és szélessége

A képek elhelyezéséhez hasonlóan pixelekben mérve vagy százalékkosan megadhajtuk a keret mérteteit. Példák

```
height="250" width="30%"
```

`frameborder` – a szegély vastagsága

Segítségével megadható, hogy a keret szegélye látható legyen-e vagy sem. Az értéke pixelekben adja meg a szegély vastagságát. A "0" érték azt jelenti, hogy ne legyen szegély.

`longdesc` – hosszabb leírás csatolása

Lehetővé teszi egy hivatkozás (link) megadását egy másik lapra, amelyen egy szöveges leírás található olyasmiről, aminek egyébként a keretben kellene látszódni. Ez különösen akkor hasznos, ha olyan ábrát, diagramot teszünk a keretbe, amelyet az oldal nem minden látogatója képes megjeleníteni. Ugyancsak hasznos lehet ez a paraméter, ha a felhasználó számára a *frame* betöltése jelent nehézséget. Példa:

```
longdesc=”../leiras/iframe1.html”
```

`marginheight` és **`marginewidth`** – belső margók

A *marginewidth* értékével pixelekben számolva megadhatjuk a térközt a bal illetve jobb oldali szegély és a tartalom között. A *margineheight* hasonlóan a felső illetve alsó szegély és a tartalom közötti térközt határozza meg. Példa:

```
marginewidth="10" margineheight="10"
```

`scrolling` – **gördítősáv beállítása**

Segítségével előírhatjuk, hogy a beágyazott keretnek legyen-e gördítősávja. Lehetséges értékei: "*no*" illetve "*yes*" Példa:

```
scrolling="yes"
```

Feladat: az oldalon 3 képet akarunk megjeleníteni, de egyesével, ugyanazon a helyen, mindig csak azt, amelyiket a felhasználó kiválaszt. Egy megoldás:

```
<html>
<head><title>Képek</title></head>
<body>
<h1>Válassz a három kép közül!</h1>
<p><a href="pic/jec.jpg" target="kepek">Jec a programozó" </a> </p>
<p><a href="pic/mdominguez.jpg" target="kepek">Dominguez az életművész" </a> </p>
<p><a href="pic/yshimizu.jpg" target="kepek">A mindig vidám Yshimizu" </a> </p>
<iframe name="kepek" height="260" width="260" frameborder="0" scrolling="no">
A képek megjelenési helye.
</iframe>
</body>
</html>
```

Felhasznált irodalom

Füstös János: Word Wide Web HTML 4.0
SZAK Kiadó Kft. 1998.

David Sawyer McFarland: CSS The Missing Manual
O'Reilly, 2009. Second Edition

Jon Ducket: HTML, XHTML, CSS and JavaScript
Wiley Publishing, Inc. 2010.

Richard York: Beginning CSS: Cascading Style Sheets for Web Design
Wiley Publishing, Inc. 2005.