CSS
IV. KÖNYV: HALADÓ SZINT

I. Kijelölések csoportosítása és egymásba ágyazása

Kijelölések csoportosítása

A stílus-oldalakban kijelölt elemek gyakran azonos formázást kapnak, pl:

```
h1
{
color:green;
}
h2
{
color:green;
}
p
{
color:green;
}
```

A kód lerövidítésére a kijelöléseket csoportosíthatjuk. A kijelölés-csoport tagjait vessző választja el. Az alábbi példában a fenti CSS kijelöléseinek csoportosítását láthatjuk:

```
<html>
<head>
<style type="text/css">
h1,h2,p
{
color:green;
}
</style>
</head>

<body>
<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Kijelölések egymásba ágyazása (nesting)

Egy adott kijelölésbe (pl. Csoportba vagy id-be) tartozó elemeket is külön kijelöléssel formázhatunk.

Az alábbi példában a dokumentum összes **<p>** elemére megadott stílustól eltérőt adunk meg a **marked** osztályba tartozó bekezdésekre:

```
<html>
<head>
<style type="text/css">
p,h3 {color:green;text-align:center;}
.marked {background-color:red;}
.marked p {color:white;}
</style>
</head>

<body>
```

```
<p>This is a green, center-aligned paragraph.</p>

<div class="marked">
<h3>This Header3 is green.</h3>
<p>This p element should not be green.</p>
</div>

<p>p elements inside a "marked" classed element keeps the alignment
style, but has a different text color.</p>
</body>
</html>
```

Mint látjuk, az első meghatározás szerint az oldal összes **<p>** és **<h3>** eleme zöld. A második meghatározás értelmében az összes **marked** részbe tartozó elem háttér-színe piros lesz; ezek közül pedig a bekezdések betűszíne is változik: fehérre.

Ugyanez a helyzet, hogyha az egyetlen ** **-et nem osztályként, hanem csak **id**-ként azonosítjuk:

```
<div id=''marked''></div>
```

és a stílus-oldalba pontok helyett kettőskeresztet írunk:

```
<style>
p,h3 {color:green;text-align:center;}
#marked {background-color:red;}
#marked p {color:white;}
</style>
```

Kijelölések csoportosítása és egymásba ágyazása:

Kijelölés-típusok:

- **Elem:** *elem*
- **Név:** *#név*
- **Csoport:** *.csoport*
- **Adott nevű elem(ek):** *elem#név*
- **Adott csoportba tartozó elem(ek):** *elem.csoport*

Többszörös kijelölés: *elem1, elem2, elem3 {meghatározás:érték;}*
A kijelölt háromféle elem azonos formázást kap.

Specifikus kijelölés (kijelölések egymásba ágyazása): *elem1 elem2 elem3 {meghatározás:érték;}*
Itt minden, **elem1**-ben lévő **elem2**-be tartozó **elem3** a meghatározott formázást kapja.

II. CSS – méretezés

A CSS méretező-jellemzők az elemek magasságának és szélességének beállítására szolgálnak.

Példák

Az alábbiakban példákon keresztül fogjuk áttekinteni a CSS-méretezési lehetőségeket.

Első példánkban a különféle elemek magasságának beállítását látjuk:

```
<html>
<head>
<style type="text/css">
img.normal {height:auto;}
img.big {height:120px;}
p.ex {height:100px;width:100px;}
</style>
</head>

<body>
<img class="normal" src="logocss.gif" width="95" height="84" /><br />

<img class="big" src="logocss.gif" width="95" height="84" />

<p class="ex">The height and width of this paragraph is 100px.</p>

<p>This is some text in a paragraph. This is some text in a paragraph.
This is some text in a paragraph. This is some text in a paragraph.
This is some text in a paragraph. This is some text in a paragraph.</p>
</body>
</html>
```

Példánkban tehát két kép és két bekezdés szerepel. Az első kép mérete automatikus, azaz megfelel a képfájl eredeti méretezésének. A második esetben ugyanezen képet manuálisan megmagasítottuk (függőlegesen elnyújtottuk). Az első bekezdésnek mind szélessége, mind magassága 100px. (A második alapértelmezett.) Hogyha az első bekezdés méreteit 0;0-nak vesszük (**p.ex** kijelölés), akkor minden egyes szava egymás alatt látszik, és a második bekezdés rögtön az első szó (és sor) legalján kezdődik, azaz az összezsugorított bekezdés és az utána következő tartalom egymásra torlódik (akkor is, hogyha kép, vagy más elem van mögötte).

Második példánkban az elemmagasság %-os beállítását láthatjuk:

```
<html>
<head>
<style type="text/css">
img.normal {height:auto;}
img.big {height:22%;}
img.small {height:10%;}
</style>
</head>

<body>
<img class="normal" src="logocss.gif" width="95" height="84" />
<img class="big" src="logocss.gif" width="95" height="84" />
<img class="small" src="logocss.gif" width="95" height="84" />
</body>
</html>
```

A három, egymás melletti kép elsője eredeti méretű; a második közel azonos nagyságú, a harmadik pedig kisebb. Hogyha a második kép magasságát 100%-nak választjuk, akkor mérete közelítőleg az oldal (frame) magasságával lesz egyező.

Harmadik példánkban az elem-szélesség px-ben való megadását láthatjuk:

```
<html>
<head>
<style type="text/css">
img {width:200px;}
</style>
</head>
<body>

<img src="logocss.gif" width="95" height="84" />
</body>
</html>
```

Az alapéretlmezésben megadott elem-szélességet itt az

```
img {width:200px;}
```

stílus felülírja, az az **<img>** elemben nem változtatható, ellentétben a magassággal. Ez érthető, hiszen az **<img>** elembe írt **width** és **height** attribútumok nem stílus-kijelölések, hanem önálló attribútumok. Tehát ez a két adat azért nem írja felül (helyi szinten) a fejrészben megadott stílust, mert nem stílus-attribútumként lett megadva. Hogyha az oldalt átalakítjuk:

```
<html>
<head>
<style type="text/css">
img {width:200px;}
</style>
</head>
<body>

<img src="logocss.gif" style="width:400px;height:10px" />
</body>
</html>
```

akkor az **<img>**-be írt **style** attribútummal már a kép szélességét is szabályozhatjuk, és a fejrészbe írt adat (az adott elemre nézbe) teljesen hatástalan marad.

Negyedik példánkban egy elem maximális magasságának megadását látjuk:

```
<html>
<head>
<style type="text/css">
p {max-height:100px;max-width:350px;}
</style>
</head>

<body>
<p>The maximum height of this paragraph is set to 100px.
The maximum height of this paragraph is set to 100px.
The maximum height of this paragraph is set to 100px.
The maximum height of this paragraph is set to 100px.
The maximum height of this paragraph is set to 100px.
</p>
</body>
</html>
```

Itt a **max-width** jellemzővel az elem szélességét korlátlanul (ill. a leghosszabb szó méretéig) változtathatjuk; a magasság tekintetében azonban meglehetős tehetetlenséget jelent, hogy a böngészőnek az összes szót ki kell írnia; tehát legfeljebb a sor- és szóköz tekintetében volna mozgástere az előírt érték betartásához.

Ötödik példánkban egy bekezdés maximális szélességének %-ban való megadásával találkozunk:

```
<html>
<head>
<style type="text/css">
p {max-width:50%;}
</style>
</head>
<body>

<p>This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>
</body>
</html>
```

A megjelenő szöveg szélessége a frame-nek 50%-a. A **max-width** attribútum értékének növelésére ill. csökkentésére a szöveg-szélesség a szóközöknél való tördeléssel megváltozik.

Következő példánkban az elemek minimális magasságára (pixelekben) vonatkozó beállításokat láthatunk:

```
<html>
<head>
<style type="text/css">
body p {min-height:100px;}
#first {background-color:red;}
#first p {background-color:red;min-height:200px;}
#first h3 {min-height:300px;}
#second {color:blue;background-color:brown;min-height:400px;}
</style>
</head>

<body>
<div id="first">
<h3>The minimum height of this header is set to 100px.</h3>
<p>The minimum height of this paragraph is set to 200px.</p>
<p id="second">The minimum height of this paragraph is set to
200px.</p>
<p>The minimum height of this paragraph is set to 200px.</p>
<p>The minimum height of this paragraph is set to 200px.</p>
<p>The minimum height of this paragraph is set to 200px.</p>
</div>

<p id="second">The minimum height of this paragraph is set to
400px.</p>

<p>The minimum height of this paragraph is set to 100px.</p>
<p>The minimum height of this paragraph is set to 100px.</p>
<p>The minimum height of this paragraph is set to 100px.</p>
</body>
</html>
```

A kijelölések itt mindig az adott elemre vonatkoznak; azaz hogyha **body** kijelöléssel beállítunk egy 50px-es minimális magasságot, az nem azt jelenti, hogy a szövegtestbe tartozó összes elem minimális magassága ekkora, hanem hogy maga a szövegtest ilyen magas.
A fenti esetben tehát a szövegtestbe tartozó bekezdések 100px magasak, fekete betű- és fehér háttérszínűek. A **first** id-jű részbe tartozó elemek magassága 200px, kivéve a **h3**-at, mely 300px magas. A **first**-ben lévő **second** bekezdés háttere piros, betűszíne kék, magassága 200px; míg az ezen kívül levő betűje kék, háttere barna, magassága pedig 400px.

Hogyha a **#first p** kijelölés meghatározásai közül a **background-color:red;**-et kitöröljük, akkor a **first**-en belül lévő **second** bekezdés háttere is barna lesz; hogyha pedig az egész **#first p** stílust kitöröljük, és a **#first** kijelölés alá írjuk át a **min-height:200px;** meghatározást, akkor **first** alatt lévő **second** magassága is 400px-re vált, míg az ott lévő többi bekezdésé 100px-re csökken.

Hogyha pedig a stílus-oldalt a következőképpen írjuk:

```
<html>
<head>
<style type="text/css">
body p {min-height:100px;}
#first {background-color:red;min-height:200px;}
#first p {background-color:red;min-height:200px;}
#first p#second {background-color:brown;min-height:400px;}
#first h3 {min-height:300px;}
#second {color:blue;background-color:brown;min-height:400px;}
</style>
</head>
```

(illetve a **#first p#second** kijelölésből a **p**-t el is hagyhatjuk), akkor már a **first**-en belüli **second** bekezdés háttere és magassága is megegyezik az azon kívülivel.

Hogyha pedig a **first**-öt **id** helyett **class**-ként állítjuk be, eleve ugyanezt kapjuk:

```
<html>
<head>
<style type="text/css">
body p {min-height:100px;}
.first {background-color:red;min-height:200px;}
.first p {background-color:red;min-height:200px;}
.first h3 {min-height:300px;}
#second {color:blue;background-color:brown;min-height:400px;}
</style>
</head>

<body>
<div class="first">
<h3>The minimum height of this header is set to 100px.</h3>
<p>The minimum height of this paragraph is set to 200px.</p>
<p id="second">The minimum height of this paragraph is set to
200px.</p>
<p>The minimum height of this paragraph is set to 200px.</p>
<p>The minimum height of this paragraph is set to 200px.</p>
<p>The minimum height of this paragraph is set to 200px.</p>
</div>

<p id="second">The minimum height of this paragraph is set to
400px.</p>

<p>The minimum height of this paragraph is set to 100px.</p>
<p>The minimum height of this paragraph is set to 100px.</p>
<p>The minimum height of this paragraph is set to 100px.</p>
</body>
</html>
```

Példánk tanulsága tehát, hogyv az egyedi (**id**-re vonatkoztatott) kollektív formázás felülírhatja a benne foglalt elemek egyedi formázását is; ha azonban a csoportot **class**-ként jelöljük ki, efféle hatás nem fordulhat elő. A ** ** elemet tehát itt egy csoport kijelölésére használtuk, ahelyett, hogy minden egyes elembe beírtuk volna a csoportnevet (**class** attribútumot); márpedig hogyha ebben az esetben **class** helyett

id-t írunk, az egyedi formázásnak számít, mely természetesen felülírthatja a további egyedi formázásokat, akkor is, ha azok stílus-leírása a sorban előbb szerepel.

Végül, mint hetedik példánk mutatja, az elemek minimális szélességét a **min-width** jellemzővel állíthatjuk be:

```
<html>
<head>
<style type="text/css">
p {min-width:50px;}
</style>
</head>

<body>
<p>The minimum width of this paragraph is set to 50px.</p>
</body>
</html>
```

CSS méret-jellemzők

Az alábbi táblázatban összefoglaltuk a címben szereplő jellemzőket és azok lehetséges értékeit. A „CSS” oszlopba írt szám azt mutatja, hogy az adott jellemzőt a CSS mely verziója tartalmazta először:

Jellemző	Leírás	Lehetséges értékek	CSS
height	Az elem magasságát határozza meg.	auto *length* % inherit	1
max-height	Alehetséges legnagyobb elem-magasságot adja meg.	none *length* % inherit	2
max-width	Alehetséges legnagyobb elem-szélességet adja meg.	none *length* % inherit	2
min-height	Az elem minimális magasságát adja meg.	*length* % inherit	2
min-width	Az elem minimális szélességét adja meg.	*length* % inherit	2
width	Az elem szélességét adja meg.	auto *length* % inherit	1

CSS – méretezés:

Magasság: {height:*100px/100em/100%/auto/inherit*;}
Szélesség: {width:*100px/100em/100%/auto/inherit*;}
Legnagyobb magasság: {max-height:*érték/auto/inherit*;}
Legnagyobb szélesség: {max-height:*érték/auto/inherit*;}
Legkisebb magasság: {min-height:*érték/auto/inherit*;}
Legkisebb szélesség: {min-height:*érték/auto/inherit*;}

Az (inline/belső/külső) CSS mindig felülírja a statikus inline attribútumokat (pl.: height,width, align).

III. CSS – megjelenítés és láthatóság

A **display** (=megjelenítés) jellemző segítségével beállíthatjuk, hogy egy elem meg legyen-e jelenítve, és ha igen, hogyan.
A **visibility** (=láthatóság) jellemzővel beállítható, hogy egy elem látható legyen vagy rejtve maradjon.

Elemek elrejtése (display:none; vagy visibility:hidden;)
Egy HTML-elemet eltüntethetünk (azaz a böngésző által nem-mutatandóvá tehetünk), hogyha a **display** jellemzőnek **"none"** vagy a **visibility**-nek **"hidden"** értéket adunk.
E két eljárás azonban eltérő eredményre vezet.

A **visibility:hidden;** meghatározással az elem ugyan láthatatlanná válik, de a helye az oldalon továbbra is (üresen) megmarad;

```
<html>
<head>
<style type="text/css">
h1.hidden {visibility:hidden;}
</style>
</head>

<body>
<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the hidden heading still takes up space.</p>
</body>
</html>
```

A példánkban szereplő első címsor és bekezdés közti második címsor nem látszik, habár az azt közrefogó elemek képe között ki van hagyva a hely számára.

A **display:none;** meghatározással azonban, mint alább láthatjuk, nemcsak az elemet, hanem a helyét is eltüntetjük a dokumentum-képből; azaz a weboldal úgy jelenik meg, minha az elem egyáltalán nem szerepelne benne:

```
<html>
<head>
<style type="text/css">
h1.hidden {display:none;}
</style>
</head>

<body>
<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the hidden heading does not take up space.</p>
</body>
</html>
```

Mint látjuk, ez a megelőző példa átalakítása; a különbség pedig az, hogy az első **<h1>** után rögtön a bekezdés látszik.

CSS-megjelenítés – tömbszerű és sorba-zárt elemek
A tömbszerű (=block) elemek a weboldal teljes szélességében kerülnek megjelenítésre, valamint (előttük és utánuk hagyott) üres sorok választják el őket a dokumentum többi elemétől.
Tömbszerű elem többek között a **<h1>**, a **<p>** és a ** **.

A sorba-zárt (=inline) elemek által elfoglalt hely csak a tényleges tartalmuktól függ, és nincsenek üres sorokkal elkerítve a többitől.
Sorba-zárt elem többek között a **** és az **<a>**.

Az elem-megjelenítés módosítása
A CSS segítségével a sorba-zárt elemek tömbszerűvé alakíthatók (ill. a tömbeszrűak sorba-zárttá). Így a webes előírások betartásával is egyedi kinézetűvé tehetjük weboldalunkat.

A következő példa (egyébként tömbszerű) lista-elemei sorba-zártan jelennek meg:

```
<html>
<head>
<style type="text/css">
li{display:inline;}
</style>
</head>

<body>
<p>Display this link list as a horizontal menu:</p>

<ul>
<li><a href="/html/default.asp" target="_blank">HTML</a></li>
<li><a href="/css/default.asp" target="_blank">CSS</a></li>
<li><a href="/js/default.asp" target="_blank">JavaScript</a></li>
<li><a href="/xml/default.asp" target="_blank">XML</a></li>
</ul>

</body>
</html>
```

A fenti listában szereplő linkek alapértelmezésben tömbszerűen, jista-jelek mellett sorakoznának egymás alatt (hogyha kitöröljük a rá vonatkozó stílust); az előbbi beállítás mellett azonban a bekezdés alatti sorban, egymás mellett láthatók, vízszintes menüsort alkotva.

Másik példánkban pedig tömbszerű **** elemeket állítunk be (az alapértelmezet sorba-zártak helyett). Mint a HTML1 dokumentumban láttuk, a **** tag nem alapértelmezésben nem okoz semmilyen látható változást a dokumentumban; hanem HTML-elemek összefogására, strukturálására szolgál. A **** elemmel összefogott elemeket együttesen formázhatjuk, vagy tartalmukat JavaScript-tel kezelhetjük, stb..

```
<html>
<head>
<style type="text/css">
span {display:block;}
</style>
</head>
<body>

<h2>Nirvana</h2>
<span>Record: MTV Unplugged in New York</span>
<span>Year: 1993</span>
<h2>Radiohead</h2><html>
<head>
<style type="text/css">
span {display:block;}
</style>
</head>
<body>

<h2>Nirvana</h2>
<span>Record: MTV Unplugged in New York</span>
<span>Year: 1993</span>
```

```
<h2>Radiohead</h2>
<span>Record: OK Computer</span>
<span>Year: 1997</span>
</body>
</html>
<span>Record: OK Computer</span>
<span>Year: 1997</span>
</body>
</html>
```

{display:block;} meghatározás nékül a címsorok alatt, egyetlen sorban szerepel az összes, utánuk jövő szó. A meghatározással együtt az egyes címsorok alát tartozó két-két **** külön sorban jelenik meg. A sortörésen kívül nincs más különbség a kétféle megjelenés között.

A megjelenítési és láthatósági beállítások csak az elem kinézetét befolyásolják, a típusát nem változtatják meg. Pl. egy **block** megjelenésű **inline** elembe sem szabad egy (valódi) **block** elemet beágyazni.

További példák

Első példánk bekezdései sorba-zártak:

```
<html>
<head>
<style type="text/css">
p {display:inline;}
</style>
</head>

<body>
<p>A display property with a value of "inline" results in</p>
<p>no distance between two elements.</p>
</body>
</html>
```

Mint a példa-szövegben is látszik, a két bekezdést (mely voltaképpen egy mondatot alkot) nem különül el két sorba, csupán egy szóköz választja el őket egymástól. A szóköz nem szereel a HTML-kódban, azt a böngésző automatikusan adja hozzá a dokumentumhoz, a sorköz helyett.

A következő példa **** elemei tömbszerűen jelennek meg:

```
<html>
<head>
<style type="text/css">
span {display:block;}
</style>
</head>
<body>

<span>A display property with a value of "block" results
in</span><span>a line break between the two elements.</span>
</body>
</html>
```

A két **** elembe írt szöveget a böngésző külön sorokba írja, azaz sortöréssel választja el az egyébként inline elemeket. Hogyha a belső stílus-oldalt (vagy a **span** kijelölés stílusát) kitöröljük, a mondat egy sort alkot, és a tagek határán nincs se sortörés, se szóköz („A display property with a value of "block" results **ina** line break between the two elements.”). Azaz míg az stílussal együtt a porgram hozzáadott egy automatikus elválasztó-elemet a dokumentumhoz, egy sortörés formájában, az utóbbi esetben ez hiányzik, így a szöveg eredeti állapotában kerül megjelenítésre, melyben nincs ilyesmi.

Harmadik példánkban egy táblázat-elemet (sort) tüntetünk el:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
tr.collapse {visibility:collapse;}
</style>
</head>
<body>

<table border="1">
<tr>
<td>Peter</td>
<td>Griffin</td>
</tr>

<tr class="collapse">
<td>Lois</td>
<td>Griffin</td>
</tr>
</table>

<p><b>Note:</b> Internet Explorer supports visibility:collapse only if
a !DOCTYPE is specified.</p>
</body>
</html>
```

A **{visibility:collapse;}** meghatározás eredménye gyakorlatilag megegyezik a **{display:none;}**-nal: hogyha beírjuk a stílus-oldalba, a táblázat második sora egyszrűen eltűnik (a helye sem marad).

CSS – megjelenítés és láthatóság:

Megjelenítés/eltüntetés(display)

Nem jelenik meg (hely sincs hagyva): {display:none;}

Tömszerű elem (teljes frame-szélesség és sorközök): {display:block;}
Alapértelmezésben tömbszerű elemek: **h1>**, **<p>**, ** **, **<li>**.

Sorbazárt elem (saját szélesség és szóközök): {display:inline;}
Alapértelmezésben sorbazárt elemek: ****, **<a>**.
[Pl.: span {display:block;} beállítás mellett a dokumentum összes **** eleme tömbszerűen jelenik meg.]

Láthatóság (visibility)

Nem látható (de helye van): {visibility:hidden;}
Nem látható (hely sincs hagyva): {visibility:collapse;}

A megjelenítési és láthatósági beállítások csak az elem kinézetét befolyásolják, a típusát nem változtatják meg. Pl. egy block megjelenésű inline elembe sem szabad egy (valódi) block elemet beágyazni!

A tömbszerű elemeket mindig sorközök választják el; a sorbazártak között azonban csak akkor van szóköz, hogyha azt a kód szóköz vagy enter formájában eleve tartalmazza!
[Tehát pl. a *PéldapéldaPélda* kód megjelenése: *PéldapéldaPélda*;
szemben a *Példa példa Példa* kóddal, mely így jelenik meg: *Példa példa Példa*.]

IV. CSS – pozícionálás

Pozícionálás

A CSS pozícionáló jellemzői az elemek elhelyezkedésének beállítására valók. Ugyancsak lehetőséget biztosítanak ez elemek egymás alatti/feletti megjelenítésére (dokumentum-„rétegek") és annak meghatározására is, mi történjen, ha egyes elemek túl nagy méretűek.

Az elemek elhelyezkedését a **top**, **bottom**, **left** és **right** jellemzőkkel szabályozhatjuk. Ezek azonban csak a **position** jellemző beállítása után használhatók, az egyes pozícionálási eljárásoktól függően eltérő módon.

Négyféle pozícionálási eljárást alkalmazhatunk:

Statikus pozícionálás

Ez az alapértelmezett pozícionálási eljárás. Az elemek statikus pozíciója mindenkor a weboldal hagyományos szerkezetéből adódik.
A statikus elhelyezésű elemkre a fenti (**top**, **bottom**, **left** és **right**) jellemzők nincsenek hatással.

Fix pozícionálás

Az elemek böngésző-ablakhoz képest való elhelyezési módját nevezzük „fixnek". Az így elhelyezett elemek akkor sem mozdulnak el, hogyha az oldalt lejjebb gördítjük, pl.:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
p.pos_fixed
{
position:fixed;
top:30px;
right:5px;
}
</style>
</head>

<body>
<p class="pos_fixed">Some more text</p>
<p><b>Note:</b> Internet Explorer 7 (and higher) supports the fixed
value if a
!DOCTYPE is specified.</p>

<p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p><p>Some text</p><p>Some text</p>
</body>
</html>
```

Itt a **pos_fixed** bekezdés, melynek teteje az oldaléhoz képest 30 pixellel, jobb széle az oldaléhoz képest pedig 5 pixellel van lejjebb ill. balrább, nem mozdul el az oldal (pontosabban frame) legördítésére. Mint látjuk, itt is célszerű **DOCTYPE**-ot megadni.

A **„Note"** bekezdés, habár a HTML-kódban hátrébb szerepel, a előrébb jelenik meg a weboldalon; tehát a fixen pozícionált tartalom ugyanott jelenik meg, akárhol is szerepel a HTML-kódban. A dokumentum megjelenése tehát nem tükrözi a HTML-kód szerkezetét, sorrendiségét. A dokumentum további elemei a fix elem(ek)től függetlenül kerülnek megjelenítésre; nincsenek szabadon hagyandó helyek (lásd alább!) nem is létezne. A fix elrendezésű elemek tehát át- illetve lefedhetnek más elemeket.

Relatív pozícionálás
A relatív pozíciót az elem normális weboldal-beli elhelyezkedésére vonatkoztatjuk, pl.:

```
<html>
<head>
<style type="text/css">
h2.pos_left {position:relative;left:-20px;}
h2.pos_right {position:relative;left:20px;}
</style>
</head>

<body>
<h2>This is a heading with no position</h2>
<h2 class="pos_left">This heading is moved left according to its normal
position</h2>
<h2 class="pos_right">This heading is moved right according to its
normal position</h2>
<p>Relative positioning moves an element RELATIVE to its original
position.</p>
<p>The style "left:-20px" subtracts 20 pixels from the element's
original left position.</p>
<p>The style "left:20px" adds 20 pixels to the element's original left
position.</p>
</body>
</html>
```

A példában szereplő első címsorhoz képest a második 20 pixellel balrább, a harmadik pedig ugyanennyivel jobbrább látható. A **left** irány (kijelölés) csak az érték +/- értékének értelmezéséhez szükséges; a vonatkoztatási pont az eredeti helyzet (azaz végeredményben egyik címsor sem 20px-re kezdődik a bal széltől, hanem a bal széltől nézve pozitív ill. negatív értelemben változik az elemek vízszintes elhelyezkedése).

A relatív pozícióval bíró elemek tehát az oldallal együtt mozognak és átfedésben lehetnek más elemekkel; a nekik járó hely azonban továbbra is megmarad a dokumentum-képben (esetleg üresen!), amint azt az alábbi példa is mutatja:

```
<html>
<head>
<style type="text/css">
h2.pos_top {position:relative;top:-50px;}
</style>
</head>

<body>
<h2>This is a heading with no position</h2>
<h2 class="pos_top">This heading is moved upwards according to its
normal position</h2>
<p><b>Note:</b> Even if the content of the relatively positioned
element is moved, the reserved space for the element is still preserved
in the normal flow.</p>
</body>
</html>
```

Itt az „alsó” kétsoros cím felső sora gyakorlatilag átfedi a felső címet, de az eredeti helye továbbra is szabadon áll a **<p>** fölött. A **top** ismét csak iránykijelölést szolgál, hiszen az alsó címsor nem emelkedik a frame teteje fölé.

A relatív elhelyezésű elemeket gyakran az abszolút elhelyezésűek „tároló-dobozaként” használjuk.

Abszolút pozícionálás
Az abszolút pozíciójú elemeket az őket tartalmazó, nem statikus elhelyezésű elemekhez képest helyeztük el. Hogyha efféle elem nincs, akkor a legkülső, **<html>** szolgál e célra, pl.:

```
<html>
<head>
<style type="text/css">
h2
{
position:absolute;
left:100px;
top:150px;
}
</style>
</head>

<body>
<h2>This is a heading with an absolute position</h2>
<p>With absolute positioning, an element can be placed anywhere on a
page. The heading below is placed 100px from the left of the page and
150px from the top of the page.</p>
</body>
</html>
```

Itt a kódban hátrébb álló bekezdés látható az oldal tetején; a címsor pedig a középen. Mint látjuk, az abszolút eljárással egy elem bárhol elhelyezhető a weboldalon (a kódbeli sorrendtől függetlenül).

Az abszolút pozícióval rendelkező elemeket kivonjuk a HTML-kód megszabta szerkezetből, a dokumentum és a többi tehát ettől függetlenül kerül megjelenítésre; nincsenek üresen hagyandó helyek. A statikus pozícionálástól eltérően, és a fix-szel, relatívval megegyezően, az abszolút elhelyezésű elemek átfedhetnek másokat.

Átfedések elemek között
A dokumentum-szerkezettől eltérően (azaz nem statikusan) elhelyezettek átfedhetnek további elemeket. A **z-index** jellemző az elemek elhelyezési szintjét (=stack order) írja le (ti. hogy mely elemeket kell a többi előtt vagy mögött megjelenítni).

Az elemek pozitív vagy negatív elhelyezési szintjük lehet:

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
left:0px;
top:0px;
z-index:-1;
}
</style>
</head>

<body>
<h1>This is a heading</h1>
<img src="w3css.gif" width="100" height="140" />
<p>Because the image has a z-index of -1, it will be placed behind the
text.</p>
</body>
```

```
</html>
```
Itt a kép felső ill. bal élei pontosan a weboldal bal felső sarkához illeszkednek, s a szöveg „rá van írva”, azaz lefedi a képet. Hogyha a kép elhelyezési szintjét -1 -ről +1 -re változtatjuk, akkor az takarja majd el a szöveget (a helye nem változik).

Mint látjuk, a magasabb elhelyezési szintű elemek mindig lefedik az alacsonyabbakat. Hogyha nem adunk meg elhelyezési szinteket, akkor a HTML-kódban hátrébb szereplő elemek helyezkednek el „magasabban”, azok fedik le a kódban előrább álló elemeket. (Ez a HTML-megjelenítés sorrendiségének köszönhető.)

További példák

Első példánk egy elem (kép) alakjának beállításáról szól. A képet a böngésző a megadott alakzatba „szorítva” jeleníti meg:

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
left:0px;
top:0px;
clip:rect(0px,100px,140px,0px);
}
</style>
</head>

<body>
<img src="w3css.gif" width="100" height="140" />
</body>
</html>
```

E példában először is beállítjuk a képet:

```
<img src="w3css.gif" width="100" height="140" />
```

majd a **clip** jellemzővel a széleiből elkezdünk „lecsípkedni”. A fenti beállítás mellett a frame bal felső sarkába illesztett kép teljes egészében látszik. Hogyha a **clip** első értékét 0-ról 25-re változtatjuk, akkor a kép tetejéből egy 25px-nyi, vízszintes sáv eltűnik. Hogyha most a harmadik értéket 140-ről 115-re csökkentjük, akkor a kép alsó részéből is eltűnik ugyanennyi. Ha a második értéket 100-ról 75-re csökkentjük, akkor a kép jobb széléből tűnik el egy 25px széles, függőleges sáv. Ha pedig a negyedik értéket 0-ról 25-re növeljük, akkor a kép bal széléből is eltűnik ugyanennyi; tehát végeredményben a kép minden széléből levágtunk 25 pixelt, tehát a képnek csak a közepe látszik, ugyanott, ahol erdedetileg is volt.

Hogyha most a meghatározást átírjuk a következőre:

```
clip:rect(0px,300px,300px,0px)
```

akkor a kép ugyanolyan, mint eredetileg volt. Hogyha a kódba a kép után bekezdéseket írunk, akkor látjuk, hogy az abszolút elhelyezésű kép ezek fölött jelenik meg; illetve hogy az igen nagy (pl. 1000px-es) lecsípési értékek beállítása mellett sem takar el többet a többi elemből, mint a maximális kiterjedése.

Második példánkban az **overflow** jellemzőt alkalmazzuk egy olyan elemek megjelenítésére, melyek túl nagyok ahhoz, hogy beleférjenek a számukra megadott térrészbe:

```
<html>
<head>
<style type="text/css">
div.scroll
{
background-color:#00FFFF;
width:100px;
height:100px;
```

```
overflow:scroll;
}

div.hidden
{
background-color:#00FF00;
width:100px;
height:100px;
overflow:hidden;
}
</style>
</head>

<body>
<p>The overflow property specifies what to do if the content of an
element exceeds the size of the element's box.</p>

<p>overflow:scroll</p>
<div class="scroll">You can use the overflow property when you want to
have better control of the layout. The default value is visible.</div>

<p>overflow:hidden</p>
<div class="hidden">You can use the overflow property when you want to
have better control of the layout. The default value is visible.</div>
</body>
</html>
```

Itt a **scroll** és **hidden** osztályú bekezdések egy-egy 100*100 pixeles (statikus) pozícióba kerülnek. A **scroll** osztályú (100*100px) területen belül egy függőleges csúszka található, mellyel a kereten túllógó tartalmat is megnézhetjük. A **hidden** osztályú ** **-be írt szöveg túllógó része azonban nem látszik.
Hogyha azonban ezt az előző példára próbáljuk alkalmazni:

```
<html>
<head>
<style type="text/css">
img.original
{
position:absolute;
left:0px;
top:0px;
clip:rect(0px,100px,140px,0px);
}

img.scroll
{
position:absolute;
left:0px;
top:200px;
clip:rect(30px,80px,80px,30px);
overflow:scroll
}
</style>
</head>

<body>
<img class="original" src="w3css.gif" width="100" height="140" />
```

```
<img class="scroll" src="w3css.gif" width="100" height="140" />
</body>
</html>
```
nem járunk eredménnyel; akkor sem, hogyha az abszolút helyett statikus pozícionálást alkalmazunk. A képek továbbra is csúszkák nélkül láthatók; statikus pozícionálás esetén pedig még a csonkolás sem működik.

Harmadik példánkban a böngészőt az **overflow** automatikus alkalmazására utasítjuk, a helyükön túlnyúló elemekre vonatkozólag:
```
<html>
<head>
<style type="text/css">
div
{
background-color:#00FFFF;
width:150px;
height:150px;
overflow:auto;
}
</style>
</head>

<body>
<p>The overflow property decides what to do if the content inside an
element exceeds the given width and height properties.</p>

<div>
You can use the overflow property when you want to have better control
of the layout. Try to change the overflow property to: visible, hidden,
scroll, or inherit and see what happens. The default value is visible.
</div>
</body>
</html>
```
Itt az **overflow** jellemző **auto** értéke mellett a kék hátterű keret jobb szélén függőleges csúszka található. A **visible** érték mellett a szöveg a kék kereten túlnyúlva, az oldal fehér részén folytatódik; ám bár magassága meghaladja a megadott függőleges helyet, szélessége továbbra is a megadott 150px-es értékhez igazodik (a kék mező alatt). A **hidden** beállításnál a tartalom mezőn túli része nem látszik; az utolsó sor közepétől le van csonkolva. A **scroll** beállítás eredménye gyakorlatilag az **auto**-val egyezik meg; csakhogy itt ismét feltűnik egy alsó, kihasználatlan csúszka is. Végül, az **inherit** beállítás mellett, a **visible**-lel megegyező eredményre jutunk.

Negyedik példánkban néhány szövegtestbeli **style** attribútum segítségével az egyes elemekre mutató egér számára különféle kurzor-jeleket írunk elő:
```
<html>
<body>
<p>Mouse over the words to change the cursor.</p>
<span style="cursor:auto">auto</span><br />
<span style="cursor:crosshair">crosshair</span><br />
<span style="cursor:default">default</span><br />
<span style="cursor:e-resize">e-resize</span><br />
<span style="cursor:help">help</span><br />
<span style="cursor:move">move</span><br />
<span style="cursor:n-resize">n-resize</span><br />
<span style="cursor:ne-resize">ne-resize</span><br />
<span style="cursor:nw-resize">nw-resize</span><br />
```

```
<span style="cursor:pointer">pointer</span><br />
<span style="cursor:progress">progress</span><br />
<span style="cursor:s-resize">s-resize</span><br />
<span style="cursor:se-resize">se-resize</span><br />
<span style="cursor:sw-resize">sw-resize</span><br />
<span style="cursor:text">text</span><br />
<span style="cursor:w-resize">w-resize</span><br />
<span style="cursor:wait">wait</span><br />
</body>
</html>
```

A **cursor** jellemző **auto** beállítása mellett a szöveges tartalomnak megfelelő szövegkijelölő kurzor jelenik meg az „auto” szöveg fölött. A **crosshair** fölött ugyanígy kereszt; a **default** felett a hagyományos nyílszerű egérmutató, az **e-resize** (=east-resize) fölött a jobboldali ablak-átméretező jel, a **help** fölött egy kérdőjel jelenik meg, a **move** fölött mozdító-kéz, az **n-resize** (=north-resize) fölött felülről-átméretező, a **ne-resize** (=northeast-resize) fölött jobb-felső – átméretező, a **nw-resize** (=northwest-resize) fölött bal felső átméretező, a **pointer** fölött mutató-kéz; a **progress** fölött homokór
ás egérnyíl, az **s-resize** (=south-resize) fölött alsó átméretező, a **se-resize** (=southeast-resize) fölött jobb-alsó – átméretező, a **sw-resize** (=southwest-resize) fölött bal-alsó – átméretező, a **text** fölött szövegkurzor, a **w-resize** (=west-resize) bal átméretező, a **wait** fölött pedig homokóra jelenik meg.

CSS pozícionáló-jellemzők

Az alábbi táblázatban összefoglaltuk a címben szereplő jellemzőket és azok lehetséges értékeit. A „CSS” oszlopba írt szám azt mutatja, hogy az adott jellemzőt a CSS mely verziója tartalmazta először:

Jellemző	Leírás	Lehetséges értékek	CSS
bottom	Sets the bottom margin edge for a positioned box	auto *length* *%* inherit	2
clip	Clips an absolutely positioned element	*shape* auto inherit	2
cursor	Specifies the type of cursor to be displayed	*url* auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help	2
left	Sets the left margin edge for a positioned box	auto *length* *%* inherit	2
overflow	Specifies what happens if content overflows an element's box	auto hidden scroll visible inherit	2
position	Specifies the type of positioning for an element	absolute fixed relative	2

		static inherit	
right	Sets the right margin edge for a positioned box	auto *length* % inherit	2
top	Sets the top margin edge for a positioned box	auto *length* % inherit	2
z-index	Sets the stack order of an element	*number* auto inherit	2

CSS–pozícionálás:

Pozícionálási típus: {position:*static/fixed/relative/absolute/inherit*;}

Statikus pozícionálás: {position:static;}

A tagek sorrendjéből és a böngésző alapértelmezéséből áll össze (nem rendelhető hozzá további stílus). Az elem körül a böngésző autiomatikusan helyet biztosít, így az nem fedhet át másokat.

Fix pozícionálás: {position:fixed;}

A böngésző-ablakhoz (frame-hez) képest állandó helyzetet adhatunk meg. A fix elrendezésű elem átfedése kerülhet más elemekkel (nincs hely kihagyva számára), és nem mozog együtt a legördülő oldallal.

Relatív pozícionálás: {position:relative;}

Az elemet eredeti, statikus helyzetéhez képest pozícionálja. Az eredeti helye megmarad, de új nem alakul ki, így átfedésbe kerülhet más elemekkel. A legördülő oldallal együtt mozog. A relatív elhelyezésű elemeket gyakran az abszolút elhelyezésűek „tároló-dobozaként” használjuk.

Abszolút pozícionálás: {position:absolute;}

Az abszolút pozíciójú elemeket az őket tartalmazó, nem statikus elhelyezésű elemekhez képest helyeztük el. Hogyha efféle elem nincs, akkor a legkülső, **<html>** szolgál e célra. Az elemnek nincs sem eredeti, sem új hely kihagyva, így átfedhet másokat és az oldallal együtt gördül le.

Pozícionálási érték: {*top/right/bottom/left*:*100px/100em/100%/auto/inherit*;}

Felső: {top:*100px/100em/100%/auto/inherit*;}
Jobb: {right:*100px/100em/100%/auto/inherit*;}
Alsó: {bottom:*100px/100em/100%/auto/inherit*;}
Bal: {left:*100px/100em/100%/auto/inherit*;}

Az érték megadja az adott oldali elemmargó vonalának távolságát a frame azonos szélétől.

Elhelyezési szint (stack-order): {z-index:-*1/auto/inherit*;}

A nem statikusan pozícionált (egymást átfedhető) elemek szintjét mutatja meg. A magasabb értékűek lefedik a kisebbeket. Alapértelmezésben a kódban hátrébb álló tagek a magasabb értékűek.

Bevágás (négyzet alakú mezőbe): {clip:*rect/auto/inherit* (*0px,100px,100px, 0px*;}

Itt egy 100x100px-es képet vágtunk be egy négyzetes keretbe. Nagyobb ill. kisebb értékek mellett a kép felső, jobb, alsó és bal oldalából egy-egy rész eltűnik. Csak fix ill. abszolút pozícionálás mellett használható. Az *auto* beállítás mellett a kép teljes alakban jelenik meg.

Túlméretes elem kezelése: {overflow:*auto/scroll/visible/hidden/inherit*;}

Automatikus (alsó, oldalsó vagy mindkettő) csúszka: {overflow:auto;}
Két csúszka: {overflow:scroll;}
Kereten túlnyúló tartalom: {overflow:visible;}
Kereten csonkolt tartalom: {overflow:hidden;}
Alapbeállítás szerint: {overflow:inherit;}

Kurzor-típusok (cursor):

Automatikus (az adott tartalomhoz illeszkedő): {cursor:auto;}
Kereszt: {cursor:crosshair;}
Alapértelmezett (hagyományos, nyílszerű): {cursor:default;}
Szöveg (szövegkurzor): {cursor:text;}
Homokóra: {cursor:wait;}
Homokórás egérnyíl: {cursor:progress;}
Súgó/segítség (kérdőjel): {cursor:help;}
Mutató-kéz (pl. linkek fölött): {cursor:pointer;}
Mozdító-kéz: {cursor:move;}

Felső ablak-átméretező jel (north-resize): {cursor:n-resize;}
Jobb-felső ablak-átméretező jel (northeast-resize): {cursor:ne-resize;}
Jobboldali ablak-átméretező jel (east-resize): {cursor:e-resize;}
Jobb-alsó ablak-átméretező jel (southeast-resize): {cursor:se-resize;}
Alsó ablak-átméretező jel (south-resize): {cursor:s-resize;}
Bal-alsó ablak-átméretező jel (southwest-resize): {cursor:sw-resize;}
Baloldali ablak-átméretező jel (west-resize): {cursor:w-resize;}
Bal-felső ablak-átméretező jel (northwest-resize): {cursor:nw-resize;}

Hogyha az igazítandó elemet tartalmazó elem szélességét is megadjuk, és a **DOCTYPE** hiányzik, az IE egy 17 pixeles margót ad a jobboldalhoz; a csúszkának szánt hely gyanánt.
Ezért a position jellemző használatakor mindig állítsuk be a DOCTYPE-ot!

V. CSS objektum-igazítás

Mi a CSS objektum-igazíás?

A CSS-igazítással (=floating) az elemeket jobbra ill. balra igazíthatjuk, miközben a többi elem is átrendeződik körülöttük. Leggyakrabban (pl. szövegben álló) képeknél találkozunk vele, de weboldal-elrendezések beállításánál is használható.

Önálló elemek igazítása

Az elemek csak vízszintesen, azaz jobbra-balra igazíthatók, függőlegesen nem. Az elem igazításakor az elérhető (bal vagy jobb) legszélső helyzetbe jut. Ez leggyakrabban azt jelenti, hogy az őt tartalmazó elem jobb- vagy bal szélére kerül.

Az elcsúsztatott elem után következők követik annak mozgását, míg az előtte állók helyzete nem változik. Hogyha pl. egy képet jobbra igazítunk, akkor az utána lévő szöveg balra kerül:

```
<html>
<head>
<style type="text/css">
img {float:right;}
</style>
</head>

<body>
<p>In the paragraph below, we have added an image with style
<b>float:right</b>. The result is that the image will float to the
right in the paragraph.</p>

<p>
<img src="logocss.gif" width="95" height="84" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
</body>
</html>
```

A második bekezdésben lévő kép a weboldal jobb szélére kerül, a szöveg pedig (balról és alulról) körbefutja azt. Hogyha a **float** jellemző értékét **left**-nek választjuk, a kép a másik szélre kerül, és a szöveg átrendeződik.

Egymást követő elemek igazítása

Hogyha számos floating elem követi egymást, azok vizszintesen egymás mellé igazodnak, majd az új sorba kerülők újra egymás mellé, stb..

A következő példában ilyen önigazodó elemekből álló képtárat készítünk:

```
<html>
<head>
<style type="text/css">
.thumbnail
{
float:left;
width:110px;
```

```
height:90px;
margin:5px;
}
</style>
</head>

<body>
<h3>Image Gallery</h3>
<p>Try resizing the window to see what happens when the images does not
have enough room.</p>
<img class="thumbnail" src="klematis_small.jpg" width="107"
height="90">
<img class="thumbnail" src="klematis2_small.jpg" width="107"
height="80">
<img class="thumbnail" src="klematis3_small.jpg" width="116"
height="90">
<img class="thumbnail" src="klematis4_small.jpg" width="120"
height="90">
<img class="thumbnail" src="klematis_small.jpg" width="107"
height="90">
<img class="thumbnail" src="klematis2_small.jpg" width="107"
height="80">
<img class="thumbnail" src="klematis3_small.jpg" width="116"
height="90">
<img class="thumbnail" src="klematis4_small.jpg" width="120"
height="90">
</body>
</html>
```

Az ablak átméretezésekor a képek átrendeződnek; a bekezdések szóközi tördeléséhez hasonlóan. Sorrendjük megmarad.
A **.thumbnail** kijelölésnél felülírjuk a képek szélességét és magasságát (pl. 107 helyett 110px-nek vagy 80 helyett 90px-nek választva azt), valamint az egyes képekhez tartozó margót 5 px-ben határozzuk meg. Ennek megfelelően a képek széle közti távolság 2*5px = 10px.

Az objektum-igazítás kikapcsolása (clear)
Az igazított elem után következők „vándorlását” meggátolhatjuk a **clear** jellemző használatával.
A **clear** jellemzővel megadhatjuk, hogy az igazított elem mely oldalaihoz nem igazodhat a következő tartalom.
Hogyha az előbbi példában szereplő képtárhoz egy címsort adunk, beállíthatjuk, hogy az mindig azonos kép után következzen:

```
<html>
<head>
<style type="text/css">
.thumbnail
{
float:left;
width:110px;
height:90px;
margin:5px;
}
.text_line
{
clear:both;
margin-bottom:20px;
}
```

```
</style>
</head>

<body>
<h3>Image Gallery</h3>
<p>Try resizing the window to see what happens when the images does not
have enough room.</p>
<img class="thumbnail" src="klematis_small.jpg" width="107"
height="90">
<img class="thumbnail" src="klematis2_small.jpg" width="107"
height="80">
<img class="thumbnail" src="klematis3_small.jpg" width="116"
height="90">
<img class="thumbnail" src="klematis4_small.jpg" width="120"
height="90">
<h3 class="text_line">Second row</h3>
<img class="thumbnail" src="klematis_small.jpg" width="107"
height="90">
<img class="thumbnail" src="klematis2_small.jpg" width="107"
height="80">
<img class="thumbnail" src="klematis3_small.jpg" width="116"
height="90">
<img class="thumbnail" src="klematis4_small.jpg" width="120"
height="90">
</body>
</html>
```

Ekkor a négy-négy képből álló két sor közt egy szabadon álló címsor található (alatta 20px margóval), mely a böngésző átméretezésekor sem változtatja státuszát. Hogyha a **clear** jellemző értékét **both**-ról **left**-re változtatjuk, a felirat az előtte lévő kép után továbbra is a bal szélre igazodik; míg **right** beállítás mellett a megelőző kép melletti jobb szélre. **none** beállítás mellett a felirat gyakorlatilag bárhol feltűnhet a képek között. **Inherit**-nél, illetve a **clear** jellemző kitörlése esetén, ugyanez a helyzet.

További példák

Első példánkban egy bekezdésben jobbra igazodó, szegéllyel és margóval rendelkező képet találunk:

```
<html>
<head>
<style type="text/css">
img
{
float:right;
border:1px dotted black;
margin:0px 0px 15px 20px;
}
</style>
</head>

<body>
<p>
In the paragraph below, the image will float to the right. A dotted
black border is added to the image.
We have also added margins to the image to push the text away from the
image:
0 px margin on the top and right side, 15 px margin on the bottom, and
20 px margin on the left side of the image.
</p>
```

```
<p>
<img src="logocss.gif" width="95" height="84" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
</body>
</html>
```

Mint látjuk, a kép a bekezdésben 1px-es, pontozott kerettel és rendre 0, 0, 15 és 20 pixeles felső, jobb, alsó és bal margóval jelenik meg. Az első bekezdés kódjában található enterek természetesen a megjelenő szövegben nem eredményeznek sortörést, csak szóközt.

Második példánkban egy aláírattal ellátott képet látunk, egy bekezdésben jobbra igazítva:

```
<html>
<head>
<style type="text/css">
div
{
float:right;
width:120px;
margin:0 0 15px 20px;
padding:15px;
border:1px solid black;
text-align:center;
}
</style>
</head>

<body>
<div>
<img src="logocss.gif" width="95" height="84" /><br />
CSS is fun!
</div>
<p>
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text
</p>
<p>
```

```
In the paragraph above, the div element is 120 pixels wide and it
contains the image.
The div element will float to the right.
Margins are added to the div to push the text away from the div.
Borders and padding are added to the div to frame in the picture and
the caption.
</p>
</body>
</html>
```

A kép, habár nem része az őt körülfutó bekezdésnek, a frame jobb oldalára kerül; alatta 15, tőle balra 20 pixeles margó van; kerete egyszeres, fekete. A ** **-be írt szöveg a kép előtt ill. után jelenik meg, attól függően, hogy az **<img>** tag előtt vagy mögött szerepel. A **text-align** jellemző a szövegre és a képre egyaránt érvényes (a kereten belül).

Harmadik példánkban egy bekezdés első betűjét baloldalra igazítjuk, és további stílussal látjuk el:

```
<html>
<head>
<style type="text/css">
span
{
float:left;
width:0.7em;
font-size:400%;
font-family:algerian,courier;
line-height:80%;
}
</style>
</head>

<body>
<p>
<span>T</span>his is some text.
This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>

<p>
In the paragraph above, the first letter of the text is embedded in a
span element.
The span element has a width that is 0.7 times the size of the current
font.
The font-size of the span element is 400% (quite large) and the line-
height is 80%.
The font of the letter in the span will be in "Algerian".
</p>
</body>
</html>
```

Ekkor a bekezdés első betűje, melyet a **span** taggel jelöltünk ki a formázáshoz (hogy ne szakítsuk ki a sorból pl. egy **<p>**, azaz tömbszerű elemmel) a szövegétől eltérő betűtípussal és négyszer nagyobb méretben jelenik meg; az utóbbi beűméretnek megfelelő sormagasság 80%-ának illetve 0.7-szeresének

megfelelő magasságú és szélességű helyen. Hogyha a **width**-et **1em**-nek, a **line-height**-et pedig **100%**-nak vesszük, akkor a T betű egy saját sorméretének megfelelő szélességű és magasságú neégyzet bal-felső sarkában foglal helyet.

Negyedik példánkban egy vízszintes menüsort készítünk, néhány linkből:

```
<html>
<head>
<style type="text/css">
ul
{
float:left;
width:100%;
padding:0;
margin:0;
list-style-type:none;
}
a
{
float:left;
width:6em;
text-decoration:none;
color:white;
background-color:purple;
padding:0.2em 0.6em;
border-right:1px solid white;
}
a:hover {background-color:#ff3300;}
li {display:inline;}
</style>
</head>

<body>
<ul>
<li><a href="#">Link one</a></li>
<li><a href="#">Link two</a></li>
<li><a href="#">Link three</a></li>
<li><a href="#">Link four</a></li>
</ul>

<p>
In the example above, we let the ul element and the a element float to
the left.
The li elements will be displayed as inline elements (no line break
before or after the element). This forces the list to be on one line.
The ul element has a width of 100% and each hyperlink in the list has a
width of 6em (6 times the size of the current font).
We add some colors and borders to make it more fancy.
</p>

</body>
</html>
```

A weboldalon lévő rendezetlen lista egészét balra igazítjuk. Szélességét az oldalla egyezőnek (100%) választjuk; térközt és margót nem hagyunk körülötte, tehát rögtön az oldal tetején jelenik meg, és a rákövetkező bekezdés margó nélkül áll mögötte.
Az **<a>** tageknek fehér betűszínt és lila háttérszínt állítunk be; mely a megnövelt térköz és a köztük hagyott

kis margó miatt dobozszerű benyomást kelt. A „dobozok” mindegyikének szélessége a betűméret hatszorosa. Az aktív link narancssárgára vált.
A **li** kijelölésre vonatkozó meghatározás révén a balra igazított **<ul>** ill. **<a>** elemek egy sorban jelenhetnek meg. Hogyha ezt a meghatározást kitöröljük, a lista-elemek egymás alatt jelennek meg (Internet Explorerben). Tehát a **float** jellemzőt az **<ul>**, **<li>** és **<a>** elemeknél egyaránt be kell állítani, a végeredmény kontrollálhatósága érdekében.

Az oldalt átalakíthatjuk oldalsó menüsorossá:

```
<html>
<head>
<style type=”text/css”>
body {margin:5% 10%;}
.ul_one
{
float:left;
padding:0
margin:0px 10px 10px 10px;
list-style-type:none;
}
.ul_one a
{
text-decoration:none;
color:white;
}
a:hover {background-color:#ff3300;}
li
{
text-align:center;
display:block;
width:9em;
background-color:purple;
margin:0.7em 0.9em;
}
p {padding-top:0.7em;}
</style>
</head>

<body>
<span class="ul_one"><ul>
<li><a href="#">Link one</a></li>
<li><a href="#">Link two</a></li>
<li><a href="#">Link three</a></li>
<li><a href="#">Link four</a></li>
</ul></span>

<p> In the example above, we let the ul element and the a element float
to the left. The li elements will be displayed as inline elements (no
line break before or after the element). This forces the list to be on
one line. The ul element has a width of 100% and each hyperlink in the
list has a width of 6em (6 times the size of the current font). We add
some colors and borders to make it more fancy.</p>
</body>
</html>
```

Ötödik példánkban a megismert pozícionálási módokat táblázat-struktúrától mentes weboldal-formázáshoz

használjuk fel. A készített wenlapnak élőfeje, élőlába, mellék- és fő-tartalmi mezője is lesz:

```
<html>
<head>
<style type="text/css">
div.container
{
width:100%;
margin:0px;
border:1px solid gray;
line-height:150%;
}
div.header,div.footer
{
padding:0.5em;
color:white;
background-color:gray;
clear:left;
}
h1.header
{
padding:0;
margin:0;
}
div.left
{
float:left;
width:160px;
margin:0;
padding:1em;
}
div.content
{
margin-left:190px;
border-left:1px solid gray;
padding:1em;
}
</style>
</head>

<body>
<div class="container">

<div class="header"><h1 class="header">W3Schools.com</h1></div>

<div class="left"><p>"Never increase, beyond what is necessary, the
number of entities required to explain anything." William of Ockham
(1285-1349)</p></div>

<div class="content">
<h2>Free Web Building Tutorials</h2>
<p>At W3Schools you will find all the Web-building tutorials you need,
from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.</p>
<p>W3Schools - The Largest Web Developers Site On The Net!</p></div>

<div class="footer">Copyright 1999-2005 by Refsnes Data.</div>
```

```
</div>
</body>
</html>
```
A belső CSS értelmezése:
A **div.container** kijelölés az egész weboldalra vonatkozik. Ez a ** ** elem az oldalszélesség 100%-át kitölti, margója nincs, külső kerete vékony, szürke; és az összes sormagasság az alapértelmezett 150%-a. Hogyha a **container** osztályú ** **-et kitöröljük, és **body**-t állítunk be első kijelölésként, az eredmény az előbbihez hasonló, de valamivel szélesebb és alacsonyabb oldalt eredményez.

A második, **div.header,div.footer** kijelölés az élőfejre és élőlábra vonatkozik; e szürke mezőkben a balra igazított szöveg körüli térköz 0.5em, a betűszín fehér. A **clear:left;** meghatározás az élőfej ill. -láb átméretezéskori elmozdulását illetve a többi elemmel való összetorlódását hivatott meggátolni. Ezt a **clear:both;** meghatározással is elérhetjük.

A harmadik, **h1.header** kijelölés értelmében az élőfejben található **<h1>** elem számára külön térközt és margót állítunk be (melynek értéke 0).

A negyedik, **div.left** kijelöléssel a baloldali szövegmezőt állítjuk be; amely mellett a teret majd a jobboldali fő-mező fogja kitölteni. A bal mező szélessége eszerint 160px, margója nincs, térköze pedig 1em.

Az ötödik, **div.content** kijelölés a fő szövegmezőre vonatkozik. Ennek vékony, szürke bal szegélye van, melyet egy 190px-es bal margó tart távol a bal szövegmező tartalmától (amit pedig a 160px-es **width** beállítása nem enged kiterjedni odáig). A mezőben lévő szöveg térköze 1em. Amint látszik, csak a legkülső, **container** osztályú ** ** elemnek van szegélye; a többi határvonal az előfej és -láb kitöltéséből adódik. Ezen kívül a fő szövegmezőnek van egyetlen, baloldali határvonala; mely azonban a **content** osztályú elemek magasságához igazodik; azaz az élőfej aljától indul, de nem mindig éri el az élőláb tetejét. Ezt a problémát orvosolhatjuk, hogyha a bal szövegmezőnek is beállítunk egy jobb margót, mely az oldal kisebb nagyításai mellett mindig eléri az élőlábat.Hogy a két határvonal egybe essen, a bal szövegmező szélességét 158px-re kell csökkentenünk, azaz:
```
div.left
{
float:left;
width:158px;
margin:0;
padding:1em;
border-right:1px solid gray;
}
div.content
{
margin-left:190px;
border-left:1px solid gray;
padding:1em;
}
```
Minthogy az egyes böngészőkben az alapértelmezett betűméret (em) változhat, a px azonban állandó, a 158px-es beállítás csak a Firefoxra tekinthető érvényesnek; a bal és jobb szövegmezőt elválasztó vonalak más böngészőkben ismét szétválhatnak.

Még egyszerűbben kezelhetjük a problémát, hogyha a **div.header, div.footer {clear:right;}** beállítással élünk!

CSS objektum-igazító jellemzők
Az alábbi táblázatban összefoglaltuk a címben szereplő jellemzőket és azok lehetséges értékeit. A „CSS” oszlopba írt szám azt mutatja, hogy az adott jellemzőt a CSS mely verziója tartalmazta először:

Jellemző	Leírás	Lehetséges értékek	CSS
clear	Specifies which sides of an element where other floating elements are not allowed	left right both none inherit	1
float	Specifies whether or not a box should float	left right none inherit	1

CSS objektum-igazítás:
Igazítási irány: {float:*left/right/none/inherit*;}

Igazítási irány tiltása (clear):
Az elem baloldala után nem igazodhat a következő tartalom: {clear:left;}
Az elem jobboldala után nem igazodhat a következő tartalom: {clear:right;}
Az elem egyik oldala után sem igazodhat tartalom: {clear:both;}
Alapértelmezett eset (általában a both-nak megfelelő): {clear:none;}
Az anya-elem beállításával megegyező: {clear:inherit;}

Tömbszerű elem (aláírással ellátott kép) formázása és igazítása (div kijelöléssel):

```
div {
float:right;
width:100px;
margin:0 0 100px 100px;
padding:100px;
border:100px solid black;
text-align:center;}

<div>
<img src="URL" width="100" height="100" /><br />
Kép-aláírás
</div>
```

Inline elem formázása és igazítása (span kijelöléssel):

```
span {
float:left;
width:1em;
font-size:400%;
font-family:algerian,courier;
line-height:100%;}

<p>
<span>T</span>his is some text.
</p>
```

Vízszintes menüsor készítése:
A **float** jellemzőt az **<ul>**, **<li>** és **<a>** elemeknél egyaránt be kell állítani, a végeredmény kontrollálhatósága érdekében:

```
ul {
float:left;
width:100%;
padding:0;
margin:0;
```

list-style-type:*none*;}

li {
float:*left*;
display:*inline*;}

a {
float:*left*;
width:*6em*;
text-decoration:*none*;
color:*white*;
background-color:*purple*;
padding:*0.2em 0.6em*;
border-right:*1px solid white*;}

a:hover {
background-color:*#ff3300*;}

Link one
Link two
Link three
Link four

Függőleges menüsor készítése:
Az előzőhöz hasonló, de a lista-elemek tömbszerűek (display:block;) és csak magát a listát igazítjuk az oldal szélére (ul {float:*left*;}), a benne lévő elemeknek nem adunk **float** jellemzőt.

Egyszerű weboldal-struktúra:
Élőfej, oldalmenü, tartalom és élőláb **div**-eket hozunk létre, majd az élőfej és élőláb után ill. előtt megtiltjuk az igazodást {clear:both;}, és az oldalmenüt a megfelelő oldalra igazítjuk {float:*left/right/none/inherit*;}.

VI. CSS – vízszintes igazítás

A HTML-elemek vízszintes igazítására számos CSS-jellemzőt használhatunk.

Tömbszerű elemek igazítása

Mint láttuk, a tömbszerű (=block) elemek a rendelkezésre álló teljes szélességet elfoglalják, és üres sorok választják el őket a többitől. Ilyen tömbszerű elem pl. a **<h1>**, a **<p>** és a ** **.

A (függőleges) szövegigazítással már foglalkoztunk a II. CSS-könyv „CSS – szövegformázás" részében.

Most a tömbszerű elemek vízszintes igazításával foglalkozunk.

Középre igazítás margóval

A tömbszerű elemek középre igatzíthatók a bal és jobb margók **auto** értékre állításával. Ez Internet Explorerben csak a **DOCTYPE** beállítása mellett lehetséges.

A margó **auto** beállítása a rendelkezésre álló tér egyenlő megoszlását eredményezi; az elem tehát középre kerül:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
.center
{
margin:auto;
width:70%;
background-color:#b0e0e6;
}
</style>
</head>

<body>
<div class="center">
<p>In my younger and more vulnerable years my father gave me some
advice that I've been turning over in my mind ever since.</p>
<p>'Whenever you feel like criticizing anyone,' he told me, just
remember that all the people in this world haven't had the advantages
that you've had.'</p>
</div>

<p><b>Note: </b>Using margin:auto will not work in Internet Explorer,
unless a !DOCTYPE is declared.</p>
</body>
</html>
```

Példánk ** **-ének szélessége az oldal 70%-a, háttere kék, és a frame közepén látszik, a **margin:auto;** meghatározásnak köszönhetően.

Hogyha az elem mérete az oldalszélesség 100%-a, az igazításnak nincs hatása.

Az Internet Explorer 5 nem tudja megfelelően kezelni a tömbszerű elemeket. A fenti példát ezért át kell alakítanunk, ha IE5-tel is megtekinthetővé kívánjuk tenni:

```
<head>
<style type="text/css">
.container
{
```

```
text-align:center;
}
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
text-align:left;
}
</style>
</head>

<body>
<div class="container">
<div class="center">
<p>In my younger and more vulnerable years my father gave me some
advice that I've been turning over in my mind ever since.</p>
<p>'Whenever you feel like criticizing anyone,' he told me, just
remember that all the people in this world haven't had the advantages
that you've had.'</p>
</div>
</div>

<p><b>Note:</b> In IE 5 there is a margin handling bug for block
elements. Block elements are sometimes treated as inline content. This
is particularly problematic when it comes to centering.
For centering to work in IE5, use the text-align property, like in this
example.</p>

</body>
</html>
```

Mint látjuk, az IE5 a tömbszerű elemeket is hajlamos szövegköziként kezelni, ezért azok középre igazításához a **margin:auto;** helyett a **text-align:center;** meghatározást használjuk.

Böngésző-közi kompatibilitás

A fentihez hasonló elemek igazításakor hasznos, hogyha a **<body>** elemhez pontos margó- és térköz-értéket rendelünk. Ezzel kiküszöbölhetjük a különféle böngészők alapértelmezéseiből adódó megjelenítésbeli eltéréseket.

A **position** jellemző használata elé az Internet Explorer még egy akadályt gördít. Hogyha az igazítandó elemet tartalmazó elem szélességét is megadjuk (itt a **container** osztályú ** **), és a **DOCTYPE** hiányzik, az IE egy 17 pixeles margót ad a jobboldalhoz; a csúszkának szánt hely gyanánt.
Ezért a **position** jellemző használatakor mindig állítsuk be a **DOCTYPE**-ot!

Ennek megfelelő beállítást látunk a következő példában:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
body
{
margin:0;
padding:0;
```

```
}
.container
{
position:relative;
width:100%;
}
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
</style>
</head>

<body>
<div class="container">
<div class="right">
<p><b>Note: </b>When aligning using the position property, always
include the !DOCTYPE declaration! If missing, it can produce strange
results in IE browsers.</p>
</div>
</div>
</body>
</html>
```

A **DOCTYPE** megadása után beállítjuk a dokumentumtest margóit és térközeit, majd egy tároló-elem helyzetét és szélességét. Ezután műr az IE számára is értelmezhetően tudjuk az alérendelt szövegmezőt a tároló-elem jobb felső sarkába igazítani:

```
.right
{
postition:absolute;
top:0px;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

azaz a szövegmező az őt tartalmazó (**container** osztályú) elem felső és jobboldali élétől egyaránt 0 pixelnyire kezdődik (amint azt tapasztalhatjuk, hogyha a **conatiner** elemnek is a jobb-felső saroktól eltérő, abszolút pozíciót adunk:

```
.container
{
position:absolute;
top:5px;
right:5px;
border:2px solid red;
}
```

Oldalra igazítás float jellemzővel

Az elemek igazítására a **float** jellemzőt is felhasználhatjuk:

```
<html>
<head>
<style type="text/css">
.right
{
```

```
float:right;
width:300px;
background-color:#b0e0e6;
}
</style>
</head>

<body>
<div class="right">
<p>In my younger and more vulnerable years my father gave me some
advice that I've been turning over in my mind ever since.</p>
<p>'Whenever you feel like criticizing anyone,' he told me, just
remember that all the people in this world haven't had the advantages
that you've had.'</p>
</div>
</body>
</html>
```

Példánkban a szövegmező a frame jobb-felső sarkába kerül, de annak élétől 7-8px üres hely (alapértelmezett margó) választja el. Hogyha a belső CSS-hez hozzáadjuk a

```
body
{
margin:0;
padding:0;
}
```

beállítást, a mező pontosan a jobb-felső sarokba illeszkedik.

Böngésző-közi kompatibilitás

Az utóbbi igazítási módszernél tehát érdemes a **<body>** elem számára egy-egy pontos margó- és térköz-értéket beállítani. Ezzel megint csak a különféle böngészők közti megjelenítésbeli eltéréseket eliminálhatjuk.

Az IE-nek azonban a **float** jellemző használata is gondot okoz. Ha nincs megadva a **DOCTYPE**, az IE megint 17px margót ad a jobboldalhoz.

Ezért a **float** jellemző hasznéálatakor is mindis adjuk meg a **DOCTYPE**-ot:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
body
{
margin:0;
padding:0;
}
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
</style>
</head>

<body>
<div class="right">
```

```
<p><b>Note: </b>When aligning using the float property, always include
the !DOCTYPE declaration! If missing, it can produce strange results in
IE browsers.</p>
</div>
</body>
</html>
```

CSS – vízszintes igazítás:

Középre igazítás:

Margóval (Firefox): {margin:auto;}

Margóval (IE, csak DOCTYPE mellett):

A pozícionálandó elemet tartalmazó elemre nézve: {text-align:center;}

A pozívionálandó elemre nézve: {margin-left:auto;margin-right:auto;}

Vízszintes igazítás position jellemzővel:

Előfeltételek: 1.) Abszolút pozícionálás esetén a pozícionálandó elemet tartalmazó elem pozícionálása nem lehet statikus.
2.) Az eltérő alapértelmezések kiküszöbölésére állítsunk be pontos térköz- és margóértékeket a **<body>**-ra nézve (pl.: {padding:0;margin:0;})!
3.) Az IE-rel való használhatóság érdekében mindig állítsunk be megfelelő DOCTYPE-ot!

Pozícionálási módok:

Statikus: {position:static;}
Fix: {position:fixed;}
Relatív: {position:relative;}
Abszolút: {position:absolute;}

Pozícionálási értékek:

Felső: {top:*100px/100em/100%/auto/inherit*;}
Jobb: {right:*100px/100em/100%/auto/inherit*;}
Alsó: {bottom:*100px/100em/100%/auto/inherit*;}
Bal: {left:*100px/100em/100%/auto/inherit*;}

Az érték megadja az adott oldali elemmargó vonalának távolságát a frame azonos szélétől.

Vízszintes igazítás float jellemzővel:

Előfeltétel: 1.) Az eltérő alapértelmezések kiküszöbölésére állítsunk be pontos térköz- és margóértékeket a **<body>**-ra nézve (pl.: {padding:0;margin:0;})!
2.) Az IE-rel való használhatóság érdekében mindig állítsunk be megfelelő DOCTYPE-ot!

Igazítási irány: {float:*left/right/none/inherit*;}
Igazítási irány tiltása: {clear:*left/right/both/none/inherit*;}

VII. CSS ál-osztályok

A CSS ál-osztályokkal különleges formázási hatásokat érhetünk el néhány kijelölésnél.

Mondattan

Az ál-osztályok mondattana:

```
kijelölés:ál-osztály {jellemző:érték;}
```

A CSS-osztály és ál-osztály – kijelölések ötvözhetők:

```
kijelölés.osztály:ál-osztály {jellemző:érték;}
```

Link/könyvjelző – ál-osztályok

Az ál-osztályok legjellemzőbb alkalmazása a linkek formázása, a CSS-t támogató böngészők számára:

```
<html>
<head>
<style type="text/css">
a:link {color:#FF0000;} /* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
</style>
</head>

<body>
<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS
definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition
in order to be effective.</p>
</body>
</html>
```

A példánkban szerepló link látogatatlanul piros, látogatottan zöld, egérrel érintve lila és kattintva kék.
Az egyes link-állapotokhoz tartozó ál-csopotoknak mindig a fenti sorrendben kell követniük egymást.

Az ál-csoportok nevei nem nagy- ill. kisbetű-érzékenyek.

Ál-osztályok és CSS-osztályok

Az ál-osztályok a CSS-osztályokkal kombinálhatók:

```
a.red:visited {color:#ff0000;}

<a class=”red” href=”css_syntax.asp”>CSS Syntax</a>
```

Hogyha az előbbi link állapota látogatott (**visited**), akkor színe piros lesz.

CSS – a :first-child ál-osztály

A **:first-child** ál-osztály mindazon elemekre vonatkozik, melyek egy adott osztályba=csoportba tartozó elemkötegek első tagjai. A **:first-child** kijelöléscsak a **DOCTYPE** megadása mellett működik.

Az első <p> elem kijelölése

Az alábbi példában a kijelölés mindazon **<p>** elemre vonatkozik, melyek valamely több elemet tartalmazóü elem első tagjai:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style type="text/css">
p:first-child {color:blue;}
```

```
</style>
</head>

<body>
<p>This is some text.</p>
<p>This is some text.</p>
<p><b>Note:</b> For :first-child to work in IE, a DOCTYPE must be
declared.</p>
</body>
</html>
```

Itt az első bekezdés kék színű lesz, hiszen az a **<body>** elembe tartozó *első* elem. A példát egy kissé átalakíthatjuk:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style type="text/css">
p:first-child {color:blue;}
span p {display:inline;}
</style>
</head>

<body>
<p>This is some text.</p>
<p>This is some text.</p>
<span>
<p>Hova</p>
<p>megyünk?</p>
</span>
<p><b>Note:</b> For :first-child to work in IE, a DOCTYPE must be
declared.</p>
</body>
</html>
```

Itt a **** elem első rész-eleme, a **Hova** bekezdés is kék lesz, az első bekezdéshez hasonlóan. Ezen túlmenően, mint a belső stílus-oldalban látjuk, a ****-ba tartozó **<p>** elemek nem külön sorokban, hanem egy sorban jelennek meg; köztük tehát nem sorköz, hanem szóköz áll (habár sem a **Hova**, sem a **megyünk?** bekezdés nem tartalmaz szóközt). Így tehát elértük, hogy egy bekezdésnek tűnő szövegünk kétféle színben jelenjen meg; ez pedig csakis az ál-csoportok és megjelenítési beállítás alkalmazásával vált lehetségessé.

A <p> elemekben lévő első <i> elemek kijelölése

Következő példánkban a dokumentum **<p>** elemeibe tartozó, első **<i>** elemeket formázzuk:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style type="text/css">
p > i:first-child {color:blue;}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p><b>Note:</b> For :first-child to work in IE, a DOCTYPE must be
```

```
declared.</p>
</body>
</html>
```
Itt az első két bekezdés első dölt betűvel álló szavai kékek.

A :first-child <p> elemek összes <i> al-elemének kijelölése
Következő példánkban az elemcsoportok első **<p>** elemeibe tartozó összes **<i>** elemet látjuk el azonos formázással:
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style type="text/css">
p:first-child i {color:blue;}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p><b>Note:</b> For :first-child to work in IE, a DOCTYPE must be
declared.</p>
</body>
</html>
```
Itt az első bekezdés minkét dőltbetűvel szedett szava kék.

A :lang ál-osztály (nyelvtani csoport)
A **:lang** ál-osztály különleges nyelvi/jelölési szabályok meghatározását teszi lehetővé. Az Internet Explorer csak **DOCTYPE** beállítása mellett képes értelmezni.

Az alábbi példában a **no** jelű **:lang**-ba (nyelvtani csoportba) tartozó **<q>** elemeket számára a szokásostól eltérő idézőjeleket határozunk meg:
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style type="text/css">
q:lang(no) {quotes: "~" "~";}
</style>
</head>

<body>
<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
<p>In this example, :lang defines the quotation marks for q elements
with lang="no":</p>
<p><b>Note:</b> Internet Explorer 8 (and higher) supports the :lang
pseudo class
if a !DOCTYPE is specified.</p>
</body>
</html>
```
Itt tehát a **<q>** elembe foglalt szöveg előtt és után nem időzőjel, hanem hullámvonal (~) látható.

További példák
Első példánkban különféle link-stílusok beállítását látjuk:
<html>

```
<head>
<style type="text/css">
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}

a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}

a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style>
</head>

<body>
<p>Mouse over the links to see them change layout.</p>
<p><b><a class="one" href="default.asp" target="_blank">This link
changes color</a></b></p>
<p><b><a class="two" href="default.asp" target="_blank">This link
changes font-size</a></b></p>
<p><b><a class="three" href="default.asp" target="_blank">This link
changes background-color</a></b></p>
<p><b><a class="four" href="default.asp" target="_blank">This link
changes font-family</a></b></p>
<p><b><a class="five" href="default.asp" target="_blank">This link
changes text-decoration</a></b></p>
</body>
</html>
```

A linkek, mint látjuk, látogatatlan állapotban pirosak, látogatott állapotban pedig kékek.
(**hover** vagy **active** ál-csoportnak megfelelő) aktív állapotában az első színe sárgára vált, a második betűmétere másfélszeresére nő, a harmadik háttere zöld lesz, a negyedik betűje **monospace**, az ötödik pedig aláhúzást nyer. Habár tehát az aktív állapothoz tartozók közül csak a **:hover** ál-osztályt specifikáltuk, a link kattintása esetén is ez a formázás érvényes, mert nem adtunk meg erre az esetre mást, és *kattintáskor* az egér mindenképpen a *link fölött van*.

A következőkben a **:focus** ál-osztály alkalmazására látunk példát:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
input:focus {background-color:yellow;}
</style>
</head>
```

```
<body>
<form action="form_action.asp" method="get">
First name: <input type="text" name="fname" value="Sanyi" /><br />
Last name: <input type="text" name="lname" value="néni" /><br />
<input type="submit" value="Submit" />
</form>
<p><b>Note:</b> Internet Explorer 8 (and higher) supports the :focus
pseudo-class if a !DOCTYPE is specified.</p>
</body>
</html>
```

Itt a két szövegbeviteli mezők valamelyikébe kattintva, annak színe (melyre „fókuszálunk”) sárgára vált. A mezőkbel a „Sanyi” és „néni” alapértelmezett szövegek találhatók. A „Submit” gomb megnyomásakor Sanyi néni neve továbbításra kerül.

CSS ál-osztályok

Az alábbi táblázatban összefoglaltuk az ál-osztályokat és azok lehetséges értékeit. A „CSS” oszlopba írt szám azt mutatja, hogy az adott ál-osztályt a CSS mely verziója tartalmazta először:

Ál-osztály	Leírás	CSS
:active	Adds a style to an element that is activated	1
:first-child	Adds a style to an element that is the first child of another element	2
:focus	Adds a style to an element that has keyboard input focus	2
:hover	Adds a style to an element when you mouse over it	1
:lang	Adds a style to an element with a specific lang attribute	2
:link	Adds a style to an unvisited link	1
:visited	Adds a style to a visited link	1

CSS ál-osztályok: **[A DOCTYPE beállítása az IE részére itt is előfeltétel!]**

Mondattan:

Ál-osztály: elem:ál-osztály {jellemző:érték;}

Elem-osztályon belüli ál-osztály: elem.osztály:ál-osztály {jellemző:érték;}

Elemen belüli elem (ál-osztállyal): elem elem:ál-osztály {jellemző:érték;} vagy
elem > elem:ál-osztály {jellemző:érték;}

Adott ál-osztályba tartozó elemeken belüli elem: elem:ál-osztály elem {jellemző:érték;} vagy
elem:ál-osztály > elem {jellemző:érték;}

Ál-osztályok:

Link (nem nézett): :link

Link (megtekintett): :visited

Egér-érintés: :hover

Aktív állapot: :active

Első al-elem: :first-child

[Pl.: **p:first-child** kijelölés esetén a dokumentum minden szintű elemébe tartozó első bekezdés adott stílusú lesz.]

Nyelvtani ál-osztály: :lang(*név*)

[Pl.: **q:lang(*név*) {quotes:”~” ”~”;}** formázás és
<q lang=”*név*”>*Szöveg*</q>
formázott elem (idézet) esetén az idézet elején és végén idézőjel helyett ~ áll.]

Kijelölt szövegbeviteli elem: :focus

[Pl.: **input:focus {background-color:yellow;}** beállítás esetén használatban lévő szövegbeviteli mezők háttere sárga lesz.]

VIII. CSS ál-elemek

A CSS ál-elemeket is különleges kijelölésekre alkalmazzuk.

Mondattan

Az ál-elemek modattana:

```
kijelölés:ál-elem {jellemző:érték;}
```

A CSS-osztályok ál-elemekkel együtt is használhatók:

```
kijelölés.osztály:ál-elem {jellemző:érték;}
```

A :first-line ál-elem

A **:first-line** ál-elemet szövegek első sorának formázására használunk.

Példánkban a **<p>** elem első sorát a **first-line** nevű ál-elem-meghatározás szerint formázza a böngésző. Az első sor mérete természetesen a böngésző-ablak méretétől is függhet (automatikus tördelés esetén).

```
<html>
<head>
<style type="text/css">
p:first-line
{
color:#ff0000;
font-variant:small-caps;
}
</style>
</head>

<body>
<p>You can use the :first-line pseudo-element to add a special effect
to the first line of a text.</p>
<p>You can use the :first-line pseudo-element to add a special effect
to the first line of a text.</p>
<p>You can use the :first-line pseudo-element to add a special effect
to the first line of a text.</p>
</body>
</html>
```

Itt a weboldal mindgárom bekezdésének első sora piros kiskapitális betűkkel jelenik meg.

A **:first-line** ál-elemet csak tömbszerű HTML-elemekre vonatkoztathatjuk. **:first-line** kijelölés mellett a következő jellemzők állíthatók be: **font**, **color**, **background**, **word-spacing**, **letter-spacing**, **text-decoration**, **vertical-align**, **text-transform**, **line-height**, **clear**.

A :first-letter ál-elem

A **:first-letter** ál-elem szövegek első betűinek egyedi formázására szolgál:

```
<html>
<head>
<style type="text/css">
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
</style>
</head>

<body>
```

```
<p>You can use the :first-letter pseudo-element to add a special effect
to
the first character of a text!</p>
<p>You can use the :first-letter pseudo-element to add a special effect
to the first character of a text!</p>
<p>You can use the :first-letter pseudo-element to add a special effect
to the first character of a text!</p>
</body>
</html>
```

Példánk mindhárom bekezdése egy **xx-large**, piros színű Y-nal kezdődik.

A **:first-letter** ál- elemet is csak tömbszerű HTML-elemek formázására használhatjuk. A beállítható jellemzők: **font**, **color**, **background**, **margin**, **padding**, **border**, **text-decoration**, **vertical-align** (csak akkor, ha **float:none;**), **text-transform**, **line-height**, **float**, **clear**.

Ál-elemek és CSS-osztályok

Az ál-elemek, az ál-osztályokhoz hasonlóan, szintén kombinálhatók a hagyományos CSS-osztályokkal:

```
p.név:ál-elem {jellemző:érték;}

<p class=”név”>Szöveg</p>
```

E beállítás az összes **név** osztályú bekezdés (**<p>** elem) kezdőbetűjére vonatkozik.

Többszörös ál-elemek

Sok ál-elemet egymással is kombinálhatunk.

A következő példában a belezdések első betűi pirosak és nagyméretűek lesznek, az első sorok többi betűje kék és kiskapitális; a bekezdések további részének megjelenése pedig alapértelmezett:

```
<html>
<head>
<style type="text/css">
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
p:first-line
{
color:#0000ff;
font-variant:small-caps;
}
</style>
</head>

<body>
<p>You can combine the :first-letter and :first-line pseudo-elements to
add a special effect to the first letter and the first line of a
text!</p>
<p>You can combine the :first-letter and :first-line pseudo-elements to
add a special effect to the first letter and the first line of a
text!</p>
<p>You can combine the :first-letter and :first-line pseudo-elements to
add a special effect to the first letter and the first line of a
text!</p>
</body>
</html>
```

A fenti beállítások mindhárom bekezdésre érvényesek.

CSS – a :before ál-elem
A **:before** ál-elemmel bizonyos tartalmat szúrhatunk be a kijelölt elemek tartalma elé.

A következő példában minden **<h1>** elé egy képet szúrunk be:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
h1:before {content:url(smiley.gif);}
</style>
</head>

<body>
<h1>This is a heading</h1>
<p>The :before pseudo-element inserts content before an element.</p>
<h1>This is a heading</h1>
<p><b>Note:</b> Internet Explorer 8 (and higher) supports the content
property
if a !DOCTYPE is specified.</p>
</body>
</html>
```

A példában szereplő mindkét címsor előtt egy-egy kép szerepel.

CSS – az :after ál-elem
Az **:after** ál-elemmel minden kijelölt elem mögé beszúrhatunk egy bizonyos tartalmat.

A következő példában minden címsor után egy kis kép látható:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
h1:after {content:url(smiley.gif);}
</style>
</head>

<body>
<h1>This is a heading</h1>
<p>The :after pseudo-element inserts content after an element.</p>
<h1>This is a heading</h1>
<p><b>Note:</b> Internet Explorer 8 (and higher) supports the content
property if a !DOCTYPE is specified.</p>
</body>
</html>
```

CSS ál-elemek
Az alábbi táblázatban összefoglaltuk az ál-elemeket és azok lehetséges értékeit. A „CSS” oszlopba írt szám azt mutatja, hogy az adott ál-elemet a CSS mely verziója tartalmazta először:

Ál-elem	Leírás	CSS
:after	Adds content after an element	2
:before	Adds content before an element	2

:first-letter	Adds a style to the first character of a text	1
:first-line	Adds a style to the first line of a text	1

CSS ál-elemek:

Mondattan:

Ál-elem: elem:ál-elem {jellemző:érték;}

Elem-osztályon belüli ál-elem: elem.osztály:ál-elem {jellemző:érték;}

Elemen belüli elem (ál-elemmel): elem elem:ál-elem {jellemző:érték;} vagy
elem > elem:ál-elem {jellemző:érték;}

Ál-elemek:

Első sor: :first-line

[A **:first-line** ál-elemet csak tömbszerű HTML-elemekre vonatkoztathatjuk. **:first-line** kijelölés mellett a következő jellemzők állíthatók be: **font**, **color**, **background**, **word-spacing**, **letter-spacing**, **text-decoration**, **vertical-align**, **text-transform**, **line-height**, **clear**.]

Első betű: :first-letter

[A **:first-letter** ál- elemet is csak tömbszerű HTML-elemek formázására használhatjuk. A beállítható jellemzők: **font**, **color**, **background**, **margin**, **padding**, **border**, **text-decoration**, **vertical-align** (csak akkor, ha **float:none;**), **text-transform**, **line-height**, **float**, **clear**.]

Megelőző tartalom: :before

[A **:before** ál-elemmel bizonyos tartalmat szúrhatunk be a kijelölt elemek tartalma elé.
Pl.: **h1:before {content:url(smiley.gif);}** beállítás esetén minden **<h1>** elem előtt egy kép lesz.]

Lezáró tartalom: :after

[Az **:after** ál-elemmel minden kijelölt elem mögé beszúrhatunk egy bizonyos tartalmat.
Pl.: **h1:after {content:url(smiley.gif);}** beállítás esetén minden **<h1>** elem után egy kép lesz.]

IX. CSS – navigációs sáv

Navigációs sávok

A könnyen használható navigácós elemek megléte minden weboldalon fontos.
A CSS segítségével a purián HTML-menüket tetszetős navigációs mezőkké alakíthatjuk.

Navigációs sáv = link-lista

A navigációs listák egyszerű HTML-kódon alapulnak. Többnyire linkek **<ul>** ill. **<li>** elemekből felépülő HTML-listáinak formázásával készítjük őket, amint azt az alábbi példákban látjuk.

Először lássunk egy link-listát:

```
<html>
<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
<li><a href="#about">About</a></li>
</ul>

<p>Note: We use href="#" for test links. In a real web site this would
be URLs.</p>
</body>
</html>
```

A linkek itt nem létező oldalakra mutatnak, de ez az érthetőséget nem zavarja.

Távolítsuk most el a lista-jeleket, és igazítsuk az oldal (bal) legszélére a listát:

```
<html>
<head>
<style>
ul
{
list-style-type:none;
margin:0;
padding:0;
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
<li><a href="#about">About</a></li>
</ul>
</body>
</html>
```

Itt tehát kikapcsoltuk a lista-jeleket, és a böngésző-alapértelmezéstől eltérve, a margót és térközt minden oldalon egyaránt nullának véve, az oldal szélére helyeztük a listát. Annak tömbszerű elemei azonban továbbra is egymás alatt találhatók.

Az így elklszült lista megfelelő kiindulásként szolgál mind vízszintes, mind függőleges navigációs sávok készítésére.

Függőleges navigációs sáv

Függőleges navigációs sávot kialakításához a listában lévő **<a>** elemek formázásával készíthetünk:

```
<html>
<head>
<style type="text/css">
ul
{
list-style-type:none;
margin:0;
padding:0;
}
a
{
display:block;
width:60px;
background-color:#dddddd;
margin-bottom:2px;
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
<li><a href="#about">About</a></li>
</ul>
<p>A background color is added to the links to show the link area.</p>
<p>Notice that the whole link area is clickable, not just the text.</p>
</body>
</html>
```

Itt az összes link 60px széles, világoskék mezőkben található, melyeket között 2px margó van. Nemcsak az **<a>** elembe írt szöveg, hanem a teljes „doboz”-terület linkként viselkedik, azaz „kattintható”.

A **display:block;** meghatározás azért szükséges az **a** kijelölésre vonatkozólag, mert a könyvjelzők alapértelmezésben sorbazárt (inline) elemek. Emellett a CSS-ben a további jellemzők (pl. szélesség) használata csak a **display** jellemző megadása után szabályos; hogy az esetleges eltérő alapértelmezésekből adódó megjelenítési eltéréseket kiküszöböljük.

A **width:60px;** meghatározás pedig a tömbszerű elemek elvilag egész oldalszélességre kiterjedő méretét 60px-re korlátozza. Ha kitöröljük, a lista-elemek az oldal jobb széléig érnek. A vízszintes navigációs mezőkbe tartozó **<a>** elemek szélességét ezért mindig adjuk meg, különösen, mert az Internet Explorer ennek elhagyása esetén hibásan jelenítheti meg az oldalt.

A következő példában egy kidolgozottabb függőleges menüt látunk:

```
<html>
<head>
<style type="text/css">
ul
{
list-style-type:none;
margin:0;
padding:0;
}
a:link,a:visited
```

```
{
display:block;
font-weight:bold;
color:#FFFFFF;
background-color:#98bf21;
width:120px;
text-align:center;
padding:4px;
text-decoration:none;
text-transform:uppercase;
margin-bottom:3px;
}
a:hover,a:active
{
background-color:#7A991A;
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
<li><a href="#about">About</a></li>
</ul>
</body>
</html>
```

Itt az alap-listára vonatkozó **ul** kijelölést követően, az **a:link,a:visited** kijelölések alatt, a linkeket ugyancsak tömbszerű, 120px széles elemekként definiáljuk. Mint látszik, a betűhatásokat, színeket is széleskörűen beálítottuk, valamint a link-szöveg körül 4px-es térköz található. Az egyes link-gombokat pedig 3px-es margók választják el egymástól.
hover ill. **active** állapotban a linkek háttér-színe megváltozik (sötétebb lesz).

Vízszintes böngészősáv kialakítása
Vízszintes böngészősávot kétféleképpen készíthetünk; sorbazárt vagy igazított listaelemekből.

Mindkét módszer eredményesen alkalmazható; de ha a gombok méretének egyeznie kell, akkor az igazításos (**floating**) módszert válasszuk!

Vízszintes böngészősáv sorbazárt listaelemekből
A vízszintes böngészősáv létrehozásának egyik módja, ha a **<li>** elemeket sorbazárttá tesszük; amint azt a fenti alap-lista alábbi, átalakított változatában is láthatjuk:

```
<html>
<head>
<style type="text/css">
ul
{
list-style-type:none;
margin:0;
padding:0;
}
li
{
display:inline;
```

```
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
<li><a href="#about">About</a></li>
</ul>
</body>
</html>
```

Ekkor az egyébként sortöréssel elválasztott listaelemek (tehát a linkek) egy sorban, szóközökkel elválasztva látszanak. A **<li>** elemek tömbszerűségére vonatkozó böngésző-alapértelmezést tehát felülírtuk egy ellenkező megjelenítési utasítással.

A következő példában egy azonos módszerrel készült, igényesebb vízszintes menüsort találunk:

```
<html>
<head>
<style type="text/css">
ul
{
list-style-type:none;
margin:0;
padding:0;
padding-top:6px;
padding-bottom:6px;
}
li
{
display:inline;
}
a:link,a:visited
{
font-weight:bold;
color:#FFFFFF;
background-color:#98bf21;
text-align:center;
padding:6px;
text-decoration:none;
text-transform:uppercase;
}
a:hover,a:active
{
background-color:#7A991A;
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
```

```
<li><a href="#about">About</a></li>
</ul>
<p><b>Note:</b> If you only set the padding for a elements (and not
ul), the links will go outside the ul element. Therefore, we have added
a top and bottom padding for the ul element.</p>
</body>
</html>
```

Itt a vízszintes menüsor linkjei eltérő szélességű, 6px-es térközű dobozokban sorakoznak egymás mellett, szóközökkel elválasztva. A menüsor fölött és alatt szintén 6px térköz van.

A gombok méretét egyezővé tehetjük, hogyha az **a:link,a:visited** kijelöléshez beírjuk a **width:90px;** meghatározást is. Egymástól való távolságukat a **margin-right:13px;** meghatározással módosíthatjuk. E meghatározás értéke (13px) azonban a továbbra is fennálló szóközhöz hozzáadva értendő!

Ezt a menüsort könnyen átalakíthatjuk függőlegessé is:

```
<html>
<head>
<style type="text/css">
ul
{
list-style-type:none;
margin:0;
padding:0;
padding-top:6px;
padding-bottom:6px;
}
li
{
display:block;
}
a:link,a:visited
{
font-weight:bold;
color:#FFFFFF;
background-color:#98bf21;
text-align:center;
height:20px;
width:90px;
padding:6px;
margin-bottom:12px;
text-decoration:none;
text-transform:uppercase;
}
a:hover,a:active
{
background-color:#7A991A;
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
```

```
<li><a href="#about">About</a></li>
</ul>
<p><b>Note:</b> If you only set the padding for a elements (and not ul), the links will go outside the ul element. Therefore, we have added a top and bottom padding for the ul element.</p>
</body>
</html>
```

Itt a szélesség megadásával ismét biztosítjuk a link-mezők egyenlő méretét; a magasság, térköz és margó beállításával pedig az egymásratorlódásukat akadályozzuk meg (ti. a linkek sortörésnyi távolságban követnék egymást, ami miatt, ha nem állítjuk be a dobozok szélességét vagy magasságát, azok egymásra torlódnak, és az alsók térközei elfedik a felsők betűit.).

E beállítás mellett az összetorlódott dobozok már elkülönülnek:

```
a:link,a:visited
{
font-weight:bold;
color:#FFFFFF;
background-color:#98bf21;
text-align:center;
width:1px;
padding:6px;
margin-bottom:1px;
text-decoration:none;
text-transform:uppercase;
}
```

A margó csak a dobozok méretének észleléséhez szükséges. Ha a szélességet kellően megnöveljük, az összes link egyező méretű lesz. Hogyha ekkor hozzadjuk a formázáshoz a **height** jellemzőt (pl. **height:50px;** meghatározásban), akkor a sorok alapértelmezett magassága a betűktől lefelé megnő; így a szöveg függőlegesen kikerül a korábban a 6px-es térközzel biztosított, központi pozícióból. Újra helyreállíthatjuk a függőleges középponti helyzetet, ha a felső térközt is megnöveljük, pl. 14px-re. Végül kiegyenlíthetjük a gombok szélesség-egyenetlenségét, a **width:110px;** meghatározással.

Vízszintes menüsor igazított listaelemekből

Az előbbi példa egyszerűbb változataiban a listaelemek szélessége változó volt. Mint láttuk, a **width** jellemzőnek az **<a>** elemekhez való hozzáadásával ezt a problémát megoldottuk.

A következő pélában az ily módon egyenlő szélességűvé tett linkek által alkotott listaelemeket egymás mellé igazítjuk:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
ul
{
list-style-type:none;
margin:0;
padding:0;
overflow:hidden;
}
li
{
float:left;
}
a
```

```
{
display:block;
width:60px;
text-align:center;
background-color:#dddddd;
margin-right:4px;
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
<li><a href="#about">About</a></li>
</ul>
<p><b>Note:</b> If a !DOCTYPE is not specified, floating items can
produce unexpected results.</p>
<p>A background color is added to the links to show the link area. The
whole link area is clickable, not just the text.</p>
<p><b>Note:</b> overflow:hidden is added to the ul element to prevent
li elements from going outside of the list.</p>
</body>
</html>
```

Mint látjuk, itt a **li** kijelölésnél a **display** jellemző helyett a **float**-tal érjük el az egy sorba rendezést. Az **ul** kijelölésbe írt **overflow:hidden;** meghatározás meggátolja, hogy az oldal átméretezéskori automatikus tördelésekor a listaelemek alkalmatlan helyre ugorjanak. Itt tehát az átméretezés nem okoz gondot, a legkisebb szélességűre méretezett ablakban a menüsor függőlegesnek látszik, de elemeui továbbra is jólo elkülöníthetők. Hogyha az előző (**display** jellemzőn alapuló) módszerrel készített menü oldalát átméretezzük, akkor is függőlegessé válik a menü, csak elemei egymásra torlódnak;ezért célszerű ott is megadni a **height**, **width** és **margin** jellemzőket; akkor ott sincs probléma. Ugyanezen okból a jelenpéldánál is érdemes volna minden oldalra kiterjedő margót megadni, hogy a függőlegessé váló menüsor elemei is elkülönüljenek. Mindenesetre itt a szürke link-mezőket 4px-es jobb margók választják el egymástól, a szöveget pedig középre igazítottuk.

Az **<a>** elemek **display:block;** beállítása itt azért van, hogy a link-mező egész területe „klikkelhető” legyen, ne csak a szöveg. A szélességet pedig azért kell mgadnunk, mert egyébként mindegyik listaelem (a **block** beállítás folytán) az egész ablakszélességet elfoglalná; tehát nem volna hely a linkek egymás utáni sorakozására.

Következő példánkban egy igazításos módszerrel készült, komolyabb vízszintes menüsort látunk:

```
<html>
<head>
<style type="text/css">
ul
{
list-style-type:none;
margin:0;
padding:0;
overflow:hidden;
}
li
{
float:left;
```

```
}
a:link,a:visited
{
display:block;
width:120px;
font-weight:bold;
color:#FFFFFF;
background-color:#98bf21;
text-align:center;
padding:4px;
margin:2px;
text-decoration:none;
text-transform:uppercase;
}
a:hover,a:active
{
background-color:#7A991A;
}
</style>
</head>

<body>
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#news">News</a></li>
<li><a href="#contact">Contact</a></li>
<li><a href="#about">About</a></li>
</ul>
</body>
</html>
```

Ez csak színeiben és betűiben különbözik az előzőtől. Az **<a>** elemek körüli margónak és az **ul {overflow:hidden;}** beállításnak köszönhetően a menüsor elemeinek megjelenése minden ablakméretnél azonos logikai kapcsolatot reprezentál.

CSS – navigációs sáv:

Kiindulási alap (linkek nem-hierarchikus listája):

ul {list-style-type:none;margin:*0*;padding:*0*;}

```
<ul>
<li><a href="URL">Szöveg</a></li>
<li><a href="URL">Szöveg</a></li>
<li><a href="URL">Szöveg</a></li>
<li><a href="URL">Szöveg</a></li>
</ul>
```

Függőleges navigációs sáv (a kiindulási alap kiegészítésével):

Egyszerű függőleges menü, tömbszerű (változó szélességű) listaelemekből:

a {display:block;width:*60px*;background-color:*#dddddd*;margin-bottom:*2px*;}

Kidolgozottabb függőleges menü, tömbszerű (egyenlő szélességű) listaelemekből:

li {display:inline;}

a:link,a:visited
{display:block;font-weight:*bold*;color:*#ffffff*;background-color:*#98bf21*;width:*110px*;text-align:*center*;padding:*25px*;text-decoration:*none*;text-transform:*uppercase*;margin-bottom:*10px*;}

a:hover,a:active

{background-color:*#7a991a*;}

Vízszintes navigációs sáv (a kiindulási alap kiegészítésével):

Egyszerű vízszintes menü, sorbazárt (változó szélességű) listaelemekből:

li {display:*inline*;}

Kidolgozottabb vízszintes menü, sorbazárt (változó szélességű) listaelemekből:

li {display:*inline*;}

a:link,a:visited
{font-weight:*bold*;color:*#ffffff*;background-color:*#98bf21*;text-align:*center*;padding:*6px*;text-decoration:*none*;text-transform:*uppercase*;}

a:hover,a:active
{background-color:*#7a991a*;}

Egyszerű vízszintes menü, igazított listaelemekből:

ul {list-style-type:none;margin:0;padding:0;overflow:hidden;}

li {float:left;}

a {display:block;width:60px;text-align:center;background-color:#dddddd;margin-right:4px;}

Kidolgozottabb vízszintes menü, igazítitt listaelemekből:

ul {list-style-type:none;margin:0;padding:0;overflow:hidden;}

li {float:left;}

a:link,a:visited
{display:block;width:120px;font-weight:bold;color:#FFFFFF; background-color:#98bf21;text-align:center;padding:4px; margin:2px;text-decoration:none;text-transform:uppercase;}

a:hover,a:active
{background-color:#7A991A;}

Vízszintes böngészősávot kétféleképpen készíthetünk; sorbazárt vagy igazított listaelemekből.

Mindkét módszer eredményesen alkalmazható; de ha a gombok méretének egyeznie kell, akkor az igazításos (**floating**) módszert válasszuk!

X. CSS – képgaléria

Az alábbiakban egy képgaláriát állítunk össze a CSS révén:

```
<html>
<head>
<style type="text/css">
div.img
{
  margin: 2px;
  border: 1px solid #0000ff;
  height: auto;
  width: auto;
  float: left;
  text-align: center;
}
div.img img
{
  display: inline;
  margin: 3px;
  border: 1px solid #ffffff;
}
div.img a:hover img {border: 1px solid #0000ff;}
div.desc
{
  text-align: center;
  font-weight: normal;
  width: 120px;
  margin: 2px;
}
</style>
</head>

<body>
<div class="img">
 <a target="_blank" href="klematis_big.htm"><img
src="klematis_small.jpg" alt="Klematis" width="110" height="90" /></a>
 <div class="desc">Add a description of the image here</div>
</div>

<div class="img">
 <a target="_blank" href="klematis2_big.htm"><img
src="klematis2_small.jpg" alt="Klematis" width="110" height="90" /></a>
 <div class="desc">Add a description of the image here</div>
</div>

<div class="img">
 <a target="_blank" href="klematis3_big.htm"><img
src="klematis3_small.jpg" alt="Klematis" width="110" height="90" /></a>
 <div class="desc">Add a description of the image here</div>
</div>

<div class="img">
 <a target="_blank" href="klematis4_big.htm"><img
src="klematis4_small.jpg" alt="Klematis" width="110" height="90" /></a>
 <div class="desc">Add a description of the image here</div>
</div>
```

```
</body>
</html>
```
Az eredmény: egy sorban, kék keretben fehér (azaz láthatatlan) keretes képek jelennek meg, alattuk középre igazított szöveggel. Hogyha az egeret egy kép fölé visszük, annak szegélye kékre vált.

A kódot értelmezve látható, hogy a **div.img** kijelölés a képeket és az aláírásokat tartalmazó keretet formázza. Vízszintes menüsort alakítunk ki a **float:left;** meghatározással. A keret szélessége és magassága automatikusan a benti tartalomhoz igazodik, amit ellenőrizhetünk, ha a szövegszélességet megnöveljük (pl. **div.desc {width:300px;}**). A keret szélessége is megnő. Hogyha azonban fix keret-szélességet állítottnk be (pl. **div.img {width:200px;}**), akkor a kép kimozdul központi helyzetéből.
Mint látjuk, a képeket sorbazárt elemként jelenítjük meg (**div.img img {display:inline;}**), de ennek nincs különösebb jelentősége. A kép-szegély 1px, fehér; azaz az alapértelmezetten fehér háttéren nem látszik. Ehhez képest az egér fölévitelére (**div.img a:hover img**) a szegélyszín kékre vált.
Végül, a kép-aláírást a **div.desc** kijelölés alatt formázzuk.

A kép-szegély színének megváltozása kiemeli a kép link-szerepét. Jól ismert eljárás, hogy az egérrel érintett vagy aktív linkek körül pontozott (fekete) szegély jelenik meg, miközben a kép ill. keret mérete nem változik. Ezt itt is elérhetjük, egy igen egyszerű módosítással:
```
div.img img
{
display:inline;
  margin: 3px;
  border: 1px solid #ffffff;
}
div.img a:hover img {border: 1px dotted black;}
```

A keret méretét növelve, és az inaktív állapotához észlelhetőbb színt rendelve is érdekes hatást érhetünk el:
```
div.img img
{
display:inline;
  margin: 3px;
  border: 5px solid red;
}
div.img a:hover img {border: 5px solid blue;}
```
A képek itt 5px-es, piros kerettel jelennek meg, mely az egér érintésére kékre vált.

Hogyha a kék keret méretét kisebbre választjuk:
```
div.img img
{
display:inline;
  margin: 3px;
  border: 5px solid red;
}
div.img a:hover img {border: 2px solid blue;}
```
akkor nemcsak a képkeret, hanem a külső keret is kisebbre esik össze, hiszen ktív állpotban a kép-doboz mérete lecsökken.

A külső keret ugrándozását kiküszöbölhetjük, ha az aktív állapotban keskenyebb keret hatását nagyobb margóval ellensúlyozzuk:
```
div.img img
{
display:inline;
  margin: 3px;
  border: 5px solid red;
}
```

```
div.img a:hover img
{
border: 2px solid blue;
margin:6px;
}
```
Itt tehát az 5px-ről 2px-re keskenyedő keret hatását az eredetinél 5px-2px = 3px-lel nagyobb, azaz 3px+3px = 6px-es, ktív állaothoz rendelt kerettel ellensúlyoztuk. Ekkor az egér érintésére ugyan átszíneződik és szűkül a kép kerete, de a külső keret nem változik.

CSS – képgaléria:

Egyszerű képgaléria balra igazított, változó keret-színű link-képekkel és képaláírásokkal:

```
div.img
        {
         margin: 2px;
         border: 1px solid #0000ff;
         height: auto;
         width: auto;
         float: left;
         text-align: center;
        }
div.img img
        {
         display: inline;
         margin: 3px;
         border: 1px solid #ffffff;
        }
div.img a:hover img {border: 1px solid #0000ff;}
div.desc
        {
         text-align: center;
         font-weight: normal;
         width: 110px;
         margin: 2px;
        }

<div class="img">
 <a target="_blank" href="URL">
<img src="URL" alt="megnevezés" width="100" height="100" />
</a>
 <div class="desc">Képaláírás</div>
</div>
```

XI. CSS kép-átlátszóság

Néhány gyors példa

A CSS segítségével könnyen készíthetünk átlátszó/áttetsző képeket is, amint azt az alábbi példák mutatják.

Első példánkban az ún. „mouseover effect”-tel (=egérérintési hatással) ismerkedünk meg, a homályos képek beállítása mellett:

```
<html>
<head>
<style type="text/css">
img
{
opacity:0.4;
filter:alpha(opacity=40);
}
</style>
</head>

<body>
<h1>Image Transparency</h1>

<img src="klematis.jpg" width="150" height="113" alt="klematis"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />

<img src="klematis2.jpg" width="150" height="113" alt="klematis"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />
</body>
</html>
```

Első példánk eredménye két kép, melyek inaktív, azaz egértől érintetlen álapotban homályosak, majd az egér érintésére kitisztulnak, teljes színerejűvé válnak.

Hogyha a belső stílus-oldal tartalmát átírjuk, pl. a következőképpen:

```
img
{
opacity:0.7;
filter:alpha(opacity=90);
}
```

akkor a képek eredetileg majdnem teljes fényerővel látszanak, majd az egér első érintésére teljesen kivilágosodnak. Ha ezután az egeret levisszük róluk, még sötétebbek lesznek, mint eredetileg voltak, és csak akkor világosodnak ki teljesen, ha az egérrel újra föléjük megyünk.

Második példánkban egy háttérképpel borított mezőben homályosított háttér-hatású szövegmezőt helyezünk el:

```
<html>
<head>
<style type="text/css">
div.background
{
  width: 500px;
  height: 250px;
  background: url(klematis.jpg) repeat;
  border: 2px solid black;
}
div.transbox
```

```
{
  width: 400px;
  height: 180px;
  margin: 30px 50px;
  background-color: #ffffff;
  border: 1px solid black;
  filter:alpha(opacity=60);
  opacity:0.6;
}
div.transbox p
{
  margin: 30px 40px;
  font-weight: bold;
  color: #000000;
}
</style>
</head>

<body>
<div class="background">

<div class="transbox">
<p>This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>

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

A **div.background** kijelölést méretezzük, feketén keretezzük és ismétlődő képes háttérrel látjuk el.
Az ebben lévő **transbox **-et (szövegmezőt) ugyancsak fekete kerettel látuk el, és megfelelő margókkal állítjuk a külső mező közepére. A fehér hátteret áttetszővé tesszük a **{filter:alpha(opacity=60); opacity:0.6;}** meghatározásokkal, így a fehérségen áttűnik a szövegmező alatti ** ** háttere.
A **transbox**-ban lévő bekezdést is margókkal igazítjuk középre.

Megjegyezzük, hogy a kép-átlátszóság egyelőre nem része a CSS-standardoknak; ám a legtöbb modern böngészőben már használható, és a W3C CSS3 előterjesztésében is szerepel.

Első példa: áttetsző kép beállítása
A fenti első példa kapcsán először is vizsgáljuk meg, hogy lehet a képeket a CSS-sel áttetszővé tenni.

Lássunk egy (normális) képet:

Az előbbi kép némi átlátszóság beállítása után így néz ki:

A kép forráskódja a következő:
```
<img src="klematis.jpg" width="150" height="113" alt="klematis"
style="opacity:0.4;filter:alpha(opacity=40)" />
```

Az átlátszóság megadására a Firefox az **opacity:*x***, az Internet Explorernél pedig a **filter:alpha(opacity=*x*)** jellemző szolgál. A CSS3 mondattan szerint az átlátszóság jellemzője az **opacity:*x***.

A Firefoxban az ***x*** értéke 0.0 és 1.0 között változhat. Minél kisebb az **opacity** (=átlátszatlanság) érték, annál átlátszóbb az elem.

Az Internet Explorerben az ***x*** értéke 0 és 100 között változhat. Minél kisebb tehát az **opacity** érték, annál átlátszóbb az elem.

Mint látjuk, a fenti HTML-kódban mind a CSS3 (és a Firefox), mind az IE előírásai szerint megadtuk a kép átlátszatlanságát, a **style** attribútummal. Az **opacity:0.4;** és a **filter:alpha(opacity=40);** meghatározás ugyanazt jelenti.

Első példa: egérérintési hatás (mouseover effect)
Tekintsük az alábbi példa-kódot:
```
<img src="klematis.jpg" style="opacity:0.4;filter:alpha(opacity=40)"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />

<img src="klematis2.jpg" style="opacity:0.4;filter:alpha(opacity=40)"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />
```
Ez a kód két képet hoz létre a HTML-oldal **<body>** részében. A képek 0.4-es átlátszatlansággal jelennek meg, azaz fényerejük a normális felénél is kisebb. Hogyha az egérrel a képhez nyúlunk, az átlátszatlanság hirtelen 1-re nő, azaz a fényerő a normális, teljes értékre emelkedik. Hogyha az egérel kimegyünk a képből, a fényerő (átlátszatlanság) ismét visszacsökken 0.4-esre.

Mint látjuk, az **onmouseover** és **onmouseout** attribútumok hozzáadásával meghatározhatjuk az elem stílusát, az egér megfelelő akciói esetére.

Az **onmouseover** attribútumot itt az átlátatlanság normálissá (egységnyivé) emelésére használjuk. Ekkor a hozzá tartozó meghatározás mondattana:
Firefoxban: **this.style.opacity=1**;
Internet Explorerben pedig: **this.filters.alpha.opacity=100**.

A kép egér általi elhagyásához rendelt átlátszóság-növekedést az **onmouseout** attribútummal állíthatjuk be, mely meghatározásának mondattana az előbbivel azonos.

Veyük most elő ismét legelső példánkat, melynek belső stílus-oldalát átírtuk:
```
<html>
<head>
<style type="text/css">
img
```

```
{
opacity:0.7;
filter:alpha(opacity=90);
}
</style>
</head>

<body>
<h1>Image Transparency</h1>

<img src="klematis.jpg" width="150" height="113" alt="klematis"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />

<img src="klematis2.jpg" width="150" height="113" alt="klematis"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />
</body>
</html>
```

Internet Explorerrel nézve ez azt jelenti, hogy kép eredeti átlátszatlansága 90; mely az egér érintésére 100-ra nő, majd annak távozására 40-re csökken.
Firefoxszal nézve majdnem ugyanez a helyzet, csak ott a kép kezdeti átlátszatlansága 0.7; és ez nő 1-re, majd csökken 0.4-re.
Mindez megegyezik a tapasztalatainkkal.

Természetsen hogyha valamelyik **<img>** tagbe külön **style**-ként a 90-től ill. 0.7-től eltérő kezdeti átlátszatlanságot határoznánk meg, az exempt stylusként felülírná a belső CSS-t. Így az adott kép más átltszósággal indul az oldal megjelenítésekor, mint a többi, melyekre a belső CSS **img** kijelölésének meghatározásai minden további nélkül érvényesek.

Második példa: áttetsző szövegdoboz beállítása
Hogyha újra megnézzük a fenti, második példa kódját:

```
<html>
<head>
<style type="text/css">
div.background
{
  width: 500px;
  height: 250px;
  background: url(klematis.jpg) repeat;
  border: 2px solid black;
}
div.transbox
{
  width: 400px;
  height: 180px;
  margin: 30px 50px;
  background-color: #ffffff;
  border: 1px solid black;
  filter:alpha(opacity=60);
  opacity:0.6;
}
div.transbox p
{
  margin: 30px 40px;
```

```
  font-weight: bold;
  color: #000000;
}
</style>
</head>

<body>
<div class="background">

<div class="transbox">
<p>This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>

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

Az előbb megismert átlátszatlanság-mondattan ismeretében már könnyebben értelmezhetjük; a **background** osztályú ** **-et fix méretezéssel, ismétlődő háttérképpel és kerettel látjuk el (**div.background** kijelölés).
Ezután a **transbox** osztályú ** **-et is fixen méretezzük, fehér hátteret állítunk be neki, melyet áttetszővé teszünk (**div.transbox** kijelölés).
Végül már csak a ** **-ba kerülő bekezdés szövegének formázása és pozícionálása van hátra (**div.transbox p** kijelölés).

CSS kép-átlátszóság:

Az opacity (=átlátszatlanság) CSS jellemző:

Internet Explorerben: {filter:alpha(opacity=*100*);}
Firefoxban és egyéb böngészőkben: {opacity:*1.00*;}

Az opacity jellemző JavaScript-ként:

Internet Explorerben: "this.filters.alpha.opacity=*100*;"
Firefoxban és egyéb böngészőkben: "this.style.opacity=*1.00*;"

Homályosított kép (helyi formázással):

<img src="*URL*" width="*100*" height="*100*" alt="*megnevezés*"
style="opacity:*1.00*;filter:alpha(opacity=*100*)" />

Egérérintésre homályosodó kép (helyi JavaScript-formázással):

<img src="*URL*" style="opacity:*0.50*;filter:alpha(opacity=*50*)"
onmouseover="this.style.opacity=*1.00*;this.filters.alpha.opacity=*100*"
onmouseout="this.style.opacity=*0.50*;this.filters.alpha.opacity=*50*" />

A kép-átlátszóság egyelőre nem része a CSS-standardoknak; ám a legtöbb modern böngészőben már használható, és a W3C CSS3 előterjesztésében is szerepel.

Az átlátszóbb elemekbe ágyazott összes további elem átlátszóbban fog megjelenni, amit nem lehet kompenzálni azok átlátszatlanságának megnövelésével, mert annak értéke legfeljebb 100 ill. 1.00 lehet, ami megfelel a kezdeti, már átlátszóbb megjelenésnek!
Ezért átlátszó elemekre átlátszatlanabbakat ne statikus pozícionálással (egymásba ágyazással) helyezzünk; hanem az összetartozó elemeket egy ** ** elemen belül egy szintre írjuk be, és a

{position=*fixed/absolute/relative*;} valamint a **{z-index:-*1/auto/inherit*;}** jellemzők segítségével pozícionáljuk őket!

XII. CSS kép-szellemek

Kép-szellemek

A szellem (=sprite) kifejezés az informatikában **mozgatható grafikus elemet** jelent.

A kép-szellem több kép egyetlen objektummás (nagyobb képpé) való összefoglalása. Egy sok képet tartalmazó weboldal letöltése hosszú folyamat, mivel ez egyes képfájlokat külön-külön kell a böngészőnek a szervertől lekérnie. A kép-szellemek használatával csökkenthetjük a szerver felé indított lekérések számát, amivel sávszélességet takarítunk meg.

Egy egyszerű példa

Egy weboldal linkként használt három külön képe helyett egyetlen (**img_navsprites.gif**) kép-szellemet használhatunk, melynek kinézete a következő:

A fenti, egyetlen képfájl három, jól elkülöníthető térrészből áll. A CSS segítségével azonban elérhetjük, hoy a képnek csak azon része jelenjen meg, amelyik szükséges.

A következő CSS-kód segítségével beállítjuk, hogy az **img_navsprites.gif** kép mely része jelenjen meg:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<style type="text/css">
img.home
{
width:46px;
height:44px;
background:url(img_navsprites.gif) 0 0;
}
img.next
{
width:43px;
height:44px;
background:url(img_navsprites.gif) -91px 0;
}
</style>
</head>

<body>
<img class="home" src="img_trans.gif" width="1" height="1" />
<br /><br />
<img class="next" src="img_trans.gif" width="1" height="1" />
</body>
</html>
```

A szövegtestben két **<img>** taget látunk; ezek jelölik ki a beillesztendő kép-szellemek helyét a dokumentumban. Az **src** attribútumot kötelező megadn, ezért egy átlátszó képre hivatkozunk, melyet igen kis (**"0"** vagy **"1"**) szélességgel-magassággal állítunk be.
Az így kijelölt két pici és átlátszó kép mögé háttérként egy-egy kép-szellemet állítunk be, a belső CSS-ben. A két (**img.home** ill. **img.next**) kijelölésben először megadjuk a beállítani kívánt háttér méretét (pl. **{width:46px;height:44px;}**). Ezzel voltaképpen egy kis ablakot hoztunk létre az átlátszó kép mögött (ami, mivel nincs kerete, nem látszik. Keretet pl. a **border:1px solid black;** meghatározással készíthetünk.).
A **background:url(img_navsprites.gif) -91px 0;** meghatározással az imént létrehozott ablakba, háttér-

képnek a három kis kép egyesítésével létrehozott **img_navsprites.gif**-et állítjuk be.
Az URL után írt két érték a háttérkép pozícionálására utal; tulajdonképpen az adott háttérkép végtelenített hálzatát mozgatjuk velük a kis ablak mögött.
Az első érték megadja, hogy az ablak (azaz a keret) bal szélével a képfájl (saját bal szélétől számított) hányadik függőleges pixelsora essen egybe. (Vagy másképpen: hogy a keret jobb szélével a kép jobb szélétől számított hányadik függőleges pixelsora essen egybe.)
A második érték pedig azt adja meg, hogy a keret felső élével a kép tetejétől felfelé számított hányadik vízszintes pixelsor essen egybe. (Vagy másképpen, hogy a keret alsó élével a kép aljától felfelé számított hányadik vízszintes pixelsor essen egybe.)
A méret-értékek közül a 0 után nem kell mértékegységet írnunk (egyébként kötelező).

Ez a kép-szellemek használatának legegyszerűbb módja; amit a következőkben linkekkel és aktivitásfüggő formázással (**hover**) egészítjünk ki.

Kép-szellemek: navigációs lista készítése
A következő példában az **img_navsprites.gif** képet egy navigációs lista készítéséhez használjuk fel.

Ehhez a **<body>** részben HTML lista-szerkezetet használunk, mivel a lista-elemek tartalmazhatnak linkeket, és háttér-képpel is rendelkezhetnek.

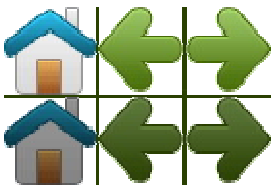
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<style type="text/css">
#navlist{position:relative;}
#navlist li{margin:0;padding:0;list-
style:none;position:absolute;top:0;}
#navlist li, #navlist a{height:44px;display:block;}

#home{left:0px;width:46px;}
#home{background:url('img_navsprites.gif') 0 0;}

#prev{left:63px;width:43px;}
#prev{background:url('img_navsprites.gif') -47px 0;}

#next{left:129px;width:43px;}
#next{background:url('img_navsprites.gif') -91px 0;}
</style>
</head>

<body>
<ul id="navlist">
  <li id="home"><a href="default.asp"></a></li>
  <li id="prev"><a href="css_intro.asp"></a></li>
  <li id="next"><a href="css_syntax.asp"></a></li>
</ul>
</body>
</html>
```

A példa magyarázata:

- **#navslist** kijelölés:
  - A **position** jellemzőt **relative**-nak választjuk, ami lehetővé teszi a **navslist** jelű elemen belüli abszolút pozícionálást.
  - A **navslist**-be tartozó **<li>** és **<a>** elemek magasságát egyaránt 44px-nek vesszük.
  - A **<li>**-elemek margóját és szegélyét 0-nak választjuk, a listajeleket eltávolítjuk, és a listaelemeket abszolút módon pozícionáljuk, az alábbiak szerint:

- **#home** kijelölés:
  - Az elem közvetlenül az őt tartalmazó **navlist** bal szélére kerül; szélessége 46px.
  - Az általános (**#navlist li, #navlist a** kijelölésre vonatkozó) magasság és az előbb megadott szélesség által kijelölt keretbe beállítjuk az **img_navsprites.gif** háttérképet, 0 0 elhelyezéssel.
- **#prev** kijelölés:
  - A második lista-elem a bal széltől már 63px abszolút poícióba kerül (azaz a **home** 46px-es szélességénél valamivel távolabbra, hogy az elemek közt hely legyen), szélessége 43px.
  - Ugyancsak beállítjuk a megfelelően pozícionált **img_navsprites.gif** háttérképet (a vízszintes pozíció-érték -47px, ami megfelel a **home** kép-rész 46px-es szélességének, és az 1px es elválasztóvonalnak).
- **#next** kijelölés:
  - A harmadik lista-elemet a balszélről már 129px-re, jobbra helyezzük el (abszolút pozícionálással; figyelembe véve, hogy a második kép 63px-nél kezdődik és 43px széles; és némi helyet is hagyva).
  - Itt is beálltjuk a **img_navsprites.gif** háttérképet, és azt megfelelően pozícionáljuk (a vízszintes pozíció -91px, ami megfelel 46+43px-es, korábbi képrész-szélességeknek, és a 2db. 1px-es osztóvonalnak.)

Kép-szellemek: egérérintési hatások
A következő példában az előbbi navigációs listát egérérintési elemmel bővítjük. Ehhez egy másik képre (**img_navsprites_hover.gif**) van szükségünk:



Megfelelő CSS-beállítással elérhetjük, hogy a kép függőlegesen elmozduljon a linkek mögött beállított háttérkép-kerethez képest; aszerint, hogy a link aktív állapotban van-e.
Mivel így a kép csak elmozdul, nem kell helyette újat letölteni, azaz nincs időbeli késés a link egér-akcióra való reakciójában.

Célunk eléréséhez csupán három sort kell hozzáadnunk a fenti kódhoz.

```
#home a:hover{background: url('img_navsprites_hover.gif') 0 -45px;}
#prev a:hover{background: url('img_navsprites_hover.gif') -47px -45px;}
#next a:hover{background: url('img_navsprites_hover.gif') -91px -45px;}
```

illetve az összes URL-t **img_navsprites_hover.gif**-re átállítani, azaz:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<style type="text/css">
#navlist{position:relative;}
#navlist li{margin:0;padding:0;list-
style:none;position:absolute;top:0;}
#navlist li, #navlist a{height:44px;display:block;}

#home{left:0px;width:46px;}
#home{background:url('img_navsprites_hover.gif') 0 0;}
#home a:hover{background: url('img_navsprites_hover.gif') 0 -45px;}

#prev{left:63px;width:43px;}
#prev{background:url('img_navsprites_hover.gif') -47px 0;}
```

```
#prev a:hover{background: url('img_navsprites_hover.gif') -47px -45px;}

#next{left:129px;width:43px;}
#next{background:url('img_navsprites_hover.gif') -91px 0;}
#next a:hover{background: url('img_navsprites_hover.gif') -91px -45px;}
</style>
</head>

<body>
<ul id="navlist">
  <li id="home"><a href="default.asp"></a></li>
  <li id="prev"><a href="css_intro.asp"></a></li>
  <li id="next"><a href="css_syntax.asp"></a></li>
</ul>
</body>
</html>
```

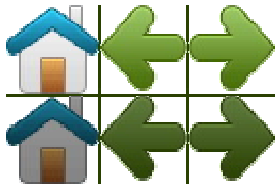
Tehát mivel a lista-elemek linket tartalmaznak, használhatjuk az **a:hover** ál-osztályt. Ezen ál-osztály kijelölése mellett az összes képnek ugyanazt a vízszintes pozíciót állítjuk be, csak épp 45px-lel lejjebb.

Az eredmény: a kép bármelyik link megérintésekor 45px-lel lejjebb csúszik, és az így megjelenő, sötétebb színű rész-kép pontosan elfedi az egyébként fölötte lévőt. Az eredeti kép azonban továbbra is a helyén marad; ti. ha az egyik link lecsúszását 45px helyett csak 25-nek vesszük, akkor az eredeti kép is kilátszik a lejjebb csúszott, új háttérkép mögül, mely azt elfedi.

CSS kép-szellemek:

A kép-szellemek előnye, hogy egyetlen képfájl betöltésével számos képet jeleníthetünk meg (kevés számú lekérdezés, nincs holtidő pl. a:hover esetén megjelenítendő kép letöltésekor).

Egy kép-szellem összképe:



Kép-szellem beállítása (a háttér-kép pozícionálásával); általános alak:

background:url(*kép-szellem-URL*) *-100px 100px*;

Kép beszúrása kép-szellemből:

```
img.home
{
width:100px;
height:100px;
background:url(kép-szellem-URL) 0 0;
}
img.next
{
width:100px;
height:100px;
background:url(kép-szellem-URL) -100px 0;
}

<img class="home" src="átlátszó_kis_kép_URL-je" width="1" height="1" />
<img class="next" src="átlátszó_kis_kép_URL-je " width="0" height="0" />
```

Statikus háttérkép kép-szellemből:

```
#navlist                {position:relative;}
#navlist li             {margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a {height:44px;display:block;}

#home {left:0px;width:46px;}
#home {background:url(kép-szellem-URL) 0 0;}

#prev  {left:63px;width:43px;}
#prev  {background:url(kép-szellem-URL) -47px 0;}

#next  {left:129px;width:43px;}
#next  {background:url(kép-szellem-URL) -91px 0;}

<ul id="navlist">
        <li id="home"><a href="URL1"></a></li>
        <li id="prev"><a href="URL2"></a></li>
        <li id="next"><a href="URL3"></a></li>
</ul>
```

Egérérintésre változó háttérkép kép-szellemből:

```
#navlist                {position:relative;}
#navlist li             {margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a {height:44px;display:block;}

#home                   {left:0px;width:46px;}
#home                   {background:url(kép-szellem-URL) 0 0;}
#home a:hover           {background: url(kép-szellem-URL) 0 -45px;}

#prev                   {left:63px;width:43px;}
#prev                   {background:url(kép-szellem-URL) -47px 0;}
#prev a:hover           {background: url(kép-szellem-URL) -47px -45px;}

#next                   {left:129px;width:43px;}
#next                   {background:url(kép-szellem-URL) -91px 0;}
#next a:hover           {background: url(kép-szellem-URL) -91px -45px;}

<ul id="navlist">
        <li id="home"><a href="URL1"></a></li>
        <li id="prev"><a href="URL2"></a></li>
        <li id="next"><a href="URL3"></a></li>
</ul>
```

A kép-szellemek alkalmazásához is célszerű DOCTYPE-ot beállítani, pl.:

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

XIII. CSS – média-típusok

A CSS-sel azt is beállíthatjuk, hogy a dokumentum hogy jelenjen meg a különféle média (=adathordozó) típusokban (pl. képernyő, kinyomtatott oldalak, hallgatható tartalom stb.). Az eltérő típusokhoz eltérő megjelenést társíthatunk.

Média-típusok használata

Egyes CSS-jellemzők csak bizonyos médiatípusokra vonatkoznak. Például a **"voice-family"** jellemzőt a hallgatható tartalmak formázására használjuk.
Más jellemzők többféle média-típusban is megjeleníthetők, így pl. a **"font-size"** jellemző mind a képerőn, mind a nyomtatásban történő megjelenésben szerepet játszik; de elkpzelhető, hogy az esettől függően más-más értékkel. A dokumentumokat ugyanis a képernyőn általában nagyobb betűmérettel szükséges megjeleníteni, mint papiroson; s ugyanígy a sans-serif betűk képernyőn olvashatók jobban, míg a serif betűk papíron.

Az @media – szabály

Az @media – szabály segítségével a különböző média-típusok számára eltérő formázást írhatunk elő, egyetlen CSS-en belül.

Az alábbi HTML-oldal **"test"** osztályú bekezdései pl. a képernyőn 14px-es Verdana betűtípussal jelennek meg, án ha az oldalt kinyomtatjuk, akkor a papíron 10px-es Times betűkkel szerepelnek. Azonban, mint látjuk, mindkét esetben félkövérek lesznek a betűk:

```
<html>
<head>
<style>
@media screen {p.test {font-family:verdana,sans-serif;font-size:14px;}}
@media print {p.test {font-family:times,serif;font-size:10px;}}
@media screen,print {p.test {font-weight:bold;}}
</style>
</head>

<body>
...
</body>
</html>
```

A média-típus beállításának mondattana tehát egyszerű: a **@media** szó után, vesszőkkel elválasztva kiírjuk, mely média-típusokra vonatkozik a formázási utasítás. (Utóbbit pedig kapcsos zárójelbe tesszük.)

Média-típusok

Az alábbi táblázatban összefoglaltuk a különböző médiatípusok CSS-elnevezését. Ezek nem esetfüggők, azaz elvileg kis- és nagybetűvel egyaránt írhatók, de a W3C ajánlásai értelmében kisbetűkkel írandók.

Média-típus (név)	Leírás
all	Minden média-típusra ill. megjelenítési eszközre vonatkozik.
aural	Beszéd- és hang-szintetizátorokra vonatkozik.
braille	Braille-rendszerű tapintó-kijelzőkre vonatkozik.
embossed	Braille-irás – nyomtatókra vonatkozik.
handheld	Kis kézi számítógépekre (pl. palmtop) vonatkozik.
print	Nyomtatókra vonatkozik.
projection	Projektoros megjelenítésre vonatkozik.
screen	Monitoros megjelenítésre vonatkozik.
tty	Betűtáblázatos/szegmeneses megjelenítő eszközökre (pl. billentyűzetek, terminálok, folyadékkristályos kijelzők) vonatkozik.
tv	Televízió-készülékkel történő megjtekintésre vonatkozik.

CSS – média-típusok:

Média-típusok:

Összes	all
Beszéd	aural
Braille-monitor	braille
Braille-nyomtató	embossed
Palmtop/mobil	handheld
Nyomtató	print
Kivetítő	projection
Monitor	screen
Szegmens-tábla	tty
Televízió	tv

Média-típus jelölése belső vagy külső stílus-oldalban:

@media *screen*, *projection*, *print* {kijelölés {jellemző:érték;}}

Média-típus jelölése a <link> tagben:

<link rel=”stylesheet” type=”text/css” media=”*screen, projection, print*” href=”*URL*” />

XIV. CSS attribútum-kijelölések

HTML-elemek formázása attribútum-kijelöléssel

Nemcsak az **id** és **class** attribútumokkal, hanem minden dokumentumtestbe írt attribútummal kijelölhetünk egy HTML elem-csoportot a CSS-formázás számára.

FONTOS: Az Internet Explorer 7 (és magasabb) verziója csak **DOCTYPE** megadása mellett támogatja az attribútum-kijelölést, az alacsonyabb sorszámú verziók pedig egyáltalán nem!

Attribútum-kijelölés

Az alábbi dokumentum minden **title** attribútummal rendelkező eleme zöld (betű)színt kap:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
[title]{color:green;}
</style>
</head>

<body>
<h2>Will apply to:</h2>
<h1 title="Hello world">Hello world</h1>
<a title="W3Schools" href="http://w3schools.com">W3Schools</a>

<hr />

<h2>Will not apply to:</h2>
<p>Hello!</p>
</body>
</html>
```

Itt a „Hello world” tartalmú **<h1>** elem és a link rendelkezik zöld betűszínnel.

Attribútum és érték – kijelölés

Az alábbi példában csakis a **”W3Schools”** értékkel rendelkező **title** attribútumot tartalmazó elemeket jelöljük ki formázásra:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
[title=W3Schools]{border:5px solid green;}
</style>
</head>

<body>
<h2>Will apply to:</h2>
<img title="W3Schools" src="w3schools_logo.gif" width="270" height="50"
/>
<br />
<br />
<a title="W3Schools" href="http://w3schools.com">W3Schools</a>

<hr />

<h2>Will not apply to:</h2>
```

```
<p title="greeting">Hi!</p>
<a class="W3Schools" href="http://w3schools.com">W3Schools</a>
</body>
</html>
```
Itt csak a **title=”W3Schools”** meghatározást tartalmazó HTML-elemek rendelkeznek zöld szegéllyel, azaz az **<img>** és **<a>** elem.

Attribútum-kijelölés többszörös értékeknél
Az alábbi példában szereplő attribútumérték-kijelölés szintén mindazon elemek formázására szolgál, melyeknél az adott attribútum a megadott értékkel bír; de akkor is működik, hogyha az attribútum több, szóközzel elválasztott értéke közt a kijelölt megtalálható:
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
[title~=hello] {color:blue;}
</style>
</head>

<body>
<h2>Will apply to:</h2>
<h1 title="hello world">Hello world</h1>
<p title="student hello">Hello CSS students!</h1>

<hr />

<h2>Will not apply to:</h2>
<p title="student">Hi CSS students!</p>
</body>
</html>
```
Itt az oldal felső térfelén lévő **<h1>** és **<p>** elem **title** attribútuma egyaránt tartalmazza a **hello** értéket, amelyet a fejrészben a **title** lehetséges értékei közül kijelöltünk; így ezek az elemek kék színnel jelennek meg. Ez a módszer tehát lehetővé teszi, hogy a formázandó elemeket olyan attribútum alapján jelöljük ki, mely a megadott mellett más értékeket is tartalmazhat.

A következő példában olyan (**lang**) attribútum – érték párosítást jelölünk ki, melynek értéke után továbbiak is szerepelhetnek, kötőjellel elválasztva.

FIGYELEM! Itt már szükséges, hogy a kijelölésre kerülő érték a felsoroltak (és kötőjellel elválasztottak) közül az első legyen!

Ehhez az előbbi kijelölési módszerben a szóközt jelképező hullámvonal (~) helyett állóvonalat (|)teszünk az egyenlőségjel elé:
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
[lang|=en] {color:blue;}
</style>
</head>

<body>
<h2>Will apply to:</h2>
```

```
<p lang="en">Hello!</p>
<p lang="en-us">Hi!</p>
<p lang="en-gb">Ello!</p>

<hr />

<h2>Will not apply to:</h2>
<p lang="us">Hi!</p>
<p lang="no">Hei!</p>
</body>
</html>
```

Itt az összes olyan **lang** attribútumot kijelöljük, melynek értékei közt az **en** megtalálható; azonban, visszatérve az előbbi figyelmeztetésünkre, fontos, hogy a **en** legyen az attribútum elő értéke.

Tehát hogyha a harmadik bekezdés

```
<p lang=”en-gb”>Ello!</p>
```

helyett

```
<p lang=”gb-en”>Ello!</p>
```

akkor a kék formázás nem lez rá érvényes. Akkor is érvénytelen a formázás, hogyha az érték után nem kötőjel áll, pl. a

```
<p lang=”en en”>Ello!</p>
```

Bekezdés sem lesz kék, holott az **en** kétszer is szerepel az értékek között, az első helyen is.

Űrlapok formázása

Az attribútum-kijelölések különösen hasznosak, hogyha **class** ill. **id** megadása nélkül szeretnénk űrlapokat formázni:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style>
input[type="text"]
{
width:150px;
display:block;
margin-bottom:10px;
background-color:yellow;
}
input[type="button"]
{
width:120px;
margin-left:35px;
display:block;
}
</style>
</head>

<body>
<form name="input" action="" method="get">
Firstname:<input type="text" name="Name" value="Peter" size="20">
Lastname:<input type="text" name="Name" value="Griffin" size="20">
<input type="button" value="Example Button">
</form>
</body>
</html>
```

Amint látjuk, az attribútum-kijelöléseket itt is ugyanúgy írjuk a CSS-kódba, mint az egyedi ill. csoportos kijelöléseket, ti. a tag neve után (ha van), különleges jelöléssel (ami itt a pont ill. kettőskereszt helyett a szögletes zárójal).
A példában szereplő belső stílus-oldal értelmében azok az **input** elemek, melyek **type=”text”** meghatározással bírnak (azaz a szövegbeviteli mezők), 150px szélesek, tömbszerű megjelenéűek, 10px alsó margóval és sárga háttérszínnel.

A **button** típus-értékkel rendelkelkező beviteli (**input**) elemek pedig 120px szélesek, tömbszerűek és 35px bal margóval rendelkeznek.
Az oldal ennek megfelelően két, címmel és alapértelmezett értékkel rendelkező, sárga hátterű szövegdobozból és egy azoknál keskenyebb, „Example Button” feliratú nyomógombból áll.

CSS attribútum-kijelölések:

Attribútum-kijelölés: [attribútum] {jellemző:érték;}

Elem-osztályon belüli attribútum-kijelölés: elem.osztály[attribútum] {jellemző:érték;}

Elemen belüli attribútum-kijelölés: elem[attribútum] {jellemző:érték;}

Attribútum és érték együttes kijelölése:

Egyetlen érték kijelölése:

[attribútum=érték] {jellemző:érték;}

<elem attribútum=”érték” />

Bármelyik kijelölése több, szóközzel elválasztott érték közül:

[attribútum~=érték2] {jellemző:érték;}

<elem attribútum=”érték1 érték2 érték3” />

Az első kijelölése több, kötőjellel elválasztott érték közül:

[attribútum|=érték1] {jellemző:érték;}

<elem attribútum=”érték1-érték2-érték3” />

Űrlap formázása attribútum-kijelölésekkel:

```
input[type="text"] {
width:100px;
display:block;
margin-bottom:100px;
background-color:yellow;
}
input[type="button"] {
width:100px;
margin-left:100px;
display:block;
}

<form name="input" method="get">
Firstname:<input type="text" name="Name" value="Peter" size="100">
Lastname:<input type="text" name="Name" value="Griffin" size="100">
<input type="button" value="Example Button">
</form>
```

Az érték-kijelöléskor az idézőjelek elhagyása vagy használata egyenértékű!

Az Internet Explorer 7 (és magasabb) verziója csak DOCTYPE megadása mellett támogatja az attribútum-kijelölést, az alacsonyabb sorszámú verziók pedig egyáltalán nem!
Így attribútum-kijelölésekkor feltétlenül adjuk meg, pl.:

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

XV. CSS – kerülendő eljárások

A következőkben bemutatunk néhány eljárást, melyek alkalmazása a CSS-ben kerülendő.

Internet Explorer „magatatások” (behaviors)

Az Internet Explorer 5. verziójában tűnt fel az ún. **behavior** (=magatartás) attribútum. Ezeket a CSS-be írva bizonyos „viselkedésre” bírhatjuk a formázott HTML-elemet.

A **behavior** jellemző használatától azonban tartózkodnunk kell, mert *csak* az Internet Explorer támogatja.

Helyette használjunk inkább JavaScriptet vagy (X)HTML DOM-ot!
A JavaScript egy könnyített programozási eljárás, mellyel a weboldalak működését interaktívabbá (felhasználóbarátabbá) és dinamikusabbá tehetjük. Az (X)HTML DOM egy programozási nyelvektől független dokumentumszerkesztési módszer. Az ennek megfelelően felépített dokumentumok elemeit megfelelő programozási eljárásokkal könnyen interaktívvá (pl. a felhasználó által átrendezhetővé) tehetjük.

Első példa: színváltás egér-érintésre

A következő példában a **<h1>** elemekhez rendelünk bizonyos viselkedést:

```
<html>
<head>
<style type="text/css">
h1 {behavior:url("behave.htc");}
</style>
</head>

<body>
<h1>Mouse over me!!!</h1>
</body>
</html>
```

Mint látjuk, a **behavior** jellemzőt a többi CSS-jellemzővel azonos mondattan szerint használjuk. A címsorra megadott stílust a **behave.htc** elérésű (URL-ű) XML-dokumentum tartalmazza, a következőképpen:

```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />

<script type="text/javascript">
function hig_lite()
{
element.style.color='red';
}

function low_lite()
{
element.style.color='blue';
}
</script>
```

A megjelenő oldal egyetlen, kék színű 1-es címsorból áll, melynek betűszíne az erér érintésére pirosra vált.

Második példa: gépírás-szimuláció

A következő HTML-oldal **typing** jelű **span** eleme gépírást utánzó módon jelenik meg, mintha valaki írógéppel (és **courier new** betűkkel) írná fel tartalmát a képernyőre:

```
<html>
<head>
<style type="text/css">
#typing
{
```

```
behavior:url(behave_typing.htc);
font-family:"courier new";
}
</style>
</head>

<body>
<span id="typing" speed="100">
<p>IE5 introduced DHTML behaviors. Behaviors are a way to add DHTML
functionality to HTML elements with the ease of CSS.<br />
<br />
How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p>
</span>
</body>
</html>
```

Mint látjuk, a belső stílus-oldal itt is egy külső XML-re hivatkozik a **behavior** jellemzőn keresztül:

```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />

<script type="text/javascript">
var i,text1,text2,textLength,t;

function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
{
clearInterval(t);
}
}
</script>
```

Az oldal tehát Courier New betűkkel, fokozatosan íródik ki a képernyőre, majd kiírva is marad, de az oldal újratöltésekor isméet elölről kezdődik a betűk feltűnése.

Amint a weboldal szövegében szerepelt, az Internet Explorer a CSS segítségével ad DHTML-elemeket a weboldalhoz. A DHTML nem más, mint a dinamikus weboldal-szerkesztési módszerek összefoglaló neve. A dinamikus HTML, azaz DHTML oldalak készítéséhez HTML, JavasCript, (X)HTML DOM és CSS elemeket és módszereket egyaránt felhaználunk.

Mint láttuk, a weboldal dinamikussá tételére az Internet Explorer **behavior** attribútuma a CSS-en keresztül

egy-egy XML oldalra hivatkozott. Az XML, vagyis „kiterjeszthető jelölőnyelv” (=eXtensible Markup Language) adatok továbbítására és tárolására szolgál. Pontos írási szabályai nincsenek, tagjei szabadon választhatók. Csupán adattárolásra és nem adatstruktúrálásra vagy -megjelenítésre szolgál. Ebben az esetben lehetőséget ad a bonyolult gépelési folyamatleírásnak a HTML-oldaltól elkülönített tárolására és a felhasználó számára kedvező, egyéni jelölések használatára.

CSS – kerülendő eljárások:

Internet Explorer „magatatások” (behaviors)

Az Internet Explorer 5. verziójában tűnt fel az ún. **behavior** (=magatartás) attribútum. Ezeket a CSS-be írva bizonyos „viselkedésre” bírhatjuk a formázott HTML-elemet. Használatától azonban tartózkodnunk kell, mert *csak* az Internet Explorer támogatja. Helyette használjunk inkább JavaScriptet vagy (X)HTML DOM-ot!

Példák a behavior jellemző alkalmazására:

I.: <h1> színváltása egér-érintésre

CSS:
```
h1 {behavior:url("behave.htc");}
```

HTML:
```
<h1>Mouse over me!!!</h1>
```

XML:
```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />

<script type="text/javascript">
function hig_lite()
{
element.style.color='red';
}

function low_lite()
{
element.style.color='blue';
}
</script>
```

II.: gépírás-szimuláció

CSS:
```
#typing
{
behavior:url(behave_typing.htc);
font-family:"courier new";
}
```

HTML:
```
<span id="typing" speed="100">
<p>IE5 introduced DHTML behaviors. Behaviors are a way to add DHTML functionality to HTML elements with the ease of CSS.</p>
</span>
```

XML:
```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />

<script type="text/javascript">
var i,text1,text2,textLength,t;

function beginTyping()
{
```

```
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
{
clearInterval(t);
}
}
</script>
```

XVI. CSS - összefoglalás

A következő táblázatokban [mely a http://www.w3schools.com/css/css_reference.asp webhelyen, a legfőbb böngészőkkel végzett tesztekkel összhangban, folyamatosan (legutóbb 2010.VII.2-án) frissül(t)] összefoglaltuk a legfontosabb CSS-kijelöléseket és jellemzőket.

CSS Selectors

In CSS, selectors are patterns used to select the element(s) you want to style.

The "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Selector	Example	Example Selects:	CSS
.class	.intro	All elements with class="intro"	1
#id	#firstname	The element with id="firstname"	1
*	*	All elements	2
element	p	All <p> elements	1
element,element	div,p	All elements and all <p> elements	1
element element	div p	All <p> elements inside elements	1
element>element	div>p	All <p> elements where the parent is a element	2
element+element	div+p	All <p> elements placed immediately after a element	2
[*attribute*]	[target]	All elements with a target attribute	2
[*attribute=value*]	[target=_blank]	All elements with a target attribute equal to "_blank"	2
[*attribute~=value*]	[title=flower]	All elements with a title attribute that contains space separated words, one of which is "flower"	2
[*attribute|=language*]	[lang|=en]	All elements where the lang attribute's value is "en", even if the value contains a hyphen (-), like "en-us"	2
:link	a:link	All links (<a> elements with href)	1
:visited	a:visited	All visited links	1
:active	a:active	Active links	1
:hover	a:hover	Links on mouse over	1
:focus	input:focus	The input element that has focus	2
:first-letter	p:first-letter	The first letter of all <p> elements	1
:first-line	p:first-line	The first line of all <p> elements	1
:first-child	p:first-child	All <p> elements that is the first child of its parent	2
:before	p:before	Content will be placed before each <p> element	2
:after	p:after	Content will be placed after each <p> element	2
:lang(*language*)	p:lang(it)	All <p> elements with the lang attribute containing "it"	2

CSS Properties

CSS Property Groups

- Background
- Border and outline
- Dimension
- Font
- Generated content
- List
- Margin
- Padding
- Positioning
- Print
- Table
- Text

The "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Background Properties

Property	Description	CSS
background	Sets all the background properties in one declaration	1
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	1
background-color	Sets the background color of an element	1
background-image	Sets the background image for an element	1
background-position	Sets the starting position of a background image	1
background-repeat	Sets how a background image will be repeated	1

Border and Outline Properties

Property	Description	CSS
border	Sets all the border properties in one declaration	1
border-bottom	Sets all the bottom border properties in one declaration	1
border-bottom-color	Sets the color of the bottom border	1
border-bottom-style	Sets the style of the bottom border	1
border-bottom-width	Sets the width of the bottom border	1
border-color	Sets the color of the four borders	1
border-left	Sets all the left border properties in one declaration	1
border-left-color	Sets the color of the left border	1
border-left-style	Sets the style of the left border	1
border-left-width	Sets the width of the left border	1
border-right	Sets all the right border properties in one declaration	1
border-right-color	Sets the color of the right border	1
border-right-style	Sets the style of the right border	1
border-right-width	Sets the width of the right border	1
border-style	Sets the style of the four borders	1
border-top	Sets all the top border properties in one declaration	1
border-top-color	Sets the color of the top border	1
border-top-style	Sets the style of the top border	1
border-top-width	Sets the width of the top border	1
border-width	Sets the width of the four borders	1
outline	Sets all the outline properties in one declaration	2
outline-color	Sets the color of an outline	2
outline-style	Sets the style of an outline	2
outline-width	Sets the width of an outline	2

Dimension Properties

Property	Description	CSS
height	Sets the height of an element	1
max-height	Sets the maximum height of an element	2
max-width	Sets the maximum width of an element	2
min-height	Sets the minimum height of an element	2
min-width	Sets the minimum width of an element	2
width	Sets the width of an element	1

Font Properties

Property	Description	CSS
font	Sets all the font properties in one declaration	1
font-family	Specifies the font family for text	1
font-size	Specifies the font size of text	1
font-style	Specifies the font style for text	1
font-variant	Specifies whether or not a text should be displayed in a small-caps font	1
font-weight	Specifies the weight of a font	1

Generated Content Properties

Property	Description	CSS
content	Used with the :before and :after pseudo-elements, to insert generated content	2
counter-increment	Increments one or more counters	2
counter-reset	Creates or resets one or more counters	2
quotes	Sets the type of quotation marks for embedded quotations	2

List Properties

Property	Description	CSS
list-style	Sets all the properties for a list in one declaration	1
list-style-image	Specifies an image as the list-item marker	1
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow	1
list-style-type	Specifies the type of list-item marker	1

Margin Properties

Property	Description	CSS
margin	Sets all the margin properties in one declaration	1
margin-bottom	Sets the bottom margin of an element	1
margin-left	Sets the left margin of an element	1
margin-right	Sets the right margin of an element	1
margin-top	Sets the top margin of an element	1

Padding Properties

Property	Description	CSS
padding	Sets all the padding properties in one declaration	1
padding-bottom	Sets the bottom padding of an element	1
padding-left	Sets the left padding of an element	1
padding-right	Sets the right padding of an element	1
padding-top	Sets the top padding of an element	1

Positioning Properties

Property	Description	CSS
bottom	Sets the bottom margin edge for a positioned box	2
clear	Specifies which sides of an element where other floating elements are not allowed	1
clip	Clips an absolutely positioned element	2
cursor	Specifies the type of cursor to be displayed	2

display	Specifies the type of box an element should generate	1
float	Specifies whether or not a box should float	1
left	Sets the left margin edge for a positioned box	2
overflow	Specifies what happens if content overflows an element's box	2
position	Specifies the type of positioning for an element	2
right	Sets the right margin edge for a positioned box	2
top	Sets the top margin edge for a positioned box	2
visibility	Specifies whether or not an element is visible	2
z-index	Sets the stack order of an element	2

Print Properties

Property	Description	CSS
orphans	Sets the minimum number of lines that must be left at the bottom of a page when a page break occurs inside an element	2
page-break-after	Sets the page-breaking behavior after an element	2
page-break-before	Sets the page-breaking behavior before an element	2
page-break-inside	Sets the page-breaking behavior inside an element	2
widows	Sets the minimum number of lines that must be left at the top of a page when a page break occurs inside an element	2

Table Properties

Property	Description	CSS
border-collapse	Specifies whether or not table borders should be collapsed	2
border-spacing	Specifies the distance between the borders of adjacent cells	2
caption-side	Specifies the placement of a table caption	2
empty-cells	Specifies whether or not to display borders and background on empty cells in a table	2
table-layout	Sets the layout algorithm to be used for a table	2

Text Properties

Property	Description	CSS
color	Sets the color of text	1
direction	Specifies the text direction/writing direction	2
letter-spacing	Increases or decreases the space between characters in a text	1
line-height	Sets the line height	1
text-align	Specifies the horizontal alignment of text	1
text-decoration	Specifies the decoration added to text	1
text-indent	Specifies the indentation of the first line in a text-block	1
text-shadow	Specifies the shadow effect added to text	2
text-transform	Controls the capitalization of text	1
unicode-bidi		2
vertical-align	Sets the vertical alignment of an element	1
white-space	Specifies how white-space inside an element is handled	1
word-spacing	Increases or decreases the space between words in a text	1

CSS-ÖSSZEFOGLALÁS

IV. KÖNYV: HALADÓ SZINT

I. Kijelölések csoportosítása és egymásba ágyazása

Kijelölés-típusok:

Elem: *elem*
Név: *#név*
Csoport: *.csoport*
Adott nevű elem(ek): *elem#név*
Adott csoportba tartozó elem(ek): *elem.csoport*

Többszörös kijelölés: *elem1*, *elem2*, *elem3* {*meghatározás*:*érték*;}
A kijelölt háromféle elem azonos formázást kap.

Specifikus kijelölés (kijelölések egymásba ágyazása): *elem1 elem2 elem3* {*meghatározás*:*érték*;}
Itt minden, **elem1**-ben lévő **elem2**-be tartozó **elem3** a meghatározott formázást kapja.

II. CSS – méretezés

Magasság: {height:*100px/100em/100%/auto/inherit*;}
Szélesség: {width:*100px/100em/100%/auto/inherit*;}
Legnagyobb magasság: {max-height:*érték/auto/inherit*;}
Legnagyobb szélesség: {max-height:*érték/auto/inherit*;}
Legkisebb magasság: {min-height:*érték/auto/inherit*;}
Legkisebb szélesség: {min-height:*érték/auto/inherit*;}

Az (inline/belső/külső) CSS mindig felülírja a statikus inline attribútumokat (pl.: height,width, align).

III. CSS – megjelenítés és láthatóság

Megjelenítés/eltüntetés(display)

Nem jelenik meg (hely sincs hagyva): {display:none;}

Tömszerű elem (teljes frame-szélesség és sorközök): {display:block;}
Alapértelmezésben tömbszerű elemek: **h1>**, **<p>**, ** **, **<li>**.

Sorbazárt elem (saját szélesség és szóközök): {display:inline;}
Alapértelmezésben sorbazárt elemek: ****, **<a>**.
[Pl.: span {display:block;} beállítás mellett a dokumentum összes **** eleme tömbszerűen jelenik meg.]

Láthatóság (visibility)

Nem látható (de helye van): {visibility:hidden;}
Nem látható (hely sincs hagyva): {visibility:collapse;}

A megjelenítési és láthatósági beállítások csak az elem kinézetét befolyásolják, a típusát nem változtatják meg. Pl. egy block megjelenésű inline elembe sem szabad egy (valódi) block elemet beágyazni!

A tömbszerű elemeket mindig sorközök választják el; a sorbazártak között azonban csak akkor van szóköz, hogyha azt a kód szóköz vagy enter formájában eleve tartalmazza!
[Tehát pl. a *PéldapéldaPélda* kód megjelenése: *PéldapéldaPélda*;

szemben a *Példa példa Példa* kóddal, mely így jelenik meg: *Példa példa Példa*.]

IV. CSS – pozícionálás

Pozícionálási típus: {position:*static/fixed/relative/absolute/inherit*;}

Statikus pozícionálás: {position:static;}
A tagek sorrendjéből és a böngésző alapértelmezéséből áll össze (nem rendelhető hozzá további stílus). Az elem körül a böngésző autiomatikusan helyet biztosít, így az nem fedhet át másokat.

Fix pozícionálás: {position:fixed;}
A böngésző-ablakhoz (frame-hez) képest állandó helyzetet adhatunk meg. A fix elrendezésű elem átfedése kerülhet más elemekkel (nincs hely kihagyva számára), és nem mozog együtt a legördülő oldallal.

Relatív pozícionálás: {position:relative;}
Az elemet eredeti, statikus helyzetéhez képest pozícionálja. Az eredeti helye megmarad, de új nem alakul ki, így átfedésbe kerülhet más elemekkel. A legördülő oldallal együtt mozog. A relatív elhelyezésű elemeket gyakran az abszolút elhelyezésűek „tároló-dobozaként” használjuk.

Abszolút pozícionálás: {position:absolute;}
Az abszolút pozíciójú elemeket az őket tartalmazó, nem statikus elhelyezésű elemekhez képest helyeztük el. Hogyha efféle elem nincs, akkor a legkülső, **<html>** szolgál e célra. Az elemnek nincs sem eredeti, sem új hely kihagyva, így átfedhet másokat és az oldallal együtt gördül le.

Pozícionálási érték: {*top/right/bottom/left*:*100px/100em/100%/auto/inherit*;}

Felső: {top:*100px/100em/100%/auto/inherit*;}
Jobb: {right:*100px/100em/100%/auto/inherit*;}
Alsó: {bottom:*100px/100em/100%/auto/inherit*;}
Bal: {left:*100px/100em/100%/auto/inherit*;}
Az érték megadja az adott oldali elemmargó vonalának távolságát a frame azonos szélétől.

Elhelyezési szint (stack-order): {z-index:-*1/auto/inherit*;}
A nem statikusan pozícionált (egymást átfedhető) elemek szintjét mutatja meg. A magasabb értékűek lefedik a kisebbeket. Alapértelmezésben a kódban hátrébb álló tagek a magasabb értékűek.

Bevágás (négyzet alakú mezőbe): {clip:*rect*/*auto/inherit* (*0px,100px,100px, 0px*;}
Itt egy 100x100px-es képet vágtunk be egy négyzetes keretbe. Nagyobb ill. kisebb értékek mellett a kép felső, jobb, alsó és bal oldalából egy-egy rész eltűnik. Csak fix ill. abszolút pozícionálás mellett használható. Az *auto* beállítás mellett a kép teljes alakban jelenik meg.

Túlméretes elem kezelése: {overflow:*auto/scroll/visible/hidden/inherit*;}

Automatikus (alsó, oldalsó vagy mindkettő) csúszka: {overflow:auto;}
Két csúszka: {overflow:scroll;}
Kereten túlnyúló tartalom: {overflow:visible;}
Kereten csonkolt tartalom: {overflow:hidden;}
Alapbeállítás szerint: {overflow:inherit;}

Kurzor-típusok (cursor):

Automatikus (az adott tartalomhoz illeszkedő): {cursor:auto;}
Kereszt: {cursor:crosshair;}
Alapértelmezett (hagyományos, nyílszerű): {cursor:default;}
Szöveg (szövegkurzor): {cursor:text;}
Homokóra: {cursor:wait;}
Homokórás egérnyíl: {cursor:progress;}
Súgó/segítség (kérdőjel): {cursor:help;}
Mutató-kéz (pl. linkek fölött): {cursor:pointer;}
Mozdító-kéz: {cursor:move;}

Felső ablak-átméretező jel (north-resize): {cursor:n-resize;}
Jobb-felső ablak-átméretező jel (northeast-resize): {cursor:ne-resize;}
Jobboldali ablak-átméretező jel (east-resize): {cursor:e-resize;}
Jobb-alsó ablak-átméretező jel (southeast-resize): {cursor:se-resize;}
Alsó ablak-átméretező jel (south-resize): {cursor:s-resize;}
Bal-alsó ablak-átméretező jel (southwest-resize): {cursor:sw-resize;}
Baloldali ablak-átméretező jel (west-resize): {cursor:w-resize;}
Bal-felső ablak-átméretező jel (northwest-resize): {cursor:nw-resize;}

Hogyha az igazítandó elemet tartalmazó elem szélességét is megadjuk, és a **DOCTYPE** hiányzik, az IE egy 17 pixeles margót ad a jobboldalhoz; a csúszkának szánt hely gyanánt.
Ezért a position jellemző használatakor mindig állítsuk be a DOCTYPE-ot!

V. CSS objektum-igazítás

Igazítási irány: {float:*left/right/none/inherit*;}

Igazítási irány tiltása (clear):
Az elem baloldala után nem igazodhat a következő tartalom: {clear:left;}
Az elem jobboldala után nem igazodhat a következő tartalom: {clear:right;}
Az elem egyik oldala után sem igazodhat tartalom: {clear:both;}
Alapértelmezett eset (általában a both-nak megfelelő): {clear:none;}
Az anya-elem beállításával megegyező: {clear:inherit;}

Tömbszerű elem (aláírással ellátott kép) formázása és igazítása (div kijelöléssel):

```
div {
float:right;
width:100px;
margin:0 0 100px 100px;
padding:100px;
border:100px solid black;
text-align:center;}

<div>
<img src="URL" width="100" height="100" /><br />
Kép-aláírás
</div>
```

Inline elem formázása és igazítása (span kijelöléssel):

```
span {
float:left;
width:1em;
font-size:400%;
font-family:algerian,courier;
line-height:100%;}

<p>
<span>T</span>his is some text.
</p>
```

Vízszintes menüsor készítése:
A **float** jellemzőt az **<ul>**, **<li>** és **<a>** elemeknél egyaránt be kell állítani, a végeredmény kontrollálhatósága érdekében:

```
ul {
float:left;
```

```
width:100%;
padding:0;
margin:0;
list-style-type:none;}

li {
float:left;
display:inline;}

a {
float:left;
width:6em;
text-decoration:none;
color:white;
background-color:purple;
padding:0.2em 0.6em;
border-right:1px solid white;}

a:hover {
background-color:#ff3300;}

<ul>
<li><a href="URL">Link one</a></li>
<li><a href="URL">Link two</a></li>
<li><a href="URL">Link three</a></li>
<li><a href="URL">Link four</a></li>
</ul>
```

Függőleges menüsor készítése:
Az előzőhöz hasonló, de a lista-elemek tömbszerűek (display:block;) és csak magát a listát igazítjuk az oldal szélére (ul {float:*left*;}), a benne lévő elemeknek nem adunk **float** jellemzőt.

Egyszerű weboldal-struktúra:
Élőfej, oldalmenü, tartalom és élőláb **div**-eket hozunk létre, majd az élőfej és élőláb után ill. előtt megtiltjuk az igazodást {clear:both;}, és az oldalmenüt a megfelelő oldalra igazítjuk {float:*left/right/none/inherit*;}.

VI. CSS – vízszintes igazítás

Középre igazítás:

Margóval (Firefox): {margin:auto;}
Margóval (IE, csak DOCTYPE mellett):
A pozícionálandó elemet tartalmazó elemre nézve: {text-align:center;}
A pozívionálandó elemre nézve: {margin-left:auto;margin-right:auto;}

Vízszintes igazítás position jellemzővel:

Előfeltételek: 1.) Abszolút pozícionálás esetén a pozícionálandó elemet tartalmazó elem pozícionálása nem lehet statikus.
2.) Az eltérő alapértelmezések kiküszöbölésére állítsunk be pontos térköz- és margóértékeket a **<body>**-ra nézve (pl.: {padding:0;margin:0;})!
3.) Az IE-rel való használhatóság érdekében mindig állítsunk be megfelelő DOCTYPE-ot!

Pozícionálási módok:

Statikus: {position:static;}
Fix: {position:fixed;}

Relatív: {position:relative;}
Abszolút: {position:absolute;}

Pozícionálási értékek:

Felső: {top:*100px/100em/100%/auto/inherit*;}
Jobb: {right:*100px/100em/100%/auto/inherit*;}
Alsó: {bottom:*100px/100em/100%/auto/inherit*;}
Bal: {left:*100px/100em/100%/auto/inherit*;}
Az érték megadja az adott oldali elemmargó vonalának távolságát a frame azonos szélétől.

Vízszintes igazítás float jellemzővel:

Előfeltétel: 1.) Az eltérő alapértelmezések kiküszöbölésére állítsunk be pontos térköz- és margóértékeket a **<body>**-ra nézve (pl.: {padding:0;margin:0;})!
2.) Az IE-rel való használhatóság érdekében mindig állítsunk be megfelelő DOCTYPE-ot!

Igazítási irány: {float:*left/right/none/inherit*;}
Igazítási irány tiltása: {clear:*left/right/both/none/inherit*;}

VII. CSS ál-osztályok

Mondattan:

Ál-osztály: elem:ál-osztály {jellemző:érték;}
Elem-osztályon belüli ál-osztály: elem.osztály:ál-osztály {jellemző:érték;}
Elemen belüli elem (ál-osztállyal): elem elem:ál-osztály {jellemző:érték;} vagy
elem > elem:ál-osztály {jellemző:érték;}
Adott ál-osztályba tartozó elemeken belüli elem: elem:ál-osztály elem {jellemző:érték;} vagy
elem:ál-osztály > elem {jellemző:érték;}

Ál-osztályok:

Link (nem nézett): :link
Link (megtekintett): :visited
Egér-érintés: :hover
Aktív állapot: :active

Első al-elem: :first-child
[Pl.: **p:first-child** kijelölés esetén a dokumentum minden szintű elemébe tartozó első bekezdés adott stílusú lesz.]

Nyelvtani ál-osztály: :lang(*név*)
[Pl.: **q:lang(*név*) {quotes:"~" "~";}** formázás és
<q lang="*név*">*Szöveg*</q>
formázott elem (idézet) esetén az idézet elején és végén idézőjel helyett ~ áll.]

Kijelölt szövegbeviteli elem: :focus
[Pl.: **input:focus {background-color:yellow;}** beállítás esetén használatban lévő szövegbeviteli mezők háttere sárga lesz.]

VIII. CSS ál-elemek

Mondattan:

Ál-elem: elem:ál-elem {jellemző:érték;}
Elem-osztályon belüli ál-elem: elem.osztály:ál-elem {jellemző:érték;}
Elemen belüli elem (ál-elemmel): elem elem:ál-elem {jellemző:érték;} vagy
elem > elem:ál-elem {jellemző:érték;}

Ál-elemek:

Első sor: :first-line

[A **:first-line** ál-elemet csak tömbszerű HTML-elemekre vonatkoztathatjuk. **:first-line** kijelölés mellett a következő jellemzők állíthatók be: **font**, **color**, **background**, **word-spacing**, **letter-spacing**, **text-decoration**, **vertical-align**, **text-transform**, **line-height**, **clear**.]

Első betű: :first-letter
[A **:first-letter** ál- elemet is csak tömbszerű HTML-elemek formázására használhatjuk. A beállítható jellemzők: **font**, **color**, **background**, **margin**, **padding**, **border**, **text-decoration**, **vertical-align** (csak akkor, ha **float:none;**), **text-transform**, **line-height**, **float**, **clear**.]

Megelőző tartalom: :before
[A **:before** ál-elemmel bizonyos tartalmat szúrhatunk be a kijelölt elemek tartalma elé.
Pl.: **h1:before {content:url(smiley.gif);}** beállítás esetén minden **<h1>** elem előtt egy kép lesz.]

Lezáró tartalom: :after
[Az **:after** ál-elemmel minden kijelölt elem mögé beszúrhatunk egy bizonyos tartalmat.
Pl.: **h1:after {content:url(smiley.gif);}** beállítás esetén minden **<h1>** elem után egy kép lesz.]

IX. CSS – navigációs sáv

Kiindulási alap (linkek nem-hierarchikus listája):

ul {list-style-type:none;margin:*0*;padding:*0*;}

```
<ul>
<li><a href="URL">Szöveg</a></li>
<li><a href="URL">Szöveg</a></li>
<li><a href="URL">Szöveg</a></li>
<li><a href="URL">Szöveg</a></li>
</ul>
```

Függőleges navigációs sáv (a kiindulási alap kiegészítésével):

Egyszerű függőleges menü, tömbszerű (változó szélességű) listaelemekből:

a {display:block;width:*60px*;background-color:*#dddddd*;margin-bottom:*2px*;}

Kidolgozottabb függőleges menü, tömbszerű (egyenlő szélességű) listaelemekből:

li {display:inline;}

a:link,a:visited
{display:block;font-weight:*bold*;color:*#ffffff*;background-color:*#98bf21*;width:*110px*;text-align:*center*;padding:*25px*;text-decoration:*none*;text-transform:*uppercase*;margin-bottom:*10px*;}

a:hover,a:active
{background-color:*#7a991a*;}

Vízszintes navigációs sáv (a kiindulási alap kiegészítésével):

Egyszerű vízszintes menü, sorbazárt (változó szélességű) listaelemekből:

li {display:*inline*;}

Kidolgozottabb vízszintes menü, sorbazárt (változó szélességű) listaelemekből:

li {display:*inline*;}

a:link,a:visited
{font-weight:*bold*;color:*#ffffff*;background-color:*#98bf21*;text-align:*center*;padding:*6px*;text-decoration:*none*;text-transform:*uppercase*;}

a:hover,a:active
{background-color:*#7a991a*;}

Egyszerű vízszintes menü, igazított listaelemekből:

ul {list-style-type:none;margin:0;padding:0;overflow:hidden;}

li {float:left;}

a {display:block;width:60px;text-align:center;background-color:#dddddd;margin-

right:4px;}

Kidolgozottabb vízszintes menü, igazított listaelemekből:

ul {list-style-type:none;margin:0;padding:0;overflow:hidden;}

li {float:left;}

a:link,a:visited

{display:block;width:120px;font-weight:bold;color:#FFFFFF; background-color:#98bf21;text-align:center;padding:4px; margin:2px;text-decoration:none;text-transform:uppercase;}

a:hover,a:active

{background-color:#7A991A;}

Vízszintes böngészősávot kétféleképpen készíthetünk; sorbazárt vagy igazított listaelemekből.

Mindkét módszer eredményesen alkalmazható; de ha a gombok méretének egyeznie kell, akkor az igazításos (**floating**) módszert válasszuk!

X. CSS – képgaléria

Egyszerű képgaléria balra igazított, változó keret-színű link-képekkel és képaláírásokkal:

```
div.img
    {
     margin: 2px;
     border: 1px solid #0000ff;
     height: auto;
     width: auto;
     float: left;
     text-align: center;
    }
div.img img
    {
     display: inline;
     margin: 3px;
     border: 1px solid #ffffff;
    }
div.img a:hover img {border: 1px solid #0000ff;}
div.desc
    {
     text-align: center;
     font-weight: normal;
     width: 110px;
     margin: 2px;
    }

<div class="img">
 <a target="_blank" href="URL">
<img src="URL" alt="megnevezés" width="100" height="100" />
</a>
 <div class="desc">Képaláírás</div>
</div>
```

XI. CSS kép-átlátszóság

Az opacity (=átlátszatlanság) CSS jellemző:

Internet Explorerben: {filter:alpha(opacity=*100*);}

Firefoxban és egyéb böngészőkben: {opacity:*1.00*;}

Az opacity jellemző JavaScript-ként:

Internet Explorerben: ”this.filters.alpha.opacity=*100*;”
Firefoxban és egyéb böngészőkben: ”this.style.opacity=*1.00*;”

Homályosított kép (helyi formázással):

<img src="*URL*" width="*100*" height="*100*" alt="*megnevezés*"
style="opacity:*1.00*;filter:alpha(opacity=*100*)" />

Egérérintésre homályosodó kép (helyi JavaScript-formázással):

<img src="*URL*" style="opacity:*0.50*;filter:alpha(opacity=*50*)"
onmouseover="this.style.opacity=*1.00*;this.filters.alpha.opacity=*100*"
onmouseout="this.style.opacity=*0.50*;this.filters.alpha.opacity=*50*" />

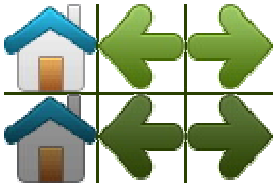
A kép-átlátszóság egyelőre nem része a CSS-standardoknak; ám a legtöbb modern böngészőben már használható, és a W3C CSS3 előterjesztésében is szerepel.

Az átlátszóbb elemekbe ágyazott összes további elem átlátszóbban fog megjelenni, amit nem lehet kompenzálni azok átlátszatlanságának megnövelésével, mert annak értéke legfeljebb 100 ill. 1.00 lehet, ami megfelel a kezdeti, már átlátszóbb megjelenésnek!
Ezért átlátszó elemekre átlátszatlanabbakat ne statikus pozícionálással (egymásba ágyazással) helyezzünk; hanem az összetartozó elemeket egy ** ** elemen belül egy szintre írjuk be, és a **{position=*fixed/absolute/relative*;}** valamint a **{z-index:*-1/auto/inherit*;}** jellemzők segítségével pozícionáljuk őket!

XII. CSS kép-szellemek

A kép-szellemek előnye, hogy egyetlen képfájl betöltésével számos képet jeleníthetünk meg (kevés számú lekérdezés, nincs holtidő pl. a:hover esetén megjelenítendő kép letöltésekor).

Egy kép-szellem összképe:



Kép-szellem beállítása (a háttér-kép pozícionálásával); általános alak:

background:url(*kép-szellem-URL*) *-100px 100px*;

Kép beszúrása kép-szellemből:

```
img.home
{
width:100px;
height:100px;
background:url(kép-szellem-URL) 0 0;
}
img.next
{
width:100px;
height:100px;
background:url(kép-szellem-URL) -100px 0;
}

<img class="home" src="átlátszó_kis_kép_URL-je" width="1" height="1" />
<img class="next" src="átlátszó_kis_kép_URL-je " width="0" height="0" />
```

Statikus háttérkép kép-szellemből:

```
#navlist                {position:relative;}
#navlist li             {margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a {height:44px;display:block;}

#home {left:0px;width:46px;}
#home {background:url(kép-szellem-URL) 0 0;}

#prev  {left:63px;width:43px;}
#prev  {background:url(kép-szellem-URL) -47px 0;}

#next  {left:129px;width:43px;}
#next  {background:url(kép-szellem-URL) -91px 0;}

<ul id="navlist">
        <li id="home"><a href="URL1"></a></li>
        <li id="prev"><a href="URL2"></a></li>
        <li id="next"><a href="URL3"></a></li>
</ul>
```

Egérérintésre változó háttérkép kép-szellemből:

```
#navlist                {position:relative;}
#navlist li             {margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a {height:44px;display:block;}

#home                   {left:0px;width:46px;}
#home                   {background:url(kép-szellem-URL) 0 0;}
#home a:hover           {background: url(kép-szellem-URL) 0 -45px;}

#prev                   {left:63px;width:43px;}
#prev                   {background:url(kép-szellem-URL) -47px 0;}
#prev a:hover           {background: url(kép-szellem-URL) -47px -45px;}

#next                   {left:129px;width:43px;}
#next                   {background:url(kép-szellem-URL) -91px 0;}
#next a:hover           {background: url(kép-szellem-URL) -91px -45px;}

<ul id="navlist">
        <li id="home"><a href="URL1"></a></li>
        <li id="prev"><a href="URL2"></a></li>
        <li id="next"><a href="URL3"></a></li>
</ul>
```

A kép-szellemek alkalmazásához is célszerű DOCTYPE-ot beállítani, pl.:

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

XIII. CSS – média-típusok

Média-típusok:

Összes	all
Beszéd	aural
Braille-monitor	braille
Braille-nyomtató	embossed
Palmtop/mobil	handheld

Nyomtató	print
Kivetítő	projection
Monitor	screen
Szegmens-tábla	tty
Televízió	tv

Média-típus jelölése belső vagy külső stílus-oldalban:

@media *screen*, *projection*, *print* {kijelölés {jellemző:érték;}}

Média-típus jelölése a <link> tagben:

<link rel=”stylesheet” type=”text/css” media=”*screen, projection, print*” href=”*URL*” />

XIV. CSS attribútum-kijelölések

Attribútum-kijelölés: [attribútum] {jellemző:érték;}

Elem-osztályon belüli attribútum-kijelölés: elem.osztály[attribútum] {jellemző:érték;}

Elemen belüli attribútum-kijelölés: elem[attribútum] {jellemző:érték;}

Attribútum és érték együttes kijelölése:

Egyetlen érték kijelölése:

[attribútum=érték] {jellemző:érték;}

<elem attribútum=”érték” />

Bármelyik kijelölése több, szóközzel elválasztott érték közül:

[attribútum~=érték2] {jellemző:érték;}

<elem attribútum=”érték1 érték2 érték3” />

Az első kijelölése több, kötőjellel elválasztott érték közül:

[attribútum|=érték1] {jellemző:érték;}

<elem attribútum=”érték1-érték2-érték3” />

Űrlap formázása attribútum-kijelölésekkel:

```
input[type="text"] {
width:100px;
display:block;
margin-bottom:100px;
background-color:yellow;
}
input[type="button"] {
width:100px;
margin-left:100px;
display:block;
}

<form name="input" method="get">
Firstname:<input type="text" name="Name" value="Peter" size="100">
Lastname:<input type="text" name="Name" value="Griffin" size="100">
<input type="button" value="Example Button">
</form>
```

Az érték-kijelöléskor az idézőjelek elhagyása vagy használata egyenértékű!

Az Internet Explorer 7 (és magasabb) verziója csak DOCTYPE megadása mellett támogatja az attribútum-kijelölést, az alacsonyabb sorszámú verziók pedig egyáltalán nem!
Így attribútum-kijelölésekkor feltétlenül adjuk meg, pl.:

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

XV. CSS – kerülendő eljárások

Internet Explorer „magatatások” (behaviors)

Az Internet Explorer 5. verziójában tűnt fel az ún. **behavior** (=magatartás) attribútum. Ezeket a CSS-be írva bizonyos „viselkedésre” bírhatjuk a formázott HTML-elemet. Használatától azonban tartózkodnunk kell, mert *csak* az Internet Explorer támogatja. Helyette használjunk inkább JavaScriptet vagy (X)HTML DOM-ot!

Példák a behavior jellemző alkalmazására:

I.: <h1> színváltása egér-érintésre

CSS:
```
h1 {behavior:url("behave.htc");}
```

HTML:
```
<h1>Mouse over me!!!</h1>
```

XML:
```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />

<script type="text/javascript">
function hig_lite()
{
element.style.color='red';
}

function low_lite()
{
element.style.color='blue';
}
</script>
```

II.: gépírás-szimuláció

CSS:
```
#typing
{
behavior:url(behave_typing.htc);
font-family:"courier new";
}
```

HTML:
```
<span id="typing" speed="100">
<p>IE5 introduced DHTML behaviors. Behaviors are a way to add DHTML functionality to HTML
elements with the ease of CSS.</p>
</span>
```

XML:
```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />

<script type="text/javascript">
var i,text1,text2,textLength,t;

function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
```

```
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
{
clearInterval(t);
}
}
</script>
```

Egyéb hasznos tudnivalók:

- A * kijelölés a dokumentum összes szöveges elemére vonatkozik!
- Hogyha a jellemző-érték mögé az **!important** szöveget írjuk, azzal a beállítás prioritását megnövelük. Pl. hogyha a beállításunk
  `* {color:red;}`
  akkor a dokumentum összes szöveges eleme, melynek nincs specifikus színe megadva (amki az előbbi formázást felülírná), piros lesz. Hogyha most a red érték mögé beírjuk az **!important** zsöveget is:
  `* {color:red !important;}`
  akkor azok az elemek is pirosak lesznek, melyeknek korábban más, specifikusabb betűszínt is megadtunk.
- A
  `<link rel="stylesheet" type="text/css" media="screen,projection,print" href="URL" />`
  helyett a
  `<style type="text/css" title="név" media="screen">`
  `@import "URL";`
  `</style>`
  taggel is beidézhetjük egy külső dokumentum stílus-adatait. Míg a **<link>** tag a dokumentum egész tartalmát beodézi, addig az **@import** csak a formázási adatokat szemezi ki. Hogyha azok eleve egy **<style>** tagben vannak a másik oldalon (HTML-oldal belső CSS-e), akkor az első formázási utasítást (ami a **<style>** után következik) sem a **<link>**, sem az **@import** nem tudja beolvasni. Hogyha a weboldal csak formázási adatokat tartalmaz (külső CSS dokumentum), akkor az első is rendben bejön. Tehát, ahogy a **<link>** tag, úgy az **@import** is csak egészében tudja kezelni a hivatkozott dokumentumot, így a formázás megfelelősége annak HTML-tartalmán is múlik.
- A
  `{font:normal 90%/170% "Trebuchet MS", Verdana, Arial, Helvetica, sans-serif;}`
  meghatározásban az első érték a betűhatásra vonatkozik, a harmadik a betűtípusra, míg a második, kettős adat a betűméretet és a sormagasságot adja meg!
- `{background-color:transparent;}` beállítás is elképzelhető.
- A **z-index** jellemző statikus pozícionálású elemre nézve nem használható.
- Inline (sorbazárt) elembe nem szabad block (tömbszerű) elemet beágyazni!
- Hogyha egy átlátszóvá tett elembe (statikusan) további elemeket ágyazunk, azok is homályosak lesznek ezért célszerű mellérendeléssel élnünk a statikus dokumentum-szerkezetben, és az elemeket egy ** **-be összefoglalva, nem-statikus **position** és a **z-index** jellemzők segítségével pozícionálni!