HTML – CSS

Ezt a tananyagot innen lehet letölteni: http://szilagyidonat.hu/tananyag.html

Szerző: Szilágyi Donát

E-mail: donat_szilagyi@hotmail.com

1 HTML

Mi az a HTML?

A HTML egy szöveges fájl, aminek a kiterjesztése html.

Két dolog van benne:

- Szöveg, ami a böngészőben fog megjelenni
- Speciális utasítások, amiket HTML elemeknek vagy tag-eknek (kiejtés: teg) hívunk

A weboldalakat HTML-ben készítik.

URL

Egy weboldalnak vagy egy fájlnak a címe. A böngésző címsorába ezt írjuk be.

Itt például topskills.eu

Az interneten lévő fájlok protokollja általában http. A böngészők általában a protokollt csak akkor írják ki, ha az nem http. A http protokollal egy távoli webszerveren szoktunk meghívni egy fájlt.

Az interneten lévő weboldalak kezdőoldala általában egy index.html nevű fájl, ezt viszont a webszerver szokta alapértelmezettnek venni, azért nem kötelező kiírni.

Szóval a topskills.eu URL-je igazából: http://topskills.eu/index.html

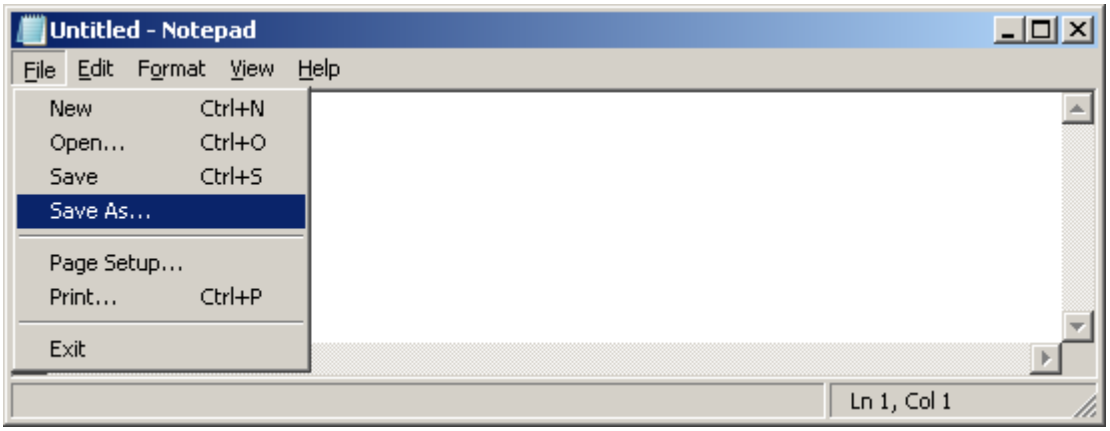
Viszont ha egy olyan fájlt nyitunk meg a böngészőben, ami a saját gépünkön van, akkor file lesz a protokoll, és az URL valahogy így fog kinézni: file:///C:/Donat/myweb.html

HTML FÁJL KÉSZTÉSE

Készítsünk egy HTML fájlt, legyen a neve például 1.html. Ezt kétféleképp érhetjük el:

- Total Commander, **Shift F4**

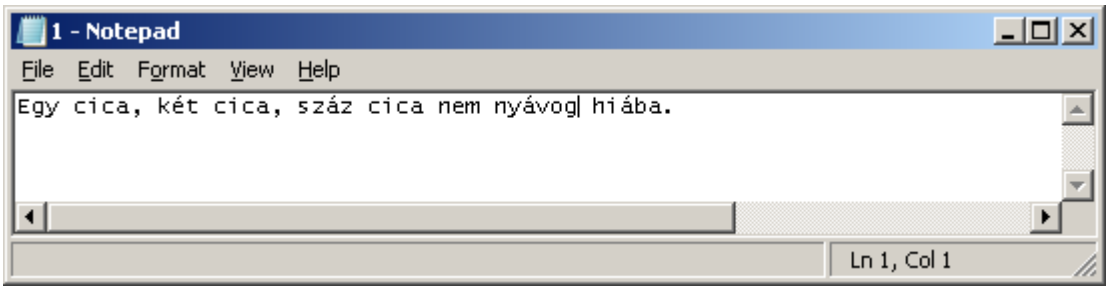
- Új Notepad nyit, **File > Save as**



Ezután állítsuk be a könyvtár és a fájl nevét.

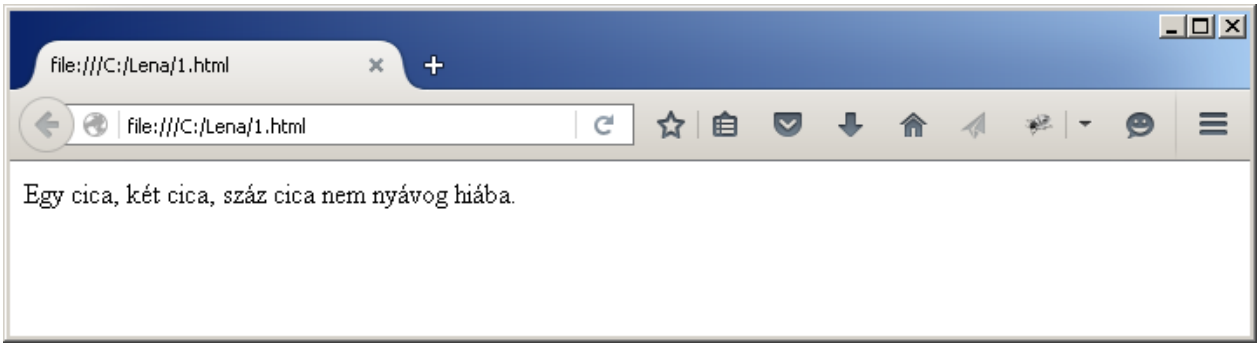
Írjuk be Notepad-ben az 1.html fájlba az alábbi szöveget:

Egy cica, két cica, száz cica nem nyávog hiába.



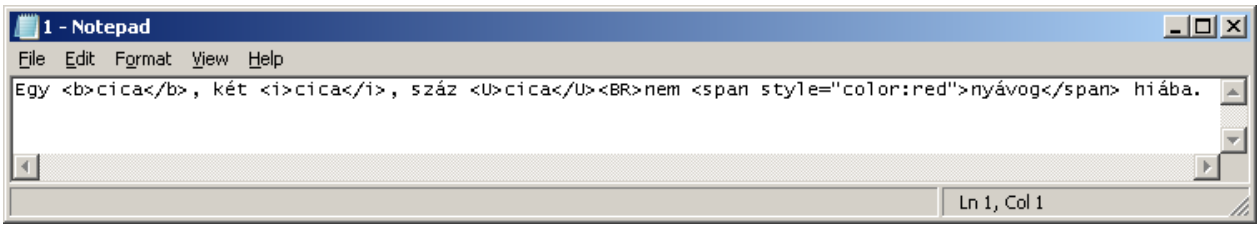
Mentsünk: **File > Save**, vagy billentyűzetről **Ctrl S**

Ha meg van nyitva a Total Commander, **Alt Tab** –bal (a Tab gomb bal oldalt található, egy balra és egy jobbra nyíl van rajta) ugorjunk át rá. Ha nincs megnyitva, nyissuk meg, és keressük meg az 1.html fájlt. Nyomjunk rajta egérrel duplaklikket. Megnyílik a böngészőben az 1.html fájl, és valami ilyesmit kellene látnunk:



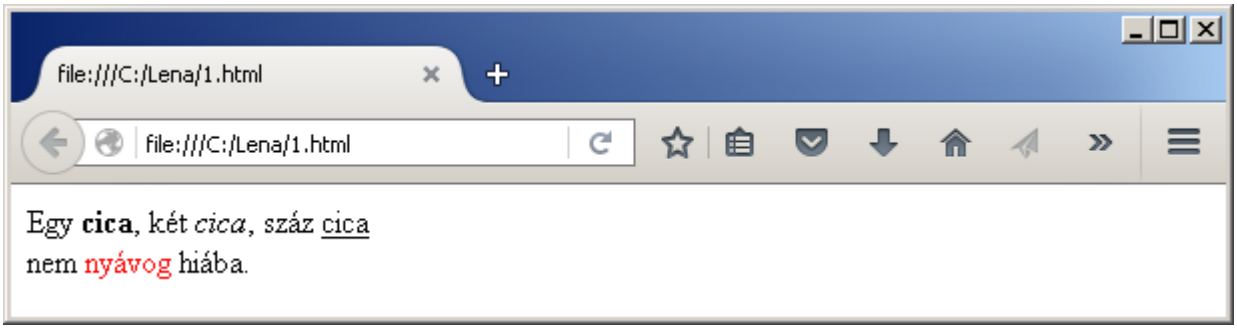
Ugorjunk vissza Notepad-ba **Alt Tab**-bal, és tegyünk HTML elemeket a szövegbe:

```
Egy <b>cica</b>, két <i>cica</i>, száz <U>cica</U><BR>nem
<span style="color:red">nyávog</span> hiába.
```



Mentsünk: **Ctrl S**

Alt Tab-bal ugorjunk át a böngészőbe. Nyomjuk meg az **F5** gombot



A böngészőben a tartalmat az **F5** gombbal frissíthetjük. Ez általában elég, de a **Ctrl F5** segítségével még mélyebb frissítést érhetünk el.

Elem (tag)

Vizsgáljuk meg alaposabban az előbb beírt szöveget:

```
Egy <b>cica</b>, két <i>cica</i>, száz <U>cica</U><BR>nem <span
style="color:red">nyávog</span> hiába.
```

A **** egy nyitó elem, a **** pedig egy záró elem. A köztük lévő szöveg vastagabb lesz. A záró elemet onnan lehet felismerni, hogy így kezdődik: **</**

A **** -hez hasonlóan működik az **<i>** (dőlt betű) és az **<u>** (aláhúzott betű) is.

Az elemeket mindegy, hogy nagybetűvel vagy kisbetűvel írjuk.

Van olyan elem is, aminek nincs záró eleme, például a **
**, ami új sort kezd. Ha ki akarjuk hangsúlyozni, hogy egy elemnek nincs záró eleme, így is írhatjuk: **
**

Egy elemnek lehetnek attribútumai (tulajdonságai) is. Például a **** esetében a **span** elemnek van egy **style** attribútuma (kiejtés: szájz), ami itt a betűk színét állítja pirosra.

Az elemek attribútumainak értékeit tehetjük dupla macskakörmök " vagy idézőjelek ' közé is. Csak az a lényeg, hogy egy elemen belül ne váltogassuk. Szóval ez is működik:

```
<span style='color:red'>nyávog</span>
```

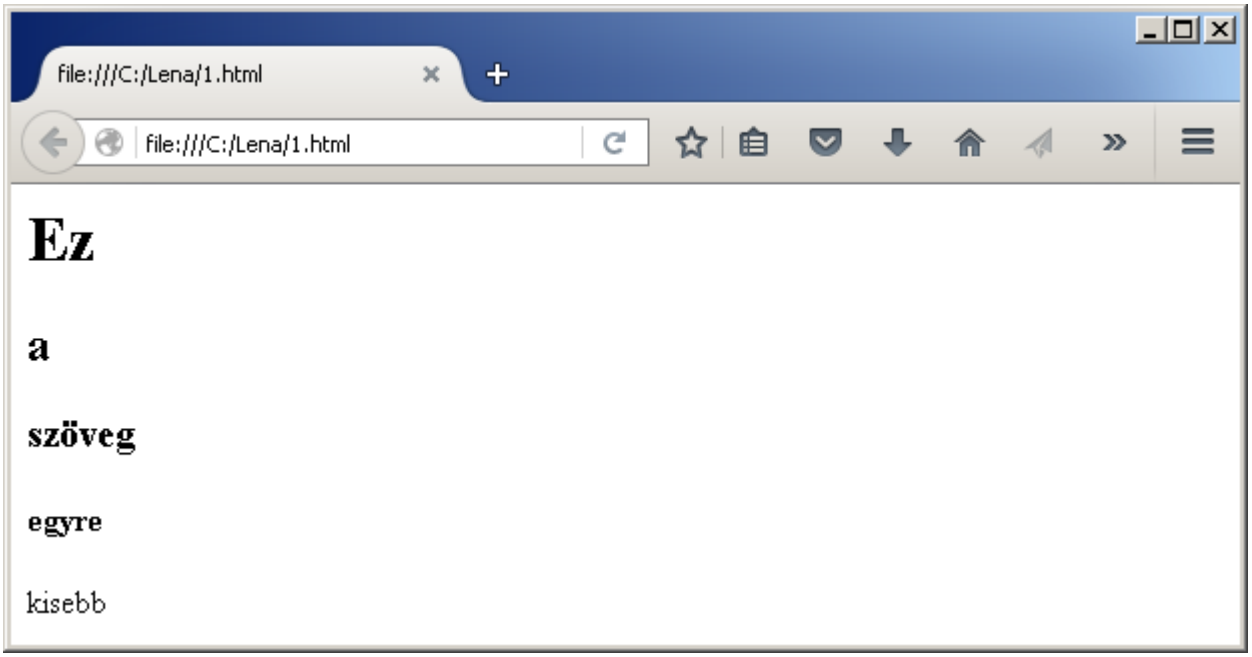
Az elemet angolul kétféleképpen is szokták hívni: element, tag

Az attribútum pedig angolul: attribute, property

MÉG NÉHÁNY ELEM

Ha ki akarunk emelni egy szövegrészt, azt a **h1**, **h2**, **h3**, **h4**, **h5**, **h6** segítségével tehetjük meg:

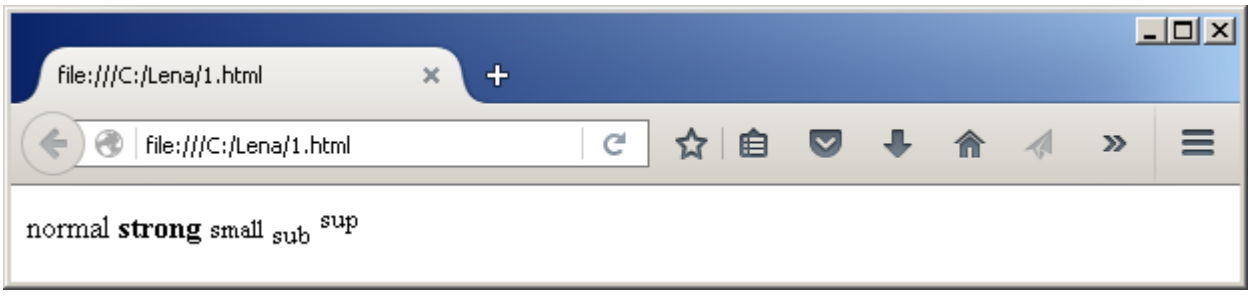
```
<h1>Ez</h1> <h2>a</h2> <h3>szöveg</h3> <h4>egyre</h4> kisebb
```



A táró **h1**, **h2**, **h3**, **h4**, **h5**, **h6** egyben soremelést is csinál, így nincs szükség utána **
**-re.

Van még néhány elem a betűk formázására: **strong**, **small**, **sub**, **sup**

```
normal <strong>strong</strong> <small>small</small> <sub>sub</sub> <sup>sup</sup>
```



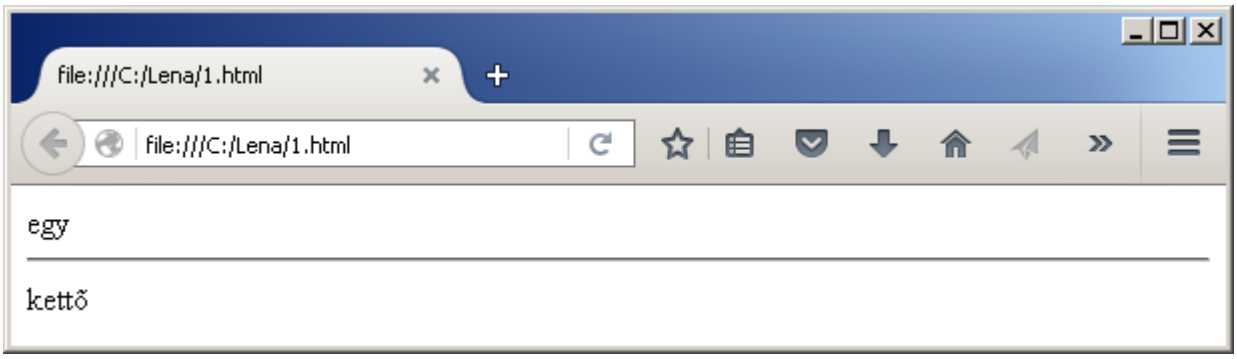
A **center** segítségével elérhetjük, hogy a szöveg ne balra, hanem középre kerüljön:

```
első sor<br>
<center>második sor<br></center>
```

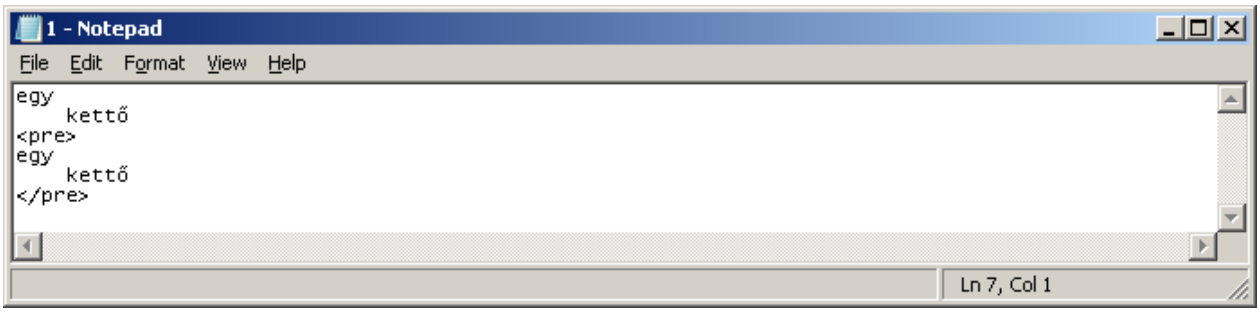
harmadik sor

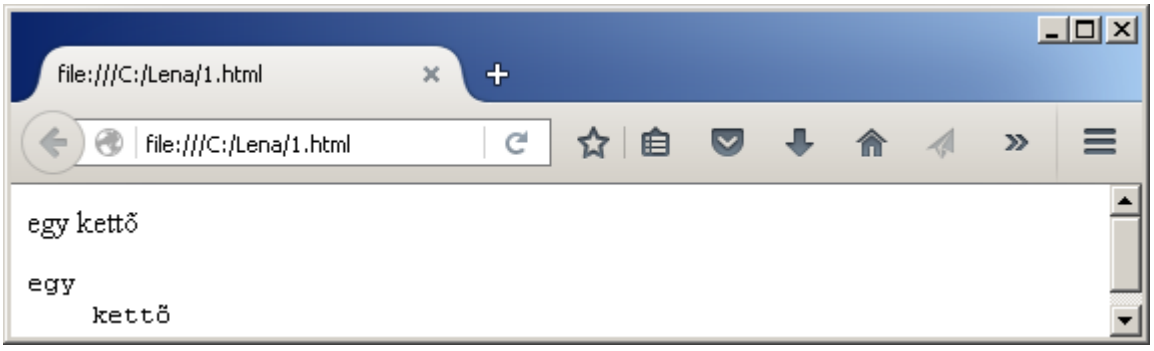
A **hr** segítségével vízszintes vonalakat húzhatunk. A **h1**, **h2** elemekhez hasonlóan nem igényel **br**-t.

egy<hr>kettő



A következő elem a **pre**. Ami a nyitó és a záró **pre** között van, az előformázott szöveg, ami megtartja az üres helyeket a szövegben, nem kell **
** -rel noszogatni.



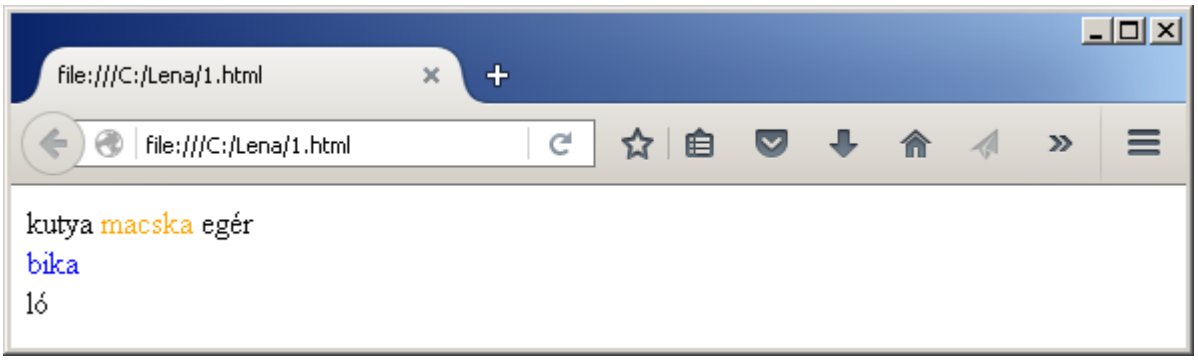


A **span** nem csinál semmit, de adhatunk neki stílust, mint korábban láthattuk.

nyávog

A **div** nagyon hasonlít a **span** elemhez, annyi a különbség, hogy új sort csinál annak, amit beleírunk.

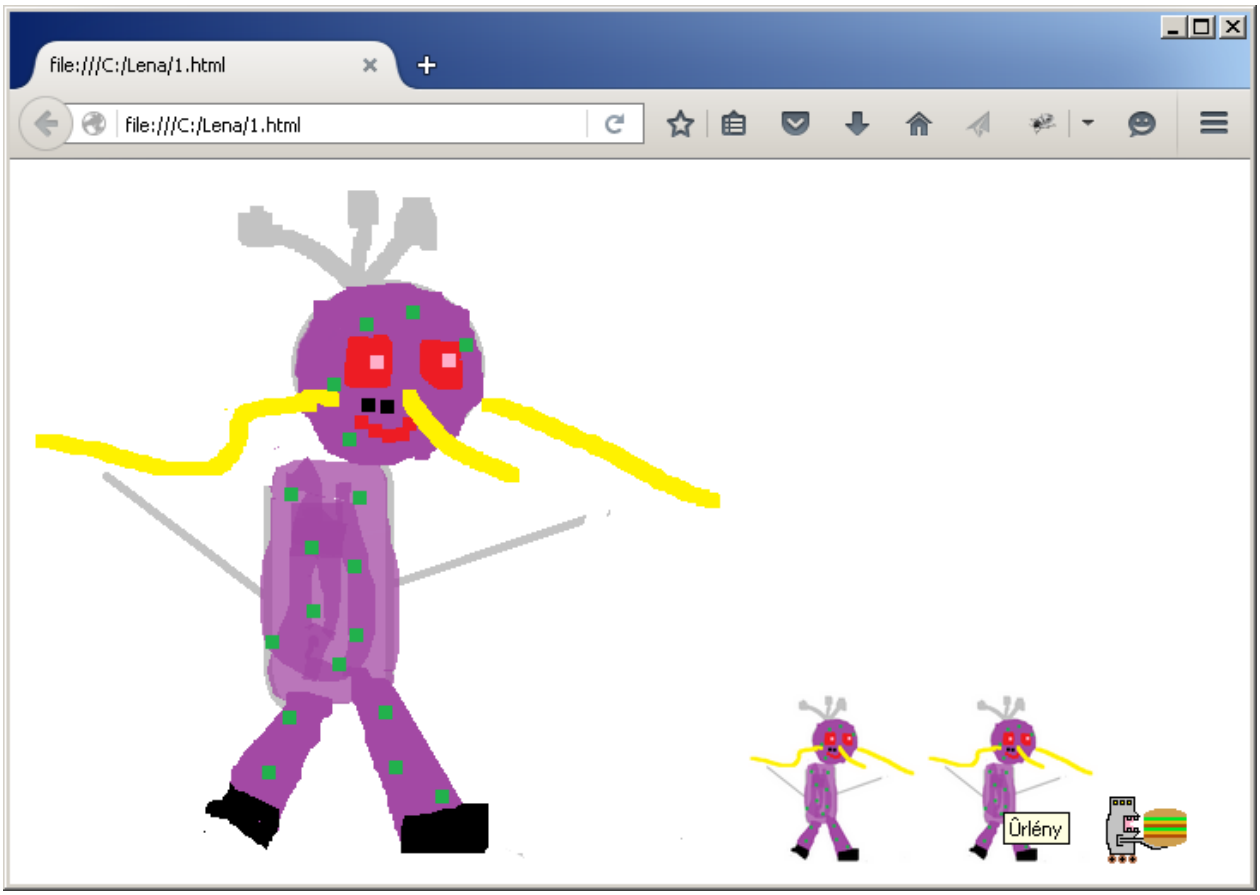
```
kutya <span style="color:orange">cica</span> egér <div style="color:blue">bika</div>ló
```



Képek

Képeket az **img** elem segítségével jeleníthetünk meg. Az **src** attribútumba kell beírni a képfájl nevét vagy URL elérését.

```
<img src="urleny.png">
<img src="urleny.png" style="width:100; height:100">
<img src="urleny.png" style="width:100; height:100" title="Űrlény">
<img src="http://topskills.eu/kepek/robot-bigmac.png">
```



A további attribútumok:

style: CSS tulajdonságokat adhatunk a képhez, a **width** a kép szélessége pixelben, a **height** a kép magassága. A tulajdonságokat pontosvesszővel választjuk el egymástól.

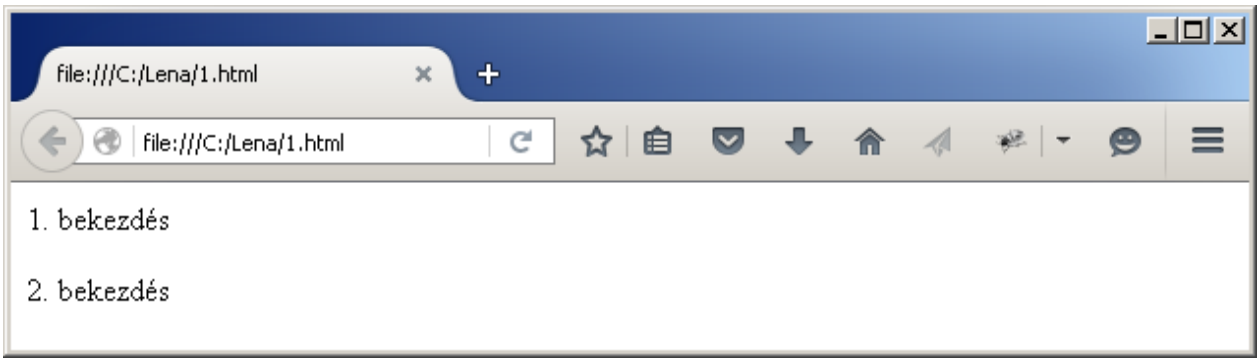
title: A kép címe, akkor jelenik meg, ha az egérrel a kép fölé megyünk.

Feladat: Rajzolj valamit Paint-ben, és jelenítsd meg HTML-ben, játszogass az attribútumokkal is. Ha az **src**-ben fájl nevét adsz meg, akkor a képfájl és a HTML fájl ugyanabban a könyvtárban kell, hogy legyen.

Bekezdések

Ha nem szeretjük a **br**-eket, **p**-t is használhatunk helyette. A **p**-hez záró elemet is kell adni.

```
<p>1. bekezdés</p>
<p>2. bekezdés</p>
```



Linkek

A linkekkel két HTML fájl között tudunk navigálni.

Csináljunk egy 1.html nevű fájlt:

```
<h1>Egy</h1>
<a href="2.html">ugorj át a 2-re</a>
```

Ugyanebbe a könyvtárba csináljunk egy 2.html nevű fájlt is:

```
<h1>Kettő</h1>
<a href="1.html">ugorj vissza az 1-re</a>
```

A **href**-ben külső weboldalt is hívhatunk:

```
<a href="http://topskills.eu">Topskills</a>
```

Ha azt is beletesszük az **a** elembe, hogy **target="blank"**, akkor egy új böngésző ablakban fog megnyílni az oldal.

```
<a href="http://topskills.eu" target="blank">Topskills</a>
```

Egy HTML oldalon belül is tudunk navigálni.

```
<a href="#alul">Ugorj az aljára</a>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
közepe
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<a name="alul">alja</a>
```

Az **<a href="#alul">** azt jelenti, hogy az oldalon belül ugorjon ide: **<a name="alul">**

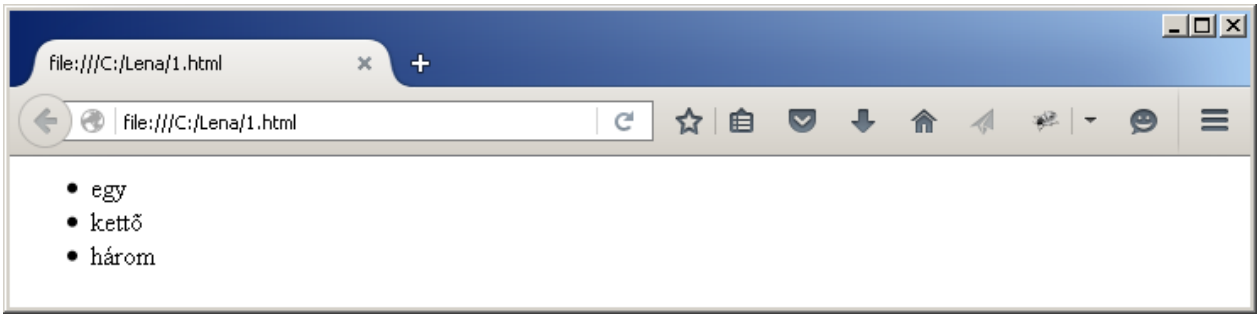
LISTÁK

Listát az **ul** és a **li** elemekkel tudunk csinálni.

```
<ul>
  <li>egy
  <li>kettő
  <li>három</li>
</ul>
```

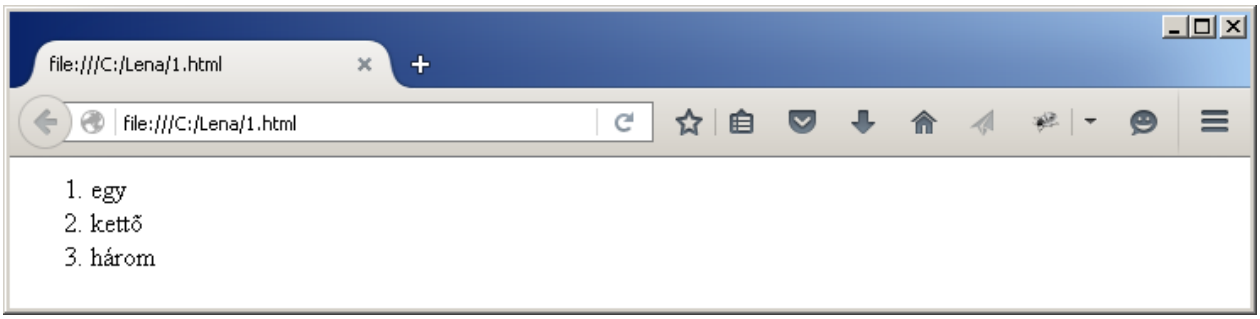
A **li** elemet nem kötelező bezárni, de megtehetjük.

HTML-ben ha egy nyitó elem és egy záró elem közé több sort is írunk, akkor a közbülső sorokat beljebb szoktuk kezdeni, például 4 space-szel.



Ha bogyók helyett számokat akarunk, akkor **ul** helyett **ol** elemet használunk:

```
<ol>
 <li>egy
 <li>kettő
 <li>három</li>
</ol>
```



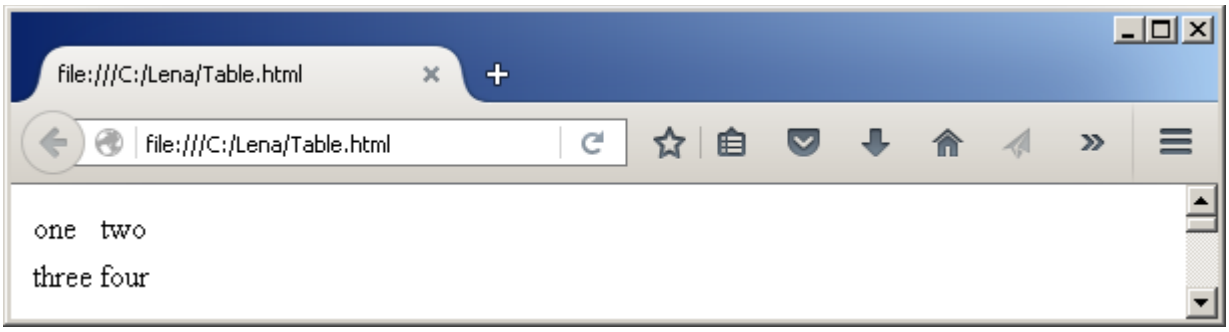
Az **ol**, **ul**, **li** az alábbiak rövidítése: ordered list, unordered list, list item.

TÁBLÁZATOK

Táblázatot a **table**, **tr**, **td**, **th** elemekkel csinálhatunk. A **table** jelenti az egész táblázatot, amin belül a **tr** egy sort jelent, és azon belül a **td** egy mezőt, más néven cellát jelent.

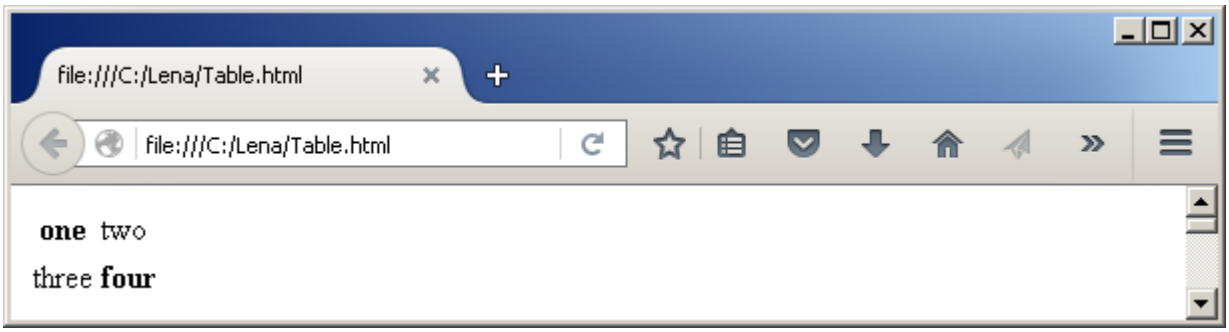
```
<table>
  <tr>
    <td>one</td><td>two</td>
  </tr>
  <tr>
    <td>three</td><td>four</td>
  </tr>
<table>
```

Alapesetben a táblázat elég egyszerűen néz ki:



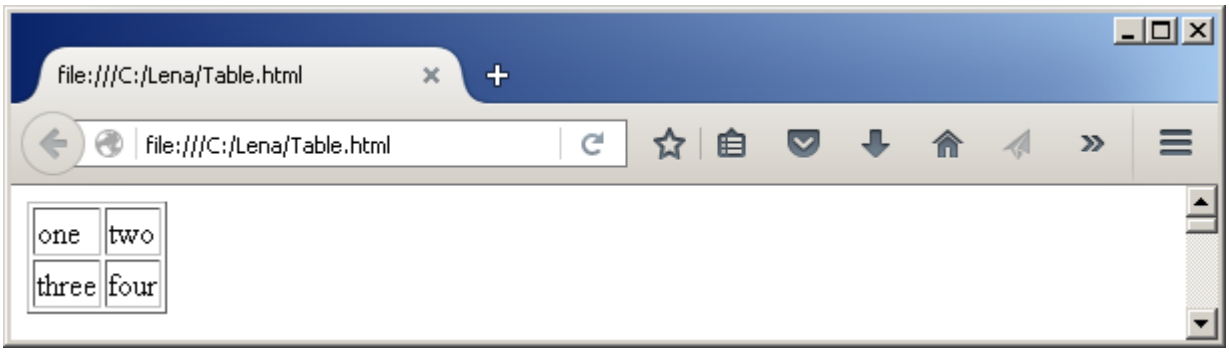
Ha valamelyik cellát ki akarjuk emelni, **td** helyett használhatunk **th**-t.

```
<table>
  <tr>
    <th>one</th><td>two</td>
  </tr>
  <tr>
    <td>three</td><th>four</th>
  </tr>
<table>
```



Ha látni akarjuk a táblázat vonalait is, akkor a **border** attribútumot kell 1-re állítani.

```
<table border="1">
   <tr>
      <td>one</td><td>two</td>
   </tr>
   <tr>
      <td>three</td><td>four</td>
   </tr>
<table>
```



Feladat: csinálj még egy sort a táblázatba.

Ha ennél szebb táblázatot szeretnénk, azt CSS segítségével érhetjük majd el.

HTML FELÉPÍTÉSE

Általában a HTML fájlok szövegét egy **html**, és azon belül egy **body** elembe szokás tenni.

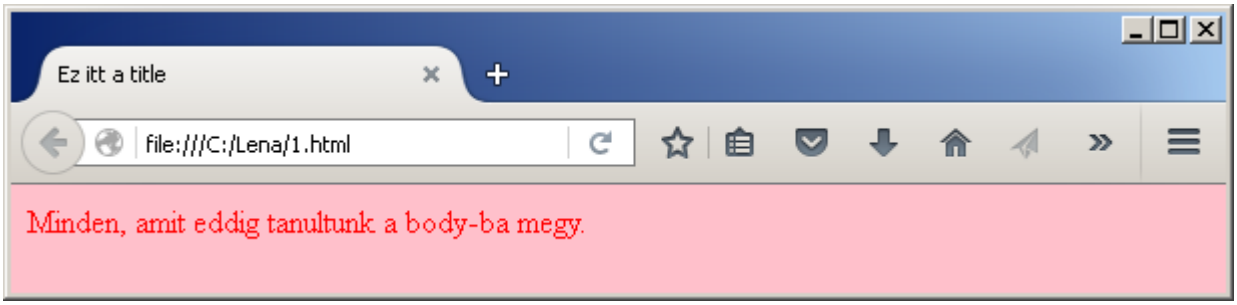
```
<html>
   <head>
   </head>
   <body>
      Minden, amit eddig tanultunk a body-ba megy.
   </body>
</html>
```

Szóval minden, amit eddig tanultunk a **body**-ba megy. De a **body** előtt van egy **head** elem is. A **head** elemet arra használhatjuk, hogy teszünk bele egy **title** elemet, amivel azt érjük el, hogy a böngésző fenn a fülön ne a HTML fájl útvonalát írja ki, hanem azt, amit a **title** elembe teszünk.

```
<html>
   <head>
      <title>Ez itt a title</title>
   </head>
   <body>
      Minden, amit eddig tanultunk a body-ba megy.
   </body>
</html>
```

A **body** elemnek pedig adhatunk **style** (kiejtés: sztájl) attribútumot.

```
<html>
  <head>
    <title>Ez itt a title</title>
  </head>
  <body style="color:red; background-color:pink">
    Minden, amit eddig tanultunk a body-ba megy.
  </body>
</html>
```



Majdnem minden elemnek adhatunk **style** attribútumot, aminek az értéke egy CSS kifejezés. A **color:red; background-color:pink** könnyen kitalálható módon az egész oldal tintaszínét és háttérszínét állítja be.

A fájl legelejére, a **<html>** elé szoktak tenni egy ilyen sort is:

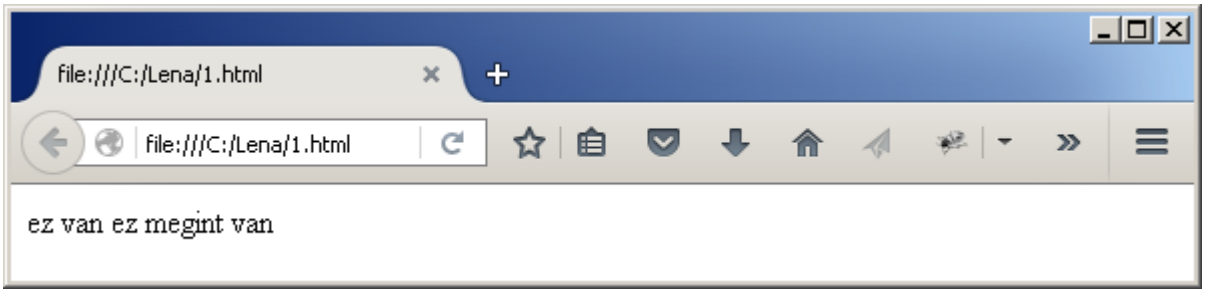
```
<!doctype html>
```

Ez annyit jelent, hogy a HTML újabb verziójában, vagyis HTML5-ben készült a fájl.

KOMMENTEK

Ha a HTML fájl egy részét ki akarjuk kapcsolni, azt úgy érhetjük el, hogy kikommentezzük. Amit <!--és --> közé írunk, az ki van kommentezve.

```
ez van <!-- ez nincs -->
ez megint van
<!--
ez megint nincs
-->
```



A kommentet arra is használhatjuk, hogy megjegyzést tegyünk a fájlba.

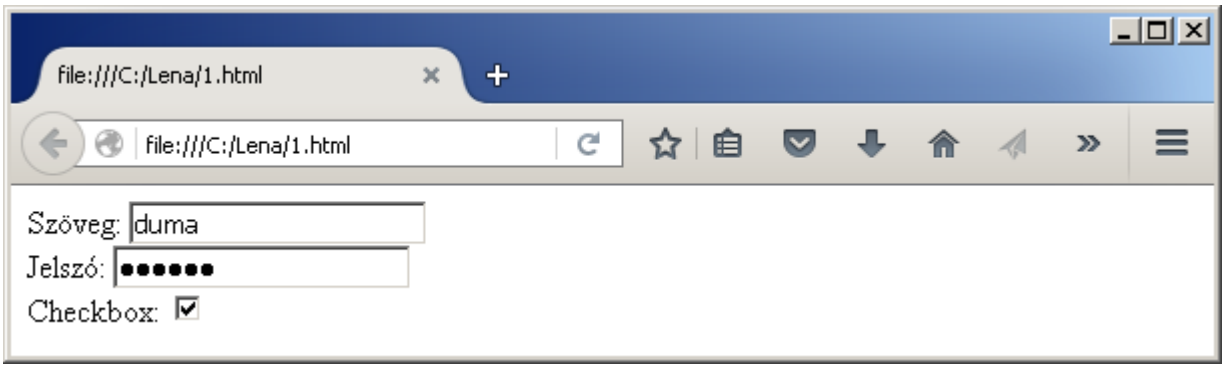
```
<!-- ez itt egy bekezdés -->
<p>bekezdés</p>
```

Űrlap

Az űrlapok olyan elemeket tartalmaznak, amikbe be lehet írni valamit, vagy amiket meg lehet nyomni.

Az űrlapokat általában egy **form** elemmel szoktuk körülvenni, de ez nem kötelező. Az **input** egy olyan űrlap elem, aminek elég sok típusa lehet, most a **text**, **password**, **checkbox** típusokkal ismerkedünk meg.

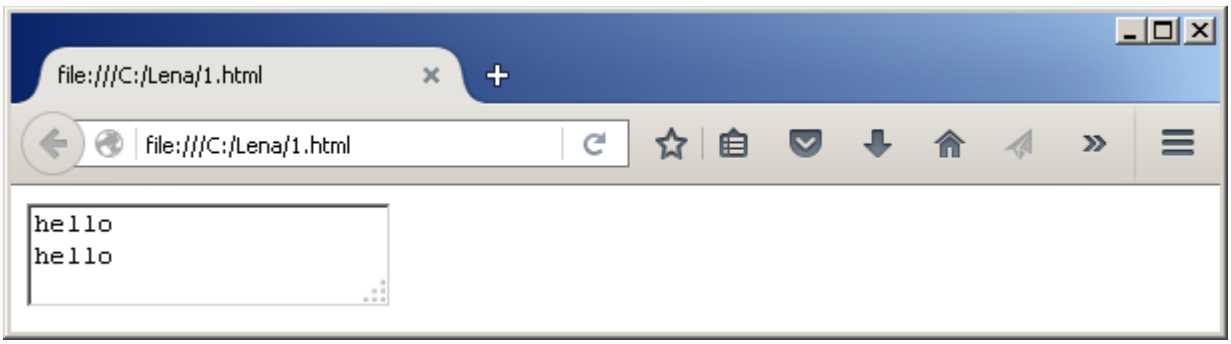
```
<form>
   Szöveg: <input type="text" value="duma"> <br>
   Jelszó: <input type="password" value="jelszó"> <br>
   Checkbox: <input type="checkbox" checked="true"> <br>
</form>
```



Az input elemnek a **value** attribútummal adhatunk kezdőértéket. Checkbox esetében pedig a **checked** attribútummal, ami true (igaz) vagy false (hamis) lehet.

A **textarea** elem egy többsoros szövegbeviteli mező.

```
<textarea rows="2" cols="20">
```

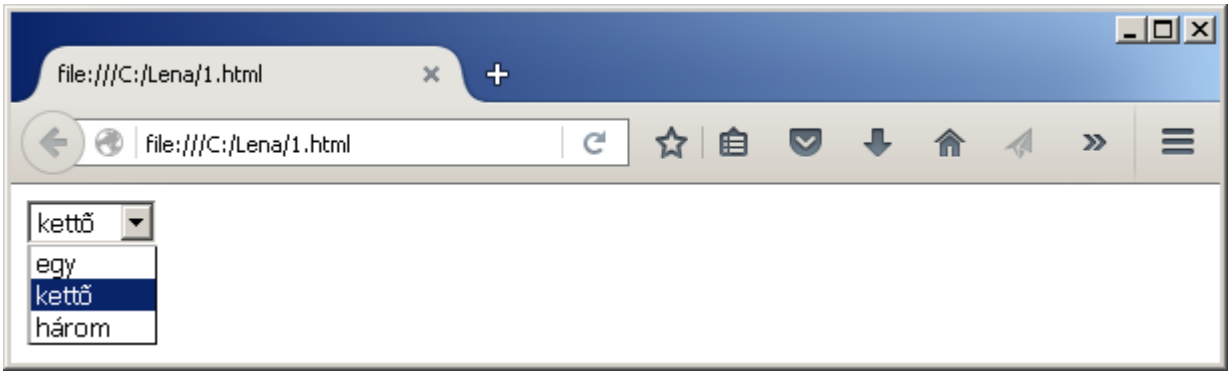


A **rows** mondja meg, hogy hány sornyi, a **cols** pedig, hogy hány oszlopnyi karakter férjen bele.

A karakter azt jelenti, hogy betű, szám vagy írásjel. Például: Ä

A **select** elemmel lenyíló listát készíthetünk.

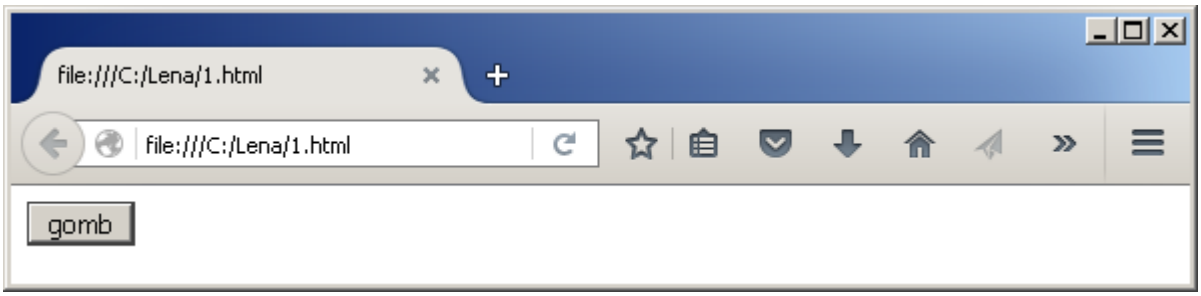
```
<select>
  <option>egy</option>
  <option selected>kettő</option>
  <option>három</option>
</select>
```



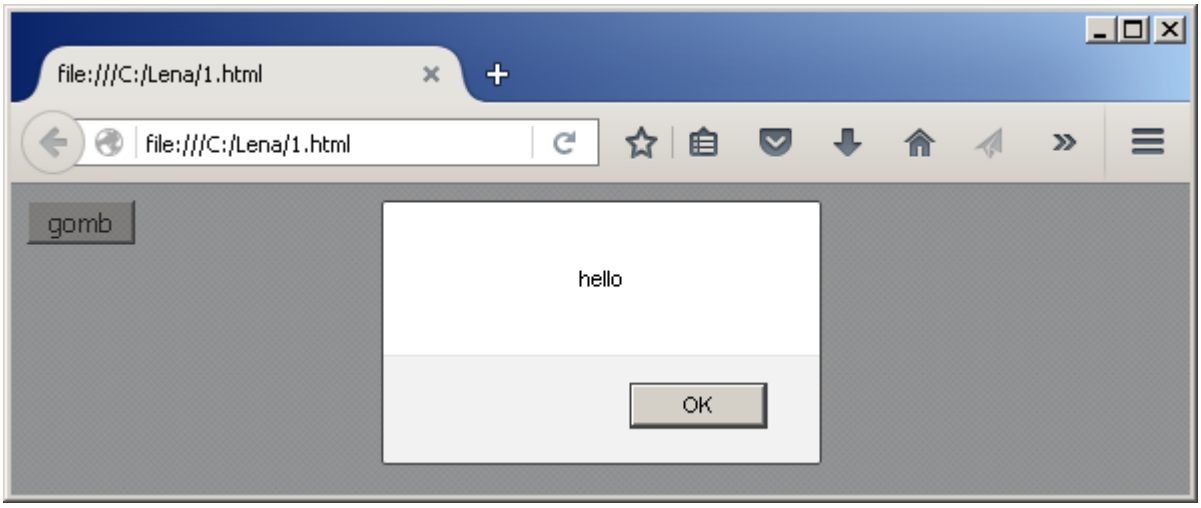
Az **option** elemben a **selected** attribútummal mondhatjuk meg, hogy melyik érték legyen kezdetben kiválasztva.

A **button** elemmel készíthetünk nyomógombot

```
<button onClick="alert('hello')">gomb</button>
```



Az **onClick** attribútumba JavaScript programot (más néven kódot) írhatunk. Az **alert** JavaScript parancs megjelenít egy üzenetet egy leokézható felugró ablakban.



SPECIÁLIS KARAKTEREK

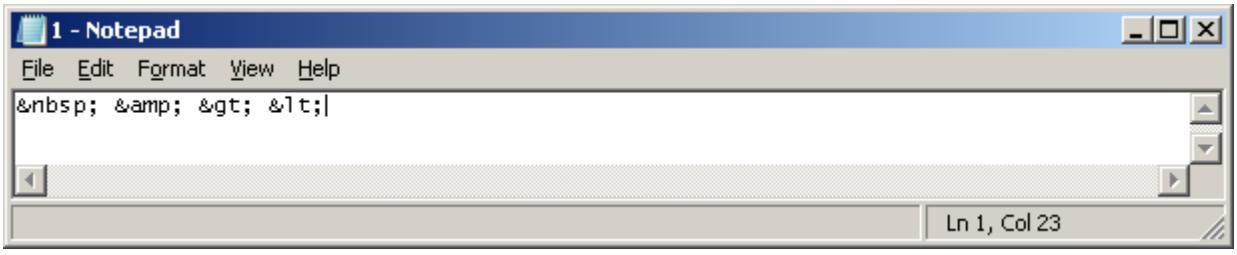
Szóval egy karakter azt jelenti, hogy egy darab betű, szám vagy írásjel. Például: Ä

Van olyan karakter, ami nem szerepelhet egy HTML fájlban, mert a böngésző azt hiszi róla, hogy egy HTML elem része. Ez a három ilyen van: < > &

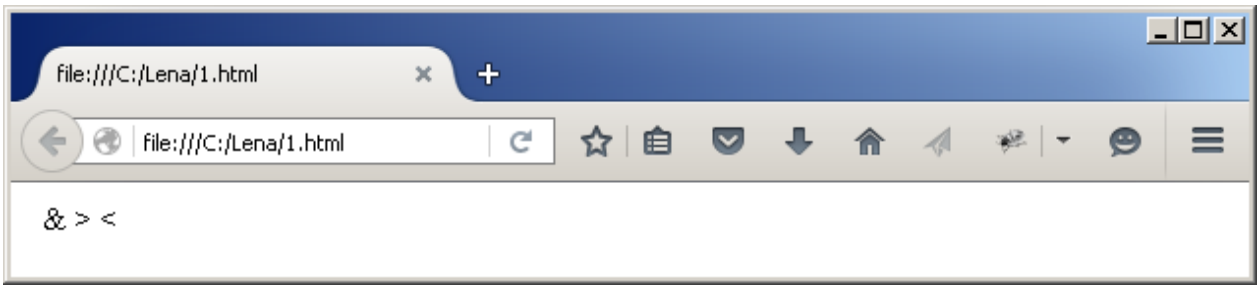
Ezeket a karaktereket ki szoktuk escape-elni (kiejtés: iszkép).

	üres hely
&amp;	&
&gt;	>
&lt;	<

  & > <



Böngészőben így fognak kinézni:



Az ** ** az üres helyet jelenti. A space gombbal (hosszú gomb, kiejtés: szpész) csinált üres helyből mindig csak 1 fog megjelenni, akárhányat írunk a HTML fájlba. Ezért ha több üres helyet akarunk kiírni, azt az ** ** –vel oldhatjuk meg:

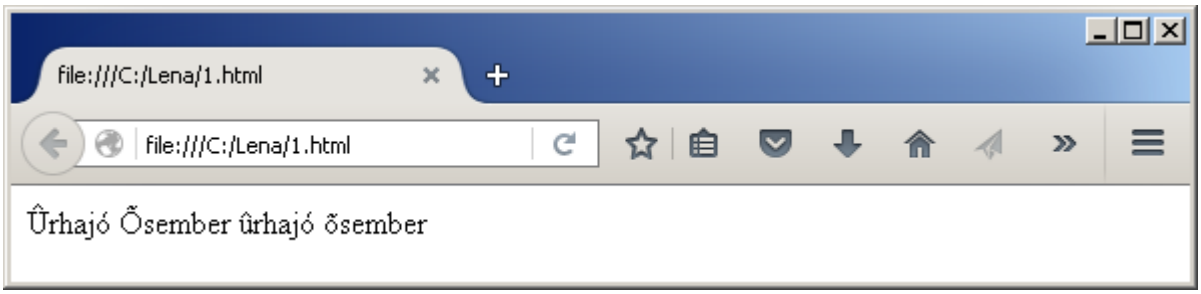
```
messze              innen
```

A korábbiakan tanult **pre** elemmel is elérhetjük, hogy több üres hely kerüljön egymás mellé.

META ELEMEK

Nézzük meg, mi történik, ha egy HTML fájlba az alábbit írjuk:

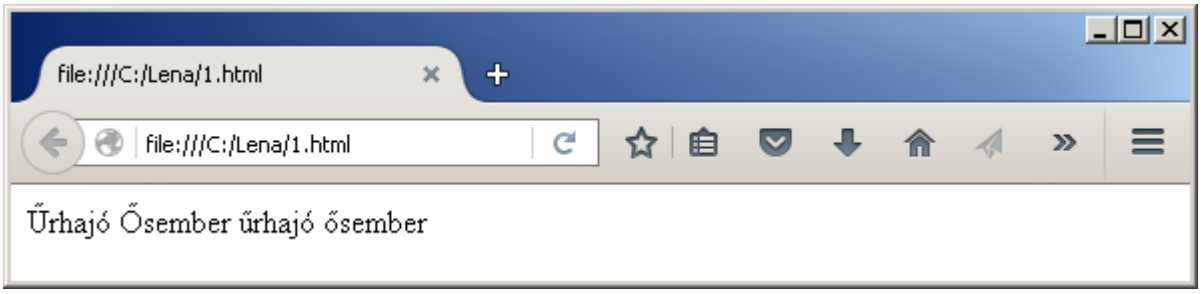
```
Űrhajó Ősember űrhajó ősember
```



Az ű és az ő betűk elég csúnyák, mert a böngésző azt hiszi, hogy nyugat-európai karakterkészletet használunk, amiben nincs ő és ű. Ezért meg kell mondani neki egy meta elemmel, hogy kelet-európai karakterkészletet szeretnénk használni. A kelet-európai karakterkészlet kódja: ISO-8859-2

A meta elemeket mindig a **head**-be írjuk, ezért kell csinálnunk egy **html**-t, egy **head**-et és egy **body**-t is.

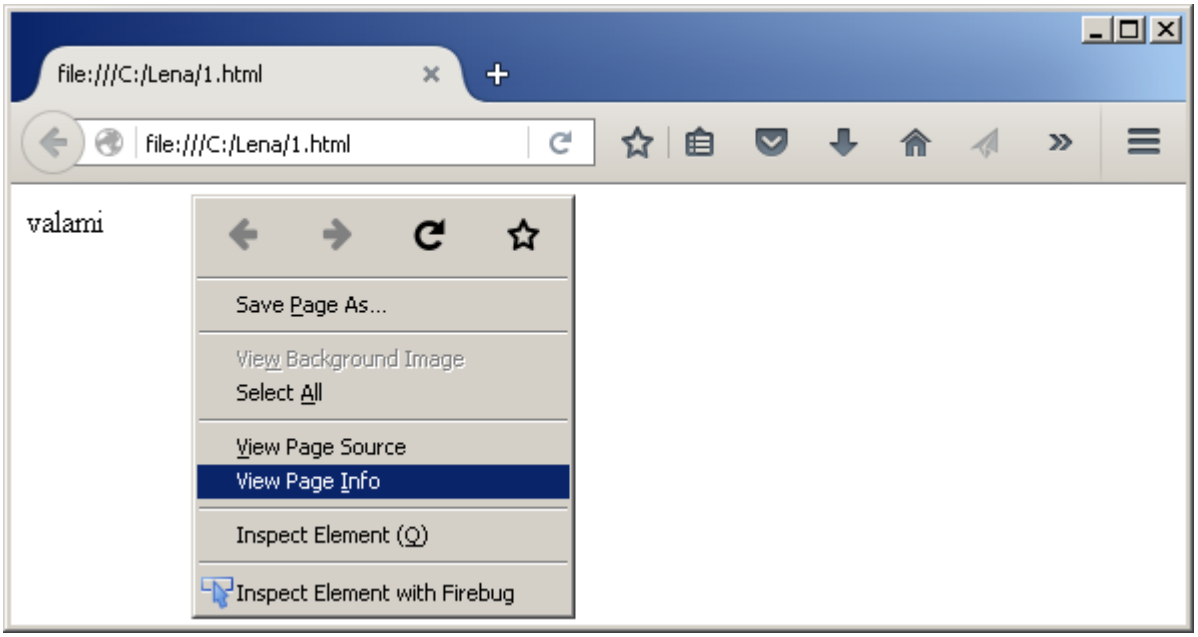
```
<html><head>
  <meta charset="ISO-8859-2">
</head><body>
  Űrhajó Ősember űrhajó ősember
</body></html>
```

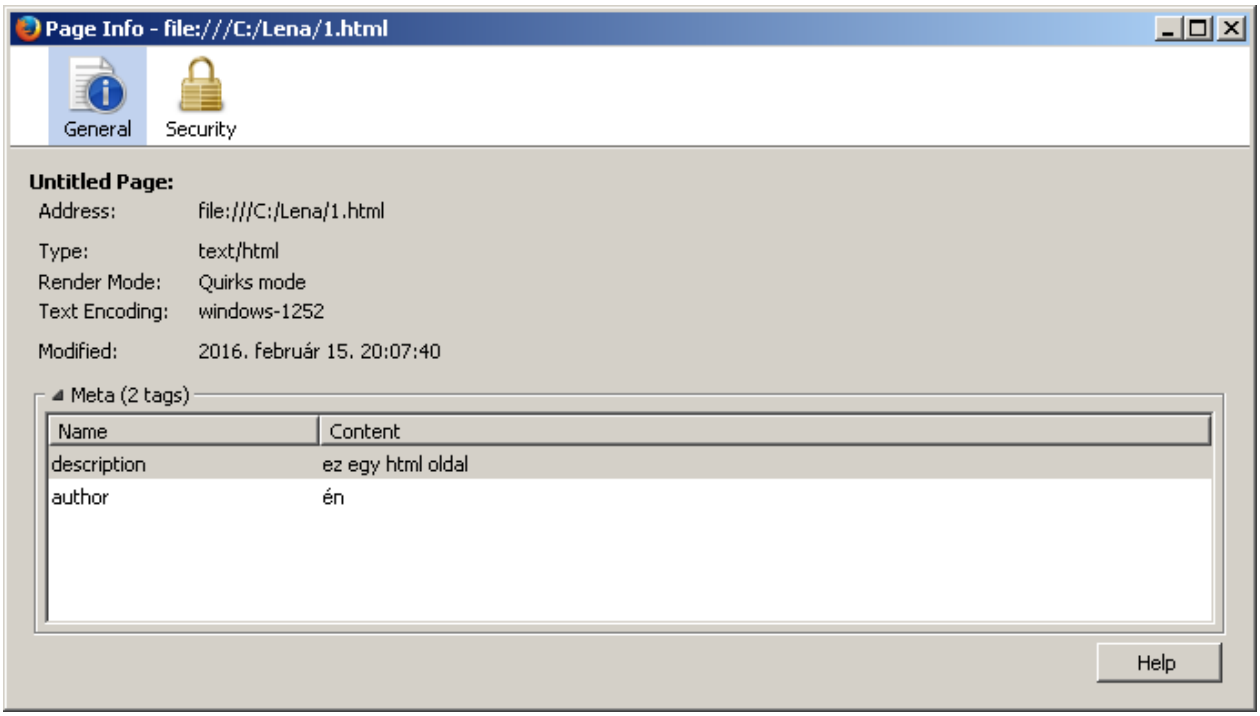


Meta elemek segítségével megadhatjuk a HTML fájl leírását (description), és a szerzőjét (author).

```
<html><head>
  <meta name="description" content="ez egy html oldal">
  <meta name="author" content="én">
</head><body>
  valami
</body></html>
```

Ez nem fog megjelenni a böngészőben. Ezt Firefox-ban úgy csalogathatjuk elő, ha valahol a nagy fehér semmi fölött nyomunk egy jobb egérklikket, és kiválasztjuk a ViewPage Info-t.





A következő meta elemhez két html fájlra lesz szükség.

1.html

```
<html><head>
   <meta http-equiv="refresh" content="5; url=2.html">
</head><body>
   <h1>Egy</h1>
</body></html>
```

2.html

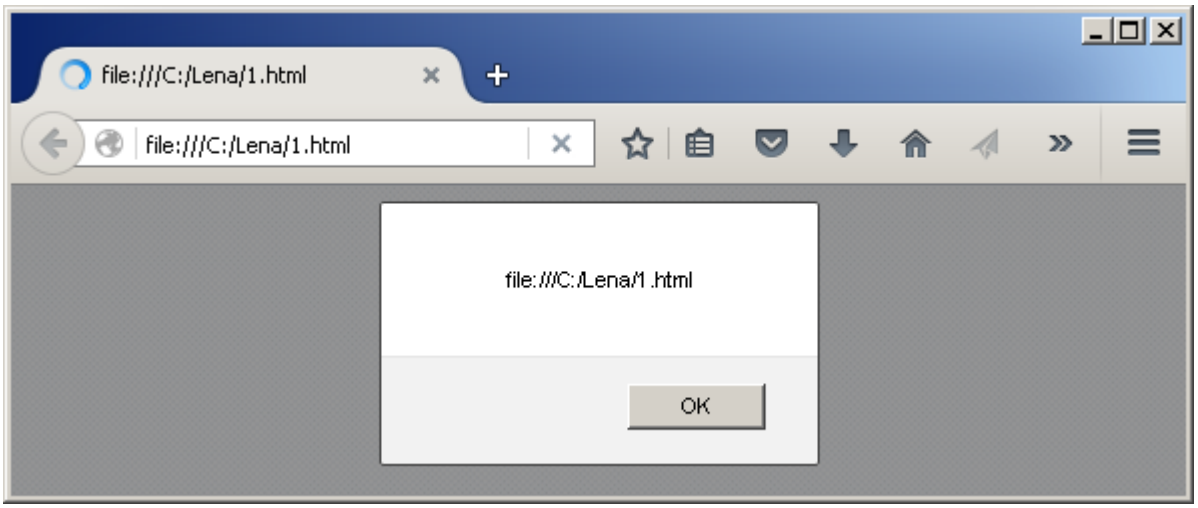
```
<h1>Kettő</h1>
```

Ha a böngészőben megnyitjuk az 1.html fájlt, a böngésző 5 másodperc múlva át fog váltani a 2.html-re.

JavaScript

A script elembe JavaScript programot írhatunk, ami az oldal megnyitásakor el is indul.

```
<script>
   alert(document.URL);
</script>
```



Az **alert** JavaScript paranccsal már találkoztunk, kiír valamit egy felugró ablakba.

A **document.URL** segítségével a HTML fájlunk útvonalát kérdezzük le.

JavaScript-ben már nem mindegy, hogy nagybetűvel vagy kisbetűvel írjuk a parancsokat.

2 CSS

MI AZ A CSS?

A CSS arra jó, hogy a HTML elemek megjelenítését, stílusát lehet vele beállítani. Például a betűk méretét és színét.

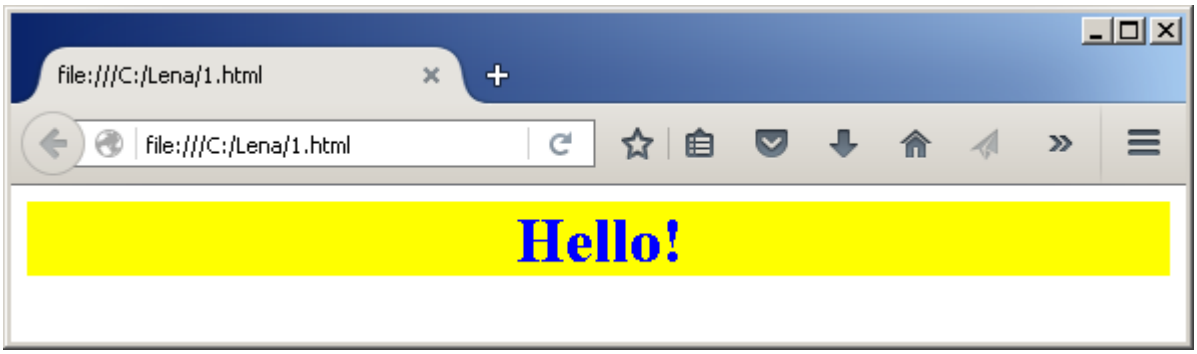
A CSS-t három féleképpen használhatjuk:

- Beleírhatjuk egy HTML elem **style** (kiejtés: sztájl) attribútumába. Ezt angolul úgy hívják, hogy inline (kiejtés: inlájn)
- Készíthetünk egy **style** elemet, és abba tesszük az összes CSS-t. Angolul: internal
- Külön fájlba tesszük az összes CSS-t, és egy **style** elembe beírjuk, hogy hol van ez a fájl. Angolul: external

CSS EGY HTML ELEM STYLE ATTRIBÚTUMÁBAN

Az alábbi példában egy **h1** HTML elem **style** attribútumába írjuk a CSS-t:

```
<h1 style="color: blue; background: yellow; text-align: center">Hello!</h1>
```



A CSS-ben a **color: blue** színezi a feliratot kékre, a **background: yellow** a hátteret sárgára, a **text-align: center** pedig középre helyezi a szöveget.

A tulajdonság neve és az értéke között kettőspont van. Ha több tulajdonságot állítunk be, akkor pontosvesszővel választjuk el őket egymástól.

CSS szaknyelven a tulajdonságot angolul declaration-nek, a nevet property-nek, az értéket value-nak hívják.

CSS EGY STYLE ELEMBEN

Az előző példa így néz ki, ha külön **style** elembe pakoljuk a CSS-t:

```
<html>
  <body>
    <h1>Hello!</h1>
    <style type="text/css">
      h1 { color: blue; background: yellow; text-align: center }
    </style>
  </body>
</html>
```

Magát a **style** elemet a body elem legaljára tettük, ami azért jó, mert így a böngésző gyorsabban fel tudja dolgozni a HTML fájlt.

A CSS kódba bekerült egy új rész: **h1 { }**, ami azt jelenti, hogy a kapcsos zárójelen belülre írt tulajdonságok az összes **h1** HTML elemre vonatkoznak a HTML fájlon belül.

CSS-ben angolul a kapcsos zárójel előtti részt, ami meghatározza, hogy melyik HTML elemekre vonatkozzon a stílus, selector (kiejtés: szelektor) -nak hívják.

CSS KÜLSŐ FÁJLBAN

A CSS-t pakolhatjuk külön css kiterjesztésű fájlba is. Ekkor a HTML így fog kinézni:

```
<html>
  <body>
    <h1>Hello!</h1>
    <link type="text/css" rel="stylesheet" href="hello.css">
  </body>
</html>
```

A CSS-t pedig egy hello.css fájlba tesszük. A **link** elem **href** attribútuma tartalmazza a CSS fájl útvonalát.

A hello.css fájl így fog kinézni:

```
h1 { color: blue; background: yellow; text-align: center }
```

HTML ELEMEK KIVÁLASZTÁSA

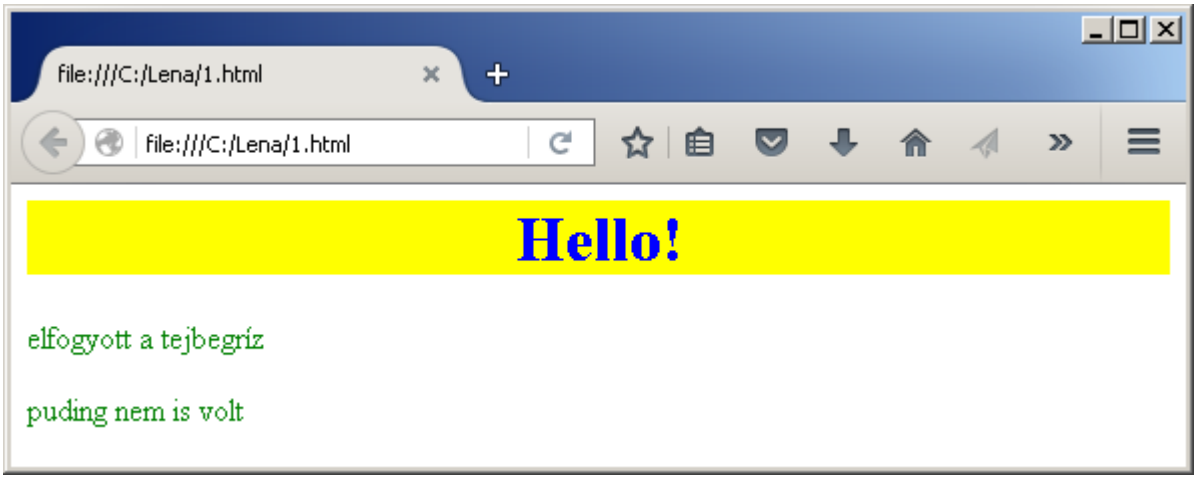
Azokat a HTML elemeket, amikre a CSS tulajdonságok vonatkoznak, többféleképp választhatjuk ki:

- az elem típusa alapján
- az elem **id** attribútuma alapján
- az elem **class** attribútuma alapján

HTML ELEMEK KIVÁLASZTÁSA TÍPUS ALAPJÁN

A kapcsos zárójel { elé beírjuk a HTML elem típusát, pl.: **p**. Az utána jövő kapcsos zárójelbe írt tulajdonságok az összes **p** elemre fognak vonatkozni.

```
<html>
  <body>
    <h1>Hello!</h1>
    <p>elfogyott a tejbegríz</p>
    <p>puding nem is volt</p>
    <style type="text/css">
      h1 { color: blue; background: yellow; text-align: center }
      p { color: green }
    </style>
  </body>
</html>
```



Vesszővel elválasztva több elemet is megadhatunk.

```
<html>
  <body>
    <h1>Hello!</h1>
    <p>elfogyott a tejbegríz</p>
    <p>puding nem is volt</p>
    <style type="text/css">
      h1, p { color: blue; background: yellow; text-align: center }
    </style>
  </body>
</html>
```

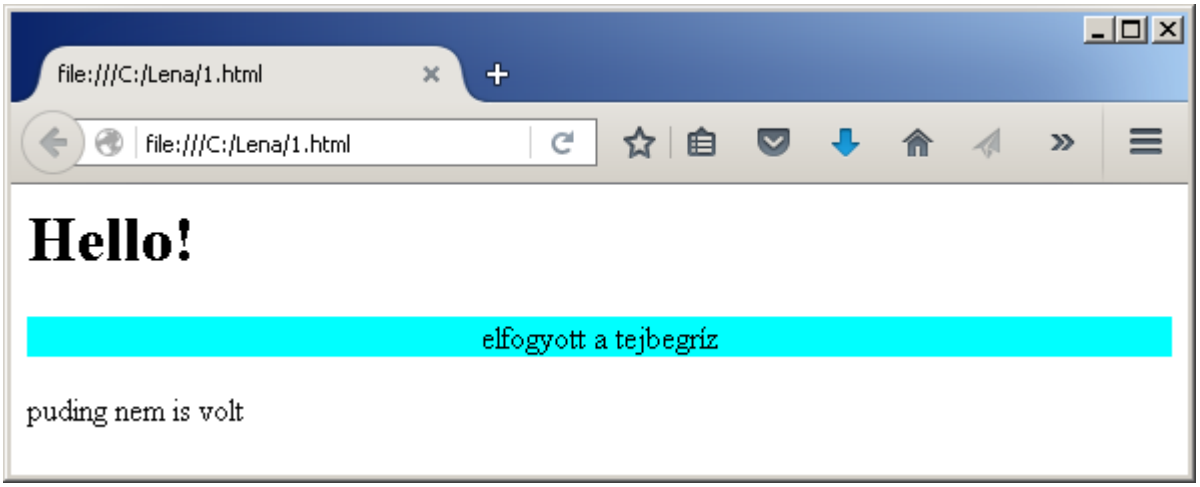
HTML ELEMEK KIVÁLASZTÁSA ID ALAPJÁN

A HTML elemeknek adhatunk egy **id** attribútumot, ami a HTML elem azonosítója lesz. Az **id** attribútum értékének egyedinek kell lennie, vagyis két HTML elemnek nem lehet ugyanaz az **id**-je.

A kettős kereszttel **#** adhatunk stílust egy **id**-vel kiválasztott elemnek.

```
<html>
  <body>
    <h1>Hello!</h1>
    <p id="p1">elfogyott a tejbegríz</p>
    <p>puding nem is volt</p>
```

```
    <style type="text/css">
      #p1 { background: cyan; text-align: center }
    </style>
  </body>
</html>
```



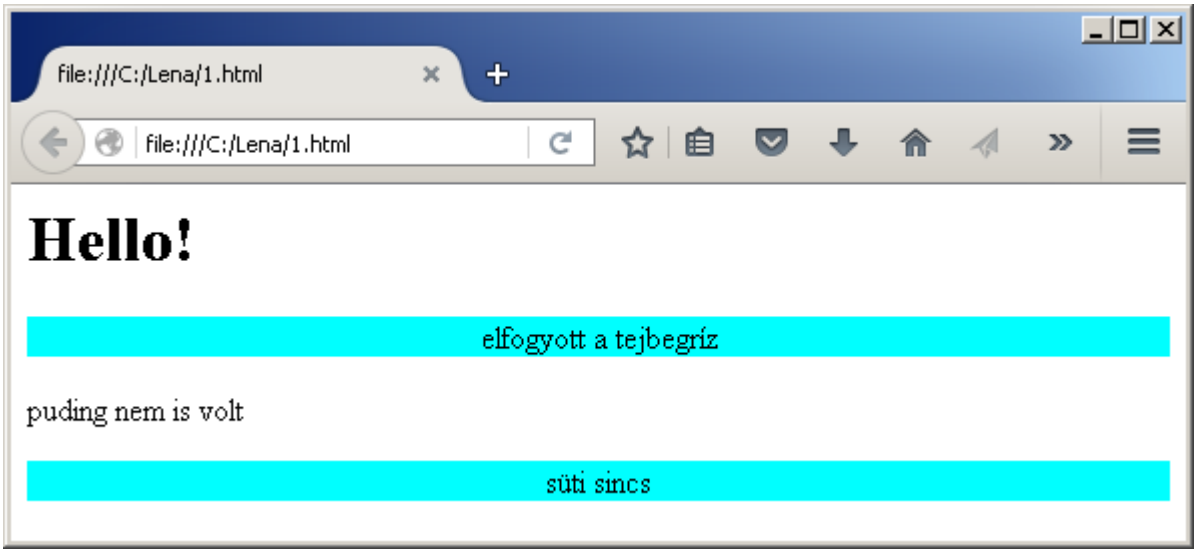
Az **id**-vel történő kiválasztást ritkán használjuk, mert egy nagy HTML fájlnál nehéz meggyőződni arról, hogy az id egyedi legyen.

HTML ELEMEK KIVÁLASZTÁSA CLASS ALAPJÁN

A HTML elemnek adhatunk egy **class** (kiejtés: klász) attribútumot, ami olyan, mint az **id**, de több HTML elemnek is adhatjuk ugyanazt az értéket.

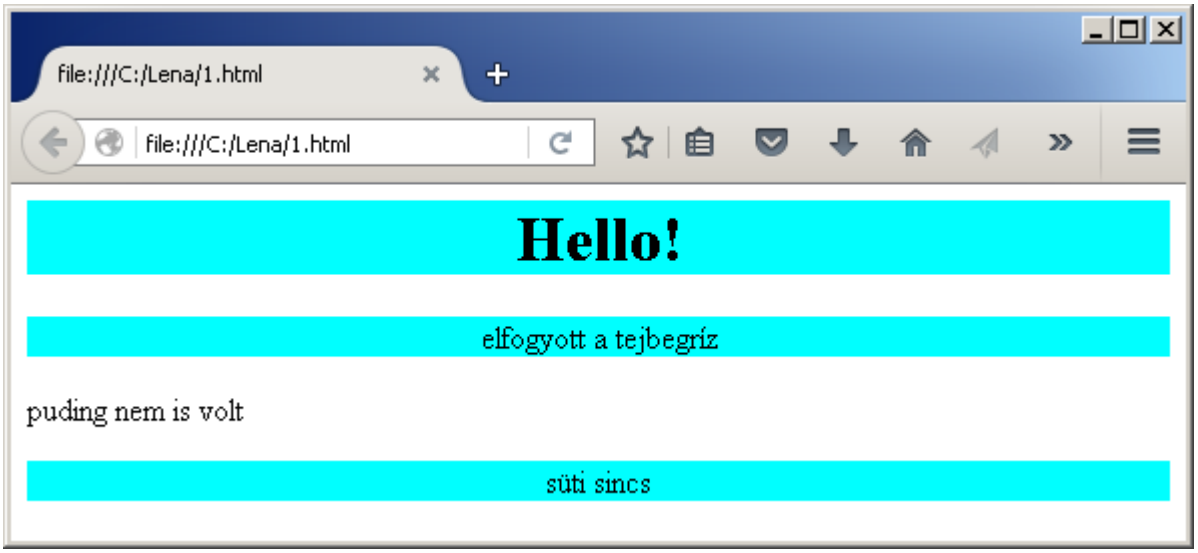
Ponttal adhatunk stílust a class-szal kiválasztott elemeknek.

```
<html>
  <body>
    <h1>Hello!</h1>
    <p class="c1">elfogyott a tejbegríz</p>
    <p>puding nem is volt</p>
    <p class="c1">süti sincs</p>
    <style type="text/css">
      .c1 { background: cyan; text-align: center }
    </style>
  </body>
</html>
```



Vesszővel elválasztva több **class**-t is megadhatunk a CSS-ben, sőt keverhetjük is elemtípussal és **id**-vel.

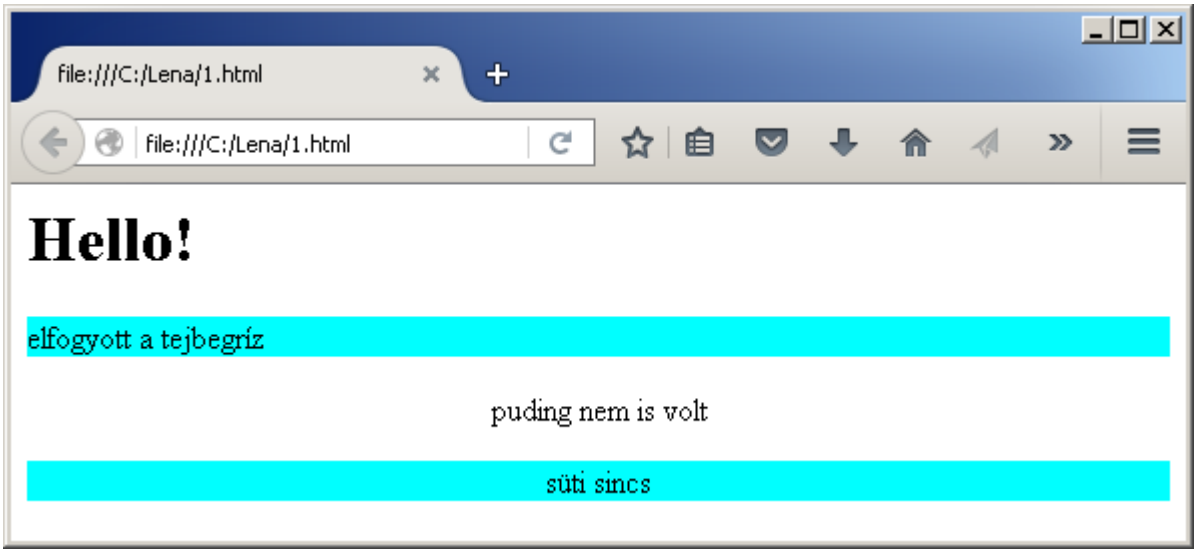
```
<html>
  <body>
    <h1>Hello!</h1>
    <p class="c1">elfogyott a tejbegríz</p>
    <p>puding nem is volt</p>
    <p id="i1">süti sincs</p>
    <style type="text/css">
      .c1, h1, #i1 { background: cyan; text-align: center }
    </style>
  </body>
</html>
```



Fordítva is működik, HTML-ben egy elemhez több **class**-t is felvehetünk a **class** attribútumban. Ott viszont üres hellyel (space) kell elválasztani őket.

```
<html>
  <body>
    <h1>Hello!</h1>
    <p class="c1">elfogyott a tejbegríz</p>
```

```
    <p class="c2">puding nem is volt</p>
    <p class="c1 c2">süti sincs</p>
    <style type="text/css">
      .c1 { background: cyan }
      .c2 { text-align: center }
    </style>
  </body>
</html>
```



A HTML-ben és a CSS-ben általában nem számít, hogy kisbetűvel vagy nagybetűvel írunk valamit, viszont az **id** és **class** attribútumokban megadott azonosítóknál, pl. c1 számít.

Olyat is le lehet írni, hogy azokat az elemeket válasszuk ki, amiknek a típusa **p**, a **class**-a pedig c1:

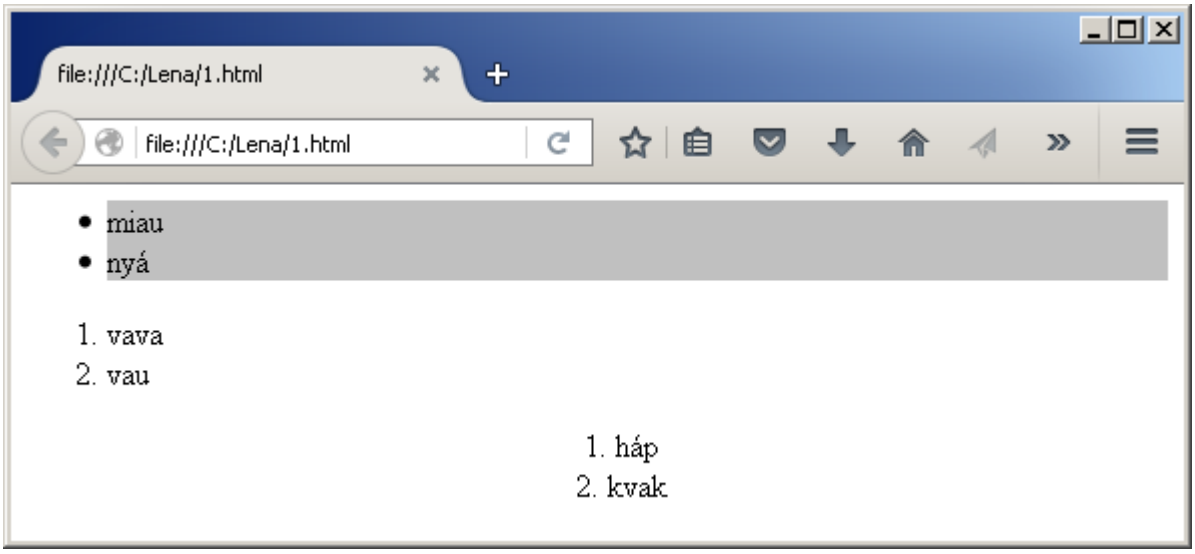
```
p.c1 { background: cyan }
```

EGYMÁSBA ÁGYAZOTT HTML ELEMEK KIVÁLASZTÁSA

Néha szeretnénk, ha egy HTML elem kinézete attól függene, hogy milyen külső HTML elemben van benne. Például azt akarjuk, hogy a **li** elemnek ezüstszürke háttere legyen, ha egy **ul** elemben van, viszont maradjon a háttere fehér, ha **ol** elemben van.

```
<html>
  <body>
    <ul>
      <li>miau
      <li>nyá
    </ul>
    <ol>
      <li>vava
      <li>vau
    </ol>
    <ol class="kacsa">
      <li>háp
      <li>kvak
    </ol>
    <style type="text/css">
```

```
        ul li { background: silver; }
        ol.kacsa li { text-align: center }
      </style>
    </body>
</html>
```

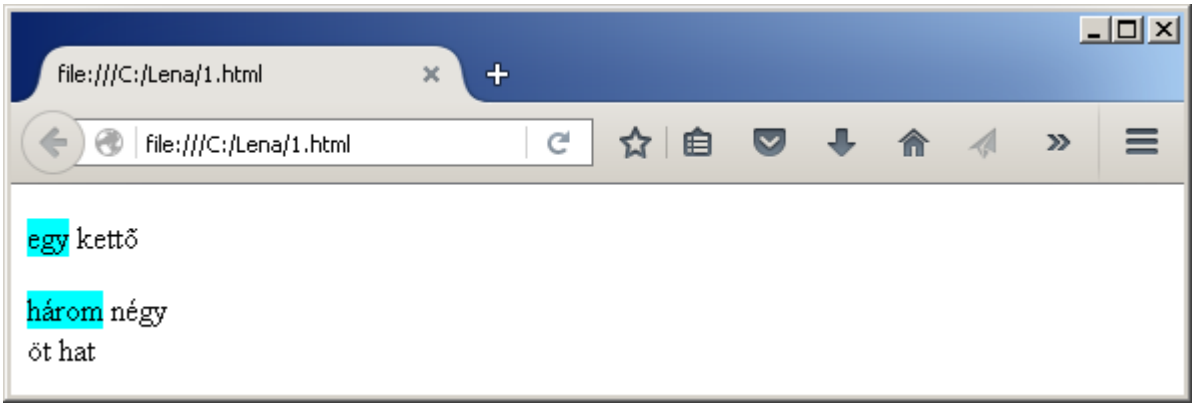


Az **ul li** azt jelenti, hogy azok a **li** elemek, amik egy **ul** elemben vannak benne.

Az **ol.kacsa li** pedig azt jelenti, hogy azok a **li** elemek, amik egy olyan **ol** elemben vannak, aminek a **class**-a kacsa.

A keresés akkor is működik, ha a külső HTML elem és a belső HTML elem között más elem is van. A következő példában például egy **p** elem van a **div** és a **span** között.

```
<html>
  <body>
    <div>
      <p>
        <span>egy</span> kettő
      </p>
      <span>három</span> négy
    </div>
    <span>öt</span> hat
    <style type="text/css">
      div span { background: cyan }
    </style>
  </body>
</html>
```



MINDEN HTML ELEM KIVÁLASZTÁSA

Ha minden HTML elemet ki akarunk választani, azt a csillaggal * tehetjük meg.

```
<html>
   <body>
      <p>maci kaki</p>
      tigris kaki
      <h1>teve kaki</h1>
      <style type="text/css">
         * { text-align: center }
      </style>
   </body>
</html>
```

Ez a *-os kiválasztási mód eléggé megterheli a böngészőt, ezért kerülendő, helyette inkább a **body { … }** a javasolt, mivel a belső elemek öröklik a külső elemek stílusát.

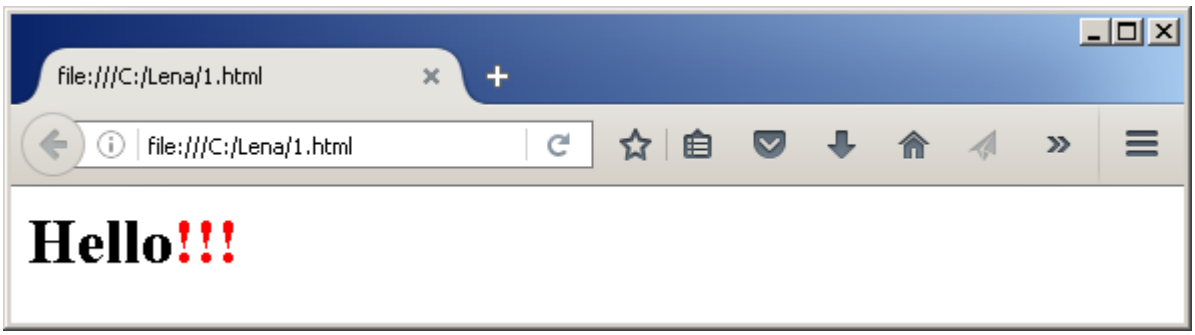
```
<html>
   <body>
      <p>maci kaki</p>
      tigris kaki
      <h1>teve kaki</h1>
      <style type="text/css">
         body { text-align: center }
      </style>
```

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

SZÖVEG GENERÁLÁSA

A **::before** segítségével egy kiválasztott elem elé, az **::after** segítségével pedig mögé generálhatunk szöveget. Az alábbi kód minden **h1** elem végéhez hozzáad három piros felkiáltójelet.

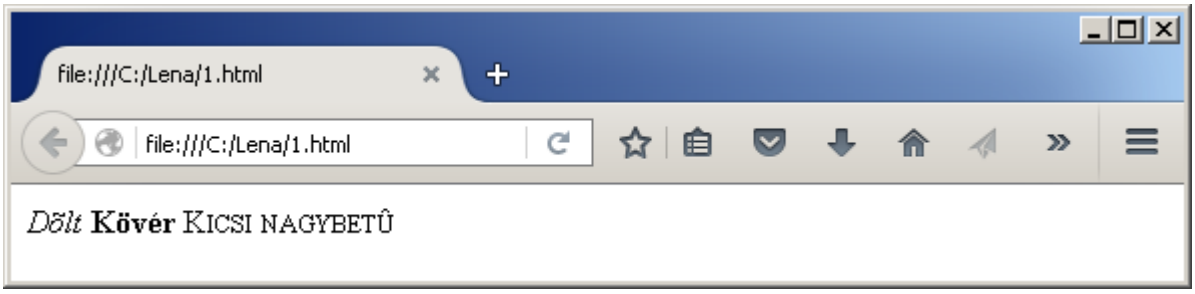
```
<html>
  <body>
    <h1>Hello</h1>
    <style type="text/css">
      h1::after { content: "!!!"; color: red }
    </style>
  </body>
</html>
```



SZÖVEG FORMÁZÁSA

Dőlt betűket a **font-style**, kövér betűket a **font-weight** segítségével készíthetünk. A **font-variant** segítségével a kisbetűkből kicsi nagybetűket csinálhatunk, a nagybetűk nem változnak.

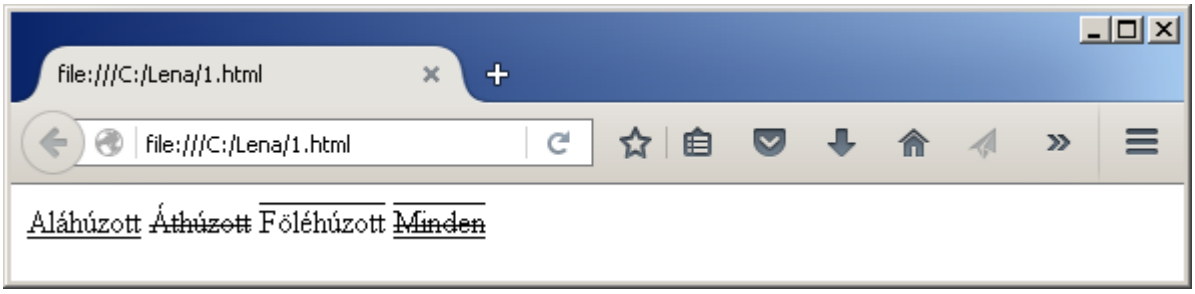
```
<html>
  <body>
    <span class="dolt">Dőlt</span>
    <span class="kover">Kövér</span>
    <span class="kicsinagybetu">Kicsi nagybetű</span>
    <style type="text/css">
      .dolt { font-style: italic }
      .kover { font-weight: bold }
      .kicsinagybetu { font-variant: small-caps }
    </style>
  </body>
</html>
```



A **font-style**, **font-weight**, **font-variant** értéke lehet **normal**, ha ki akarjuk hangsúlyozni, hogy ne legyen dőlt, kövér vagy kicsi nagybetű.

A **text-decoration** segítségével aláhúzhatjuk, áthúzhatjuk vagy föléhúzhatjuk a szöveget. Akár egyszerre is.

```
<html>
  <body>
    <span class="alahuzott">Aláhúzott</span>
    <span class="athuzott">Áthúzott</span>
    <span class="folehuzott">Föléhúzott</span>
    <span class="minden">Minden</span>
    <style type="text/css">
      .alahuzott { text-decoration: underline }
      .athuzott { text-decoration: line-through }
      .folehuzott { text-decoration: overline }
      .minden { text-decoration: underline line-through overline }
    </style>
  </body>
</html>
```

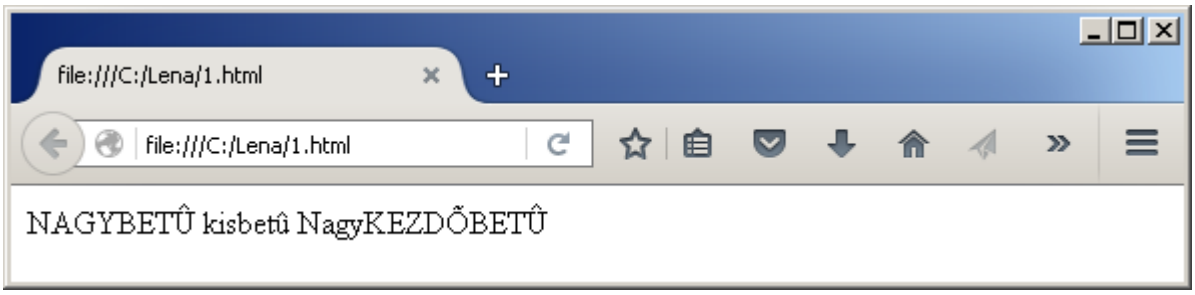


Ha ki akarjuk hangsúlyozni, hogy ne legyen áthúzva, a **none** értéket kell adni a **text-decoration**-nek.

A **text-transform** csupa nagybetűssé, kisbetűssé vagy nagy kezdőbetűssé teszi a szöveget.

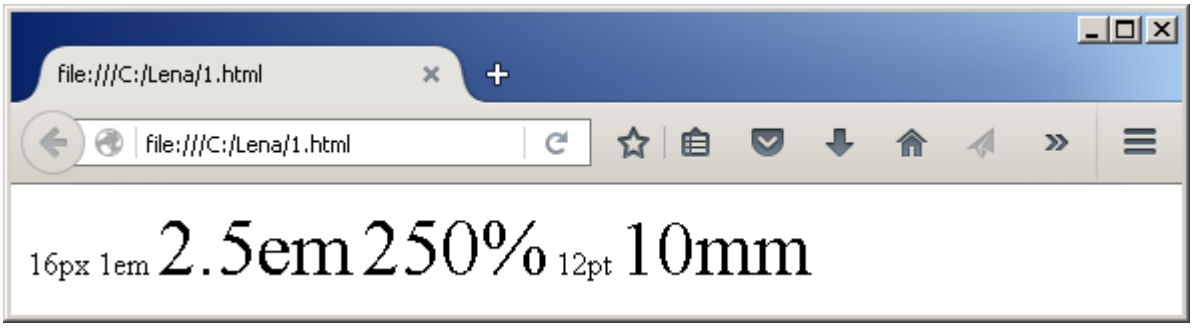
```
<html>
  <body>
    <span class="nagybetu">nagyBETŰ</span>
    <span class="kisbetu">kisBETŰ</span>
    <span class="nagykezdobetu">nagyKEZDŐBETŰ</span>
    <style type="text/css">
      .nagybetu { text-transform: uppercase }
      .kisbetu { text-transform: lowercase }
      .nagykezdobetu { text-transform: capitalize }
    </style>
  </body>
```

```
</html>
```



A betűk méretét a **font-size** tulajdonság segítségével állíthatjuk be.

```
<html>
  <body>
    <span class="c1">16px</span>
    <span class="c2">1em</span>
    <span class="c3">2.5em</span>
    <span class="c4">250%</span>
    <span class="c5">12pt</span>
    <span class="c6">10mm</span>
    <span class="c7">0</span>
    <style type="text/css">
      span.c1 { font-size: 16px }
      span.c2 { font-size: 1em }
      span.c3 { font-size: 2.5em }
      span.c4 { font-size: 250% }
      span.c5 { font-size: 12pt }
      span.c6 { font-size: 10mm }
      span.c7 { font-size: 0 }
    </style>
  </body>
</html>
```



Mint látszik, a betűméretnek több mértékegysége is lehet. A **px** képernyőpontot jelent, angolul pixel. A böngészők többségénél **16 px** az alapértelmezett betűméret. Az **em** azt jelenti, hogy az alapértelmezett betűméret hányszorosa legyen a betűméret; előnye, hogy ha a felhasználó nagyobbra állítja az alapértelmezett betűméretet, az **em**-mel megadott betűméret is nagyobb lesz. A százalékos **%** megadás ugyanúgy működik, mint az **em**. A **pt**-t nyomdászok használják. Ezen kívül használhatunk **cm**-t és **mm**-t.

Ha bármilyen méretnek nullát **0** adunk meg, akkor nem szoktunk mértékegységet írni.

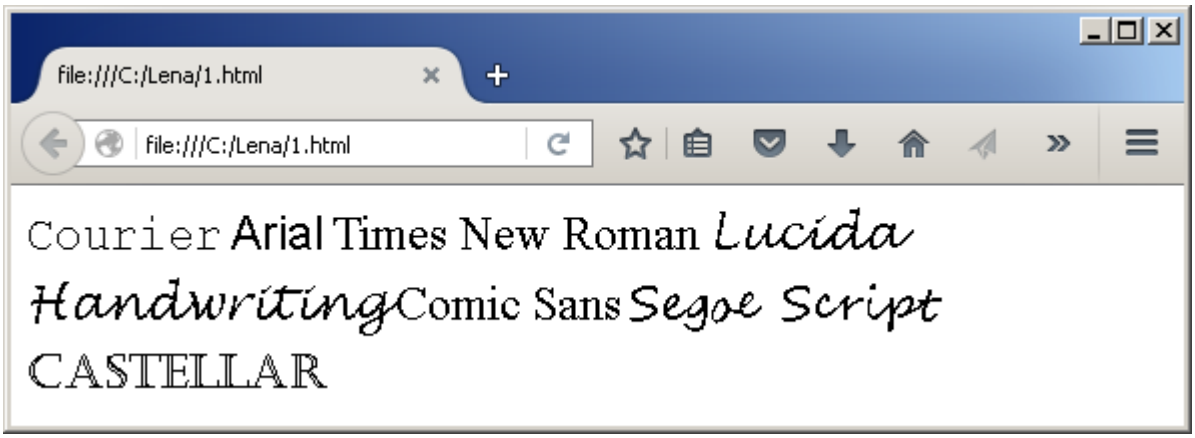
A betű típusát a **font-family** tulajdonsággal állíthatjuk be. Öt alaptípusa van: **serif** (talpas betűk), **sans-serif** (talpnélküli betűk), **monospace** (minden betű egyforma széles), **cursive** (kézírás), **fantasy** (vicces betűk).

```
<html>
  <body>
    <span class="s">serif</span>
    <span class="ss">sans-serif</span>
    <span class="m">monospace</span>
    <span class="c">cursive</span>
    <span class="f">fantasy</span>
    <style type="text/css">
      span { font-size: 1.5em }
      .s { font-family: serif }
      .ss { font-family: sans-serif }
      .m { font-family: monospace }
      .c { font-family: cursive }
      .f { font-family: fantasy }
    </style>
  </body>
</html>
```

Az alaptípusokat minden böngésző támogatja. De az alaptípusokon kívül számtalan típus van még. Ha nem alaptípust használunk, azt macskakörömbe szokás tenni.

```
<html>
  <body>
    <span class="c">Courier</span>
    <span class="a">Arial</span>
    <span class="tms">Times New Roman</span>
    <span class="lh">Lucida Handwriting</span>
    <span class="cs">Comic Sans</span>
    <span class="ss">Segoe Script</span>
    <span class="c2">Castellar</span>
    <style type="text/css">
      span { font-size: 1.5em }
      .c { font-family: "Courier" }
      .a { font-family: "Arial" }
      .tms { font-family: "Times New Roman" }
      .lh { font-family: "Lucida Handwriting" }
      .cs { font-family: "Comic Sans" }
      .ss { font-family: "Segoe Script" }
      .c2 { font-family: "Castellar" }
```

```
    </style>
  </body>
</html>
```



A betűk stílusát rövidített stílusban, egy sorban is megadhatjuk a **font** tulajdonsággal. Ilyenkor először megmondjuk, hogy kövér vagy dőlt legyen-e, aztán a betűméretet, végül a betűtípust. A betűtípusnál a **"Courier", monospace** azt jelenti, hogy legyen a betűtípus **Courier**, de ha azt a böngésző nem ismeri, akkor legyen a **monospace** alaptípus.

```
<html>
  <body>
    <p class="c1">bold italic 20px "Courier", monospace</p>
    <p class="c2">2em "Castellar", fantasy</p>
    <style type="text/css">
      .c1 { font: bold italic 20px "Courier", monospace }
      .c2 { font: 2em "Castellar", fantasy }
    </style>
  </body>
</html>
```

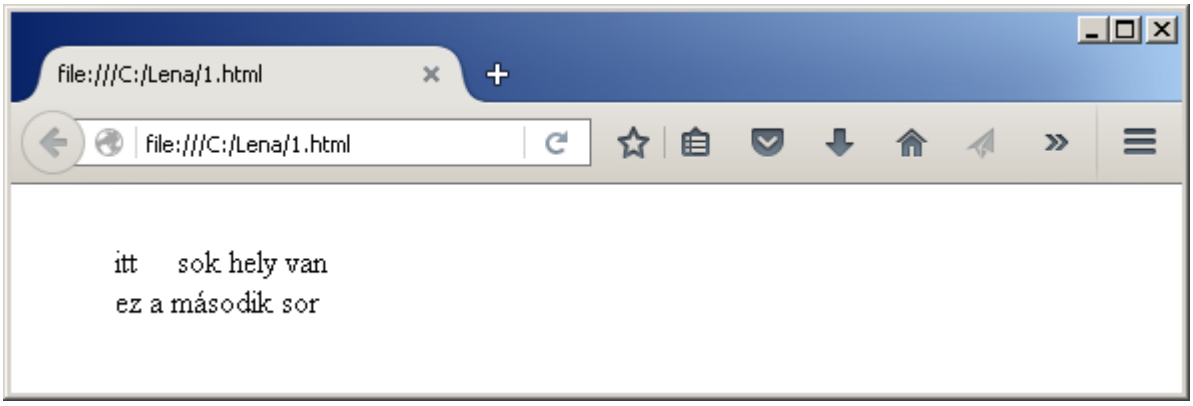
A betűk távolságát a **letter-spacing**, a szavak távolságát a **word-spacing**, a sorok közti távolságot pedig a **line-height** segítségével állíthatjuk be.

```
<html>
  <body>
```

```
    <p>normál szöveg<br>ez a második sor<p>
    <p class="c1">letter-spacing: 5px<br>ez a második sor</p>
    <p class="c2">word-spacing: 25px<br>ez a második sor</p>
    <p class="c3">line-height: 10px<br>ez a második sor</p>
    <style type="text/css">
      .c1 { letter-spacing: 5px }
      .c2 { word-spacing: 25px }
      .c3 { line-height: 10px }
    </style>
  </body>
</html>
```

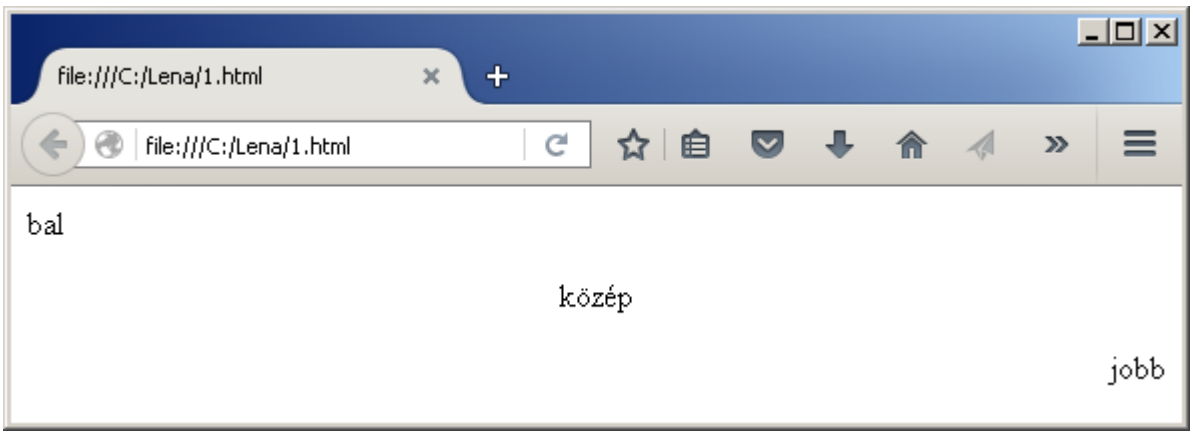
A **pre** HTML elemhez hasonló hatást érhetünk el, ha a **white-space: pre** stílust használjuk. Vagyis a szöveg megtartja az üres helyeket és a soremeléseket.

```
<html>
  <body>
    <p>
      itt    sok hely van
      ez a második sor
    </p>
    <style type="text/css">
      p { white-space: pre }
    </style>
  </body>
</html>
```



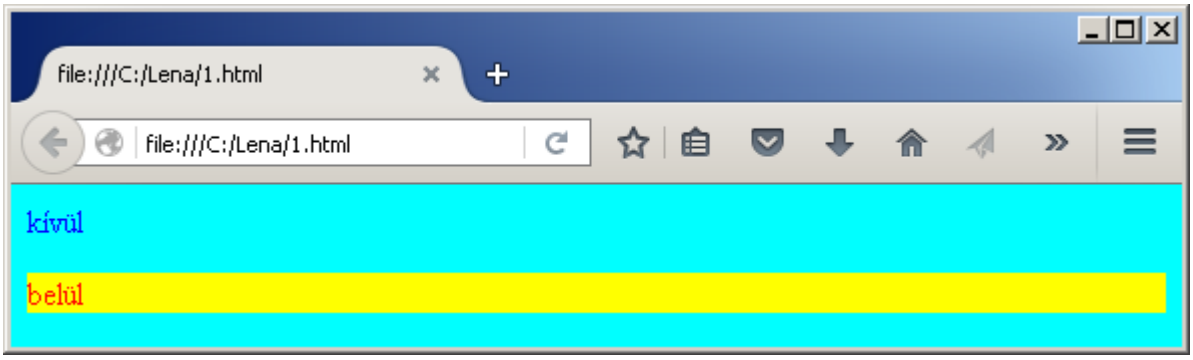
Ha egy szöveget balra, középre vagy jobbra akarunk rendezni, azt a **text-align** tulajdonsággal tehetjük meg. A **text-align** mindig a befoglaló elemhez képest igazítja jobbra, balra vagy középre a szöveget. Az alábbi példában a **body** a befoglaló elem.

```
<html>
  <body>
    <p style="text-align: left">bal</p>
    <p style="text-align: center">közép</p>
    <p style="text-align: right">jobb</p>
  </body>
</html>
```



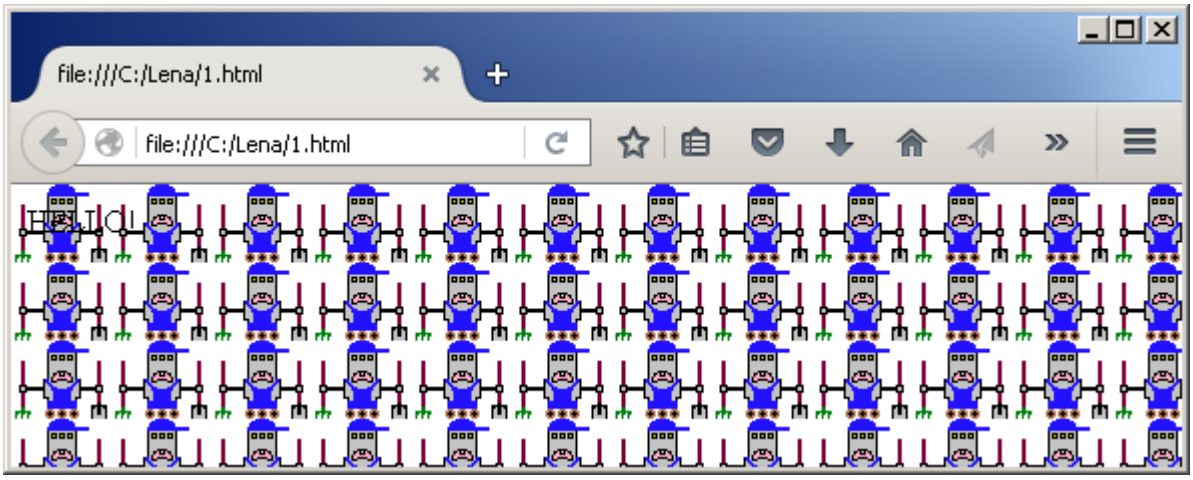
A **color** tulajdonság állítja be a betűszínt, a **background-color** pedig a háttérszínt. A **background-color** helyett használhatunk egyszerűen **background**-ot is.

```
<html>
  <body>
    kívül
    <p>belül</p>
    <style type="text/css">
      body { color: blue; background: cyan}
      p { color: red; background-color: yellow }
    </style>
  </body>
</html>
```



Háttérszín helyett háttérképet is beállíthatunk a **background-image** segítségével. Értéknek URL-t is adhatunk távoli kép esetben, vagy simán beírhatjuk a fájl nevét, ha a képfájl a HTML fájllal egy helyen van.

```
<html>
  <body>
    <p>HELLO!</p>
    <style type="text/css">
      body { background-image: url('http://topskills.eu/kepek/robot-take-your-job.png') }
    </style>
  </body>
</html>
```



Ha a kép a HTML fájllal egy helyen van, az így néz ki:

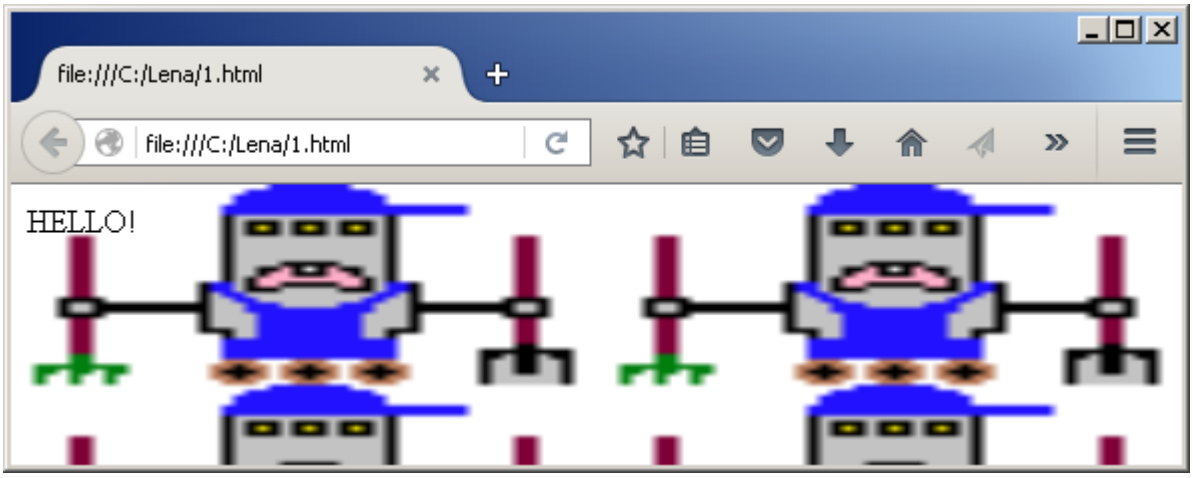
```
body { background-image: url('robot-take-your-job.png') }
```

Háttérképet tipikusan a **body** vagy egy **div** elemnek szoktak beállítani, de más olyan HTML elemnél is működik, ami új sort kezd magának. Feladat: próbáld ki, hogy a **body** helyett a **p** elemnek állítasz be háttérképet.

A **background-size** segítségével átállíthatjuk a kép méretét. Az alábbi példában azt mondjuk, hogy a kép a rendelkezésre álló szélesség 50%-át, vagyis felét foglalja el, a magassága pedig 100 pont legyen.

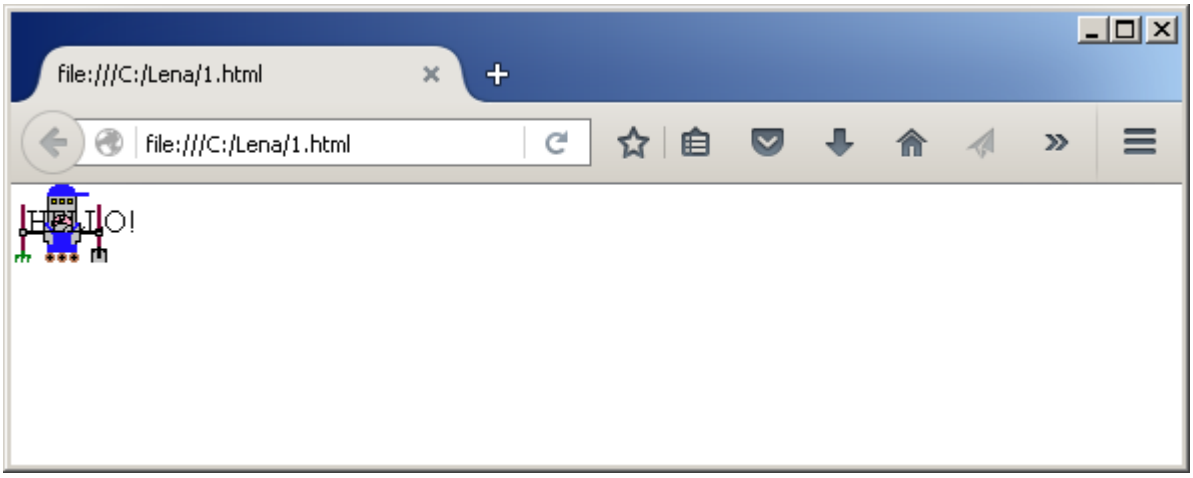
```
<html>
  <body>
    <p>HELLO!</p>
    <style type="text/css">
      body {
        background-image: url('http://topskills.eu/kepek/robot-take-your-job.png');
```

```
            background-size: 50% 100px
        }
    </style>
  </body>
</html>
```



A **background-repeat** segítségével állíthatjuk be, hogy ismétlődjön-e a kép. A **no-repeat** kikapcsolja az ismétlést. További lehetséges értékek: **repeat**, **repeat-x**, **repeat-y**.

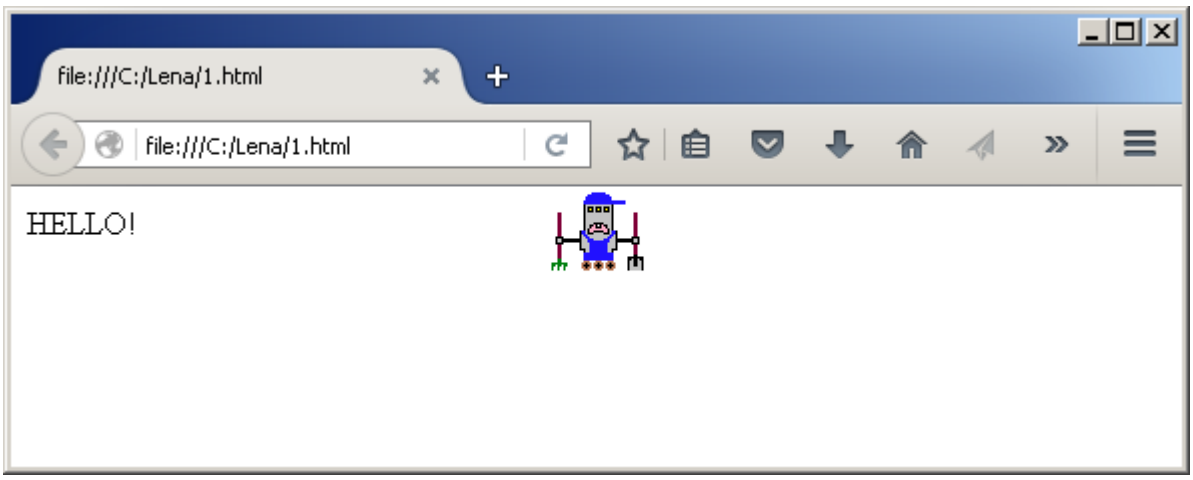
```
<html>
  <body>
    <p>HELLO!</p>
    <style type="text/css">
      body {
        background-image: url('http://topskills.eu/kepek/robot-take-your-job.png');
        background-repeat: no-repeat
      }
    </style>
  </body>
</html>
```



Ha már kikapcsoltuk az ismétlést, az egy szál képet **a background-position**-nal tudjuk jobbra: **right**, balra: **left**, vagy középre: **center** rendezni.

```
<html>
```

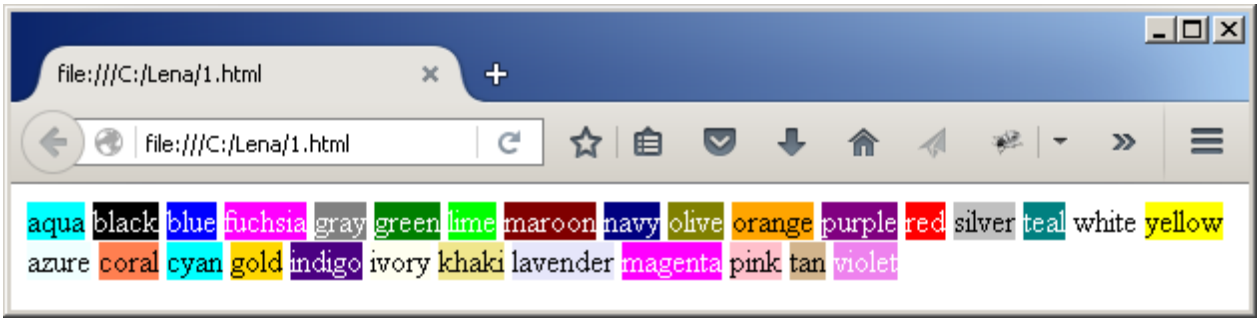
```
  <body>
    <p>HELLO!</p>
    <style type="text/css">
      body {
        background-image: url('http://topskills.eu/kepek/robot-take-your-job.png');
        background-repeat: no-repeat;
        background-position: center
      }
    </style>
  </body>
</html>
```



Színek

Van 17 szín, amit a CSS szabvány szerint minden böngészőnek ismernie kell: **aqua**, **black**, **blue**, **fuchsia**, **gray**, **green**, **lime**, **maroon**, **navy**, **olive**, **orange**, **purple**, **silver**, **red**, **teal**, **white**, **yellow**. De ennél általában több színt ismernek.

```
<span style="background:aqua; color:black">aqua</span>
<span style="background:black; color:white">black</span>
<span style="background:blue; color:white">blue</span>
<span style="background:fuchsia; color:white">fuchsia</span>
...
```



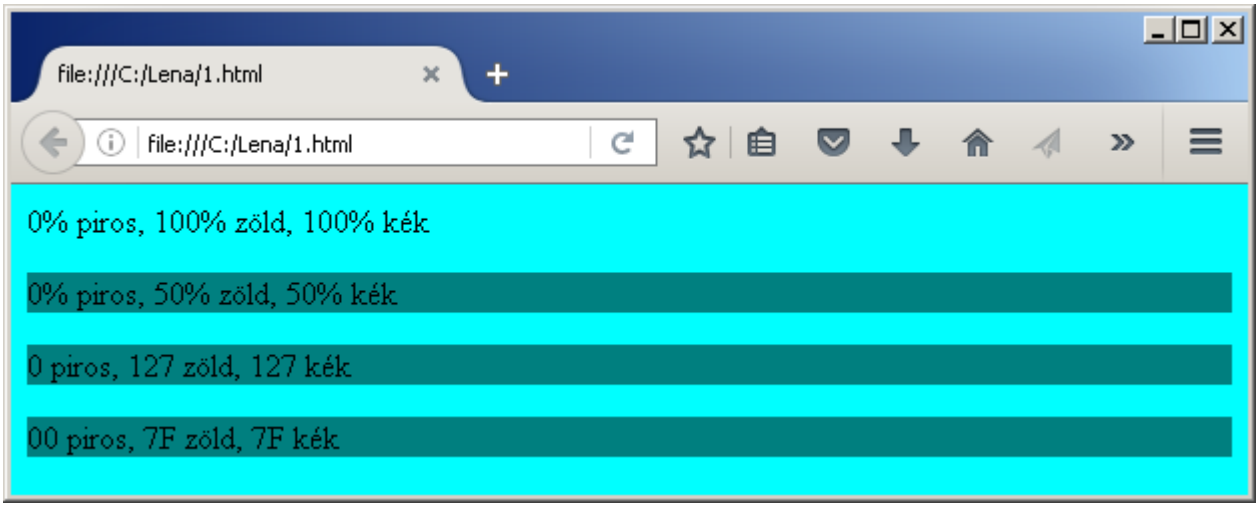
A számítógép a színeket három alapszínből rakja össze: piros (red), zöld (green), kék (blue). Megadhatjuk, hogy melyik alapszínből mennyit adjon egy színhez a számítógép. Ha például pirosból többet adunk, akkor pirosabb színt kapunk. Ha minden színből többet adunk, akkor pedig világosabb színt kapunk. Az **rgb** (red-green-blue) függvénynek megadhatjuk százalékosan is, hogy melyik színből mennyit keverjen.

Ekkor a három alapszínnek egy 0% és 100% közötti értéket adunk meg. Az **rgb(0%, 0%, 0%)** a fekete, az **rgb(100%, 100%, 100%)** a fehér, az **rgb(100%, 0%, 0%)** a piros.

Százalék helyett három 0-tól 255-ig terjedő számot is megadhatunk.

Vagy kihagyhatjuk az **rgb** függvényt és három tizenhatos számrendszerbeli számot is megadhatunk. Ekkor kettős keresztet kell a szám elé írni.

```
<html>
 <body style="background: rgb(0%, 100%, 100%)">
  0% piros, 100% zöld, 100% kék
  <p style="background: rgb(0%, 50%, 50%)">0% piros, 50% zöld, 50% kék</p>
  <p style="background: rgb(0, 127, 127)">0 piros, 127 zöld, 127 kék</p>
  <p style="background: #007F7F">00 piros, 7F zöld, 7F kék</p>
 </body>
</html>
```



A tizenhatos számrendszerbeli számokat hexadecimális számoknak is szokták hívni. Így kell nullától harminckettőig számolni tizenhatos számrendszerben:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20

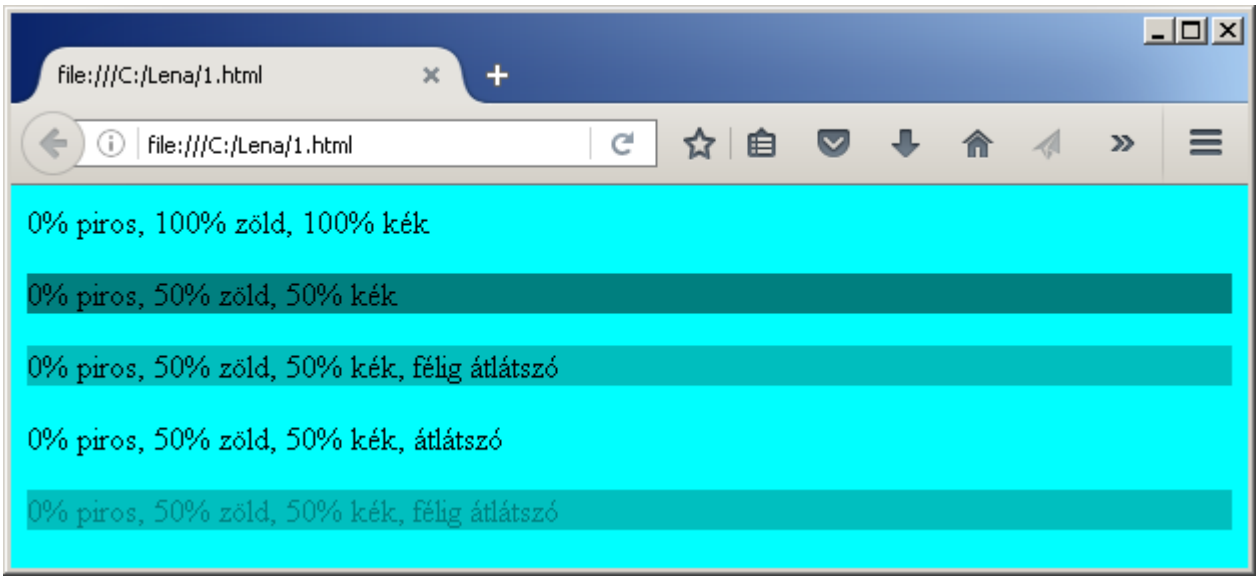
A 255 például tizenhatos számrendszerben így néz ki: FF.

Átlátszó színt az **rgba** függvénnyel tudunk csinálni, ennek adunk egy negyedik paramétert is, ami megmondja, hogy mennyire legyen átlátszó a szín: 0 =teljesen átlátszó, 0.5 = félig átlátszó, 1 = nem átlátszó.

Az **rgba** helyett használhatjuk az **opacity** tulajdonságot is. Annyi a különbség, hogy az **opacity** minden színre vonatkozik, ami az elemen belül van.

```
<html>
 <body style="background: rgb(0%, 100%, 100%)">
  0% piros, 100% zöld, 100% kék
  <p style="background: rgb(0, 127, 127)">0% piros, 50% zöld, 50% kék</p>
  <p style="background: rgba(0, 127, 127, 0.5)">0% piros, 50% zöld, 50% kék, félig átlátszó</p>
  <p style="background: rgba(0, 127, 127, 0)">0% piros, 50% zöld, 50% kék, átlátszó</p>
  <p style="background: rgb(0, 127, 127); opacity: 0.5">0% piros, 50% zöld, 50% kék, félig
átlátszó</p>
```

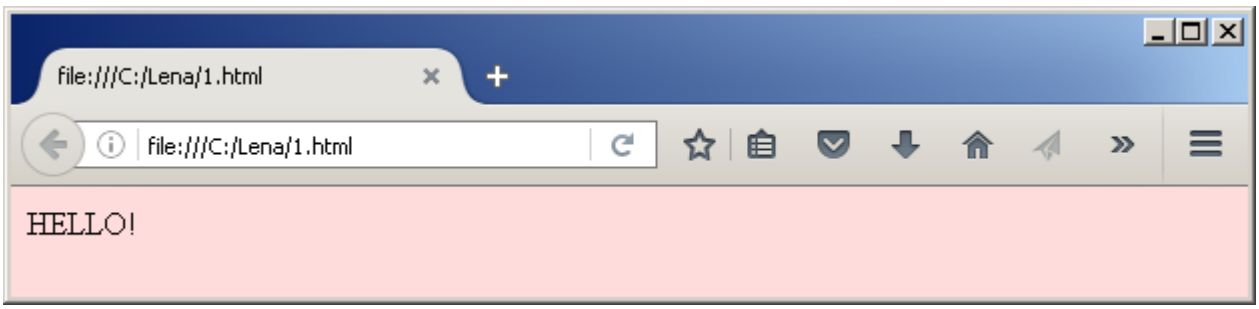
```
  </body>
</html>
```



KOMMENT

CSS-ben amit /* és */ közé teszünk, az olyan mintha ott se lenne.

```
<html>
  <body>
    <p>HELLO!</p>
    <style type="text/css">
      /* A body elem színeit adom itt meg */
      body {
        background: rgb(255, 220, 220);
  /*      color: rgb(255, 0, 0) */
      }
    </style>
  </body>
</html>
```

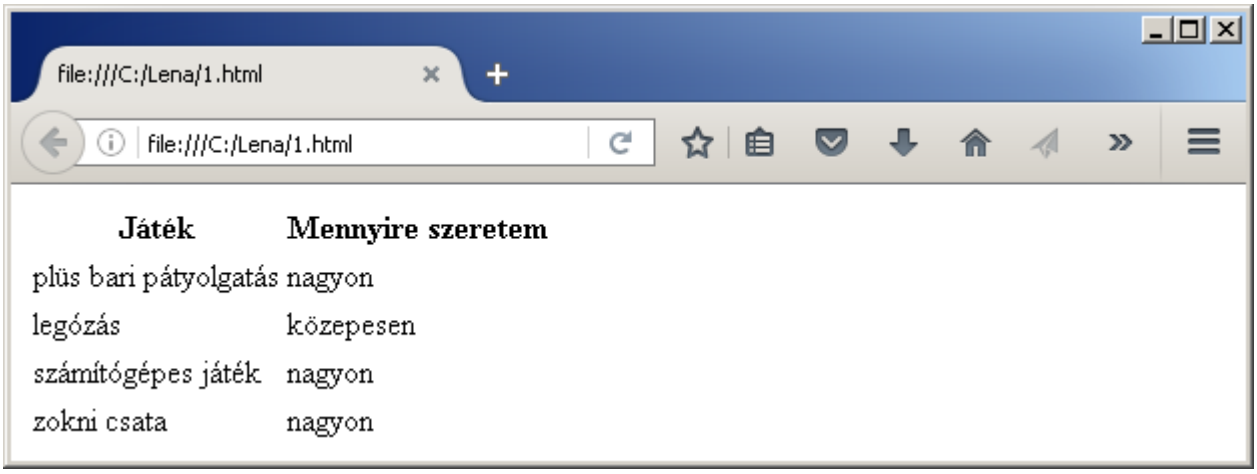


Ebben a példában a piros betűszínt kommenteztük ki, de a rózsaszín hátteret meghagytuk. A komment arra is jó, ha valami emberi nyelven írt megjegyzést akarunk beilleszteni a CSS-be.

TÁBLÁZAT

Először csináljunk egy HTML táblázatot CSS formázás nélkül.

```
<html>
  <body>
    <table>
      <tr><th>Játék</th><th>Mennyire szeretem</th></tr>
      <tr><td>plüs bari pátyolgatás</td><td>nagyon</td></tr>
      <tr><td>legózás</td><td>közepesen</td></tr>
      <tr><td>számítógépes játék</td><td>nagyon</td></tr>
      <tr><td>zokni csata</td><td>nagyon</td></tr>
    </table>
    <style type="text/css">
    </style>
  </body>
</html>
```



A **style** elembe fogjuk írni a CSS kódot, a HTML rész végig ugyanez marad, ezért azzal nem töltöm a helyet.

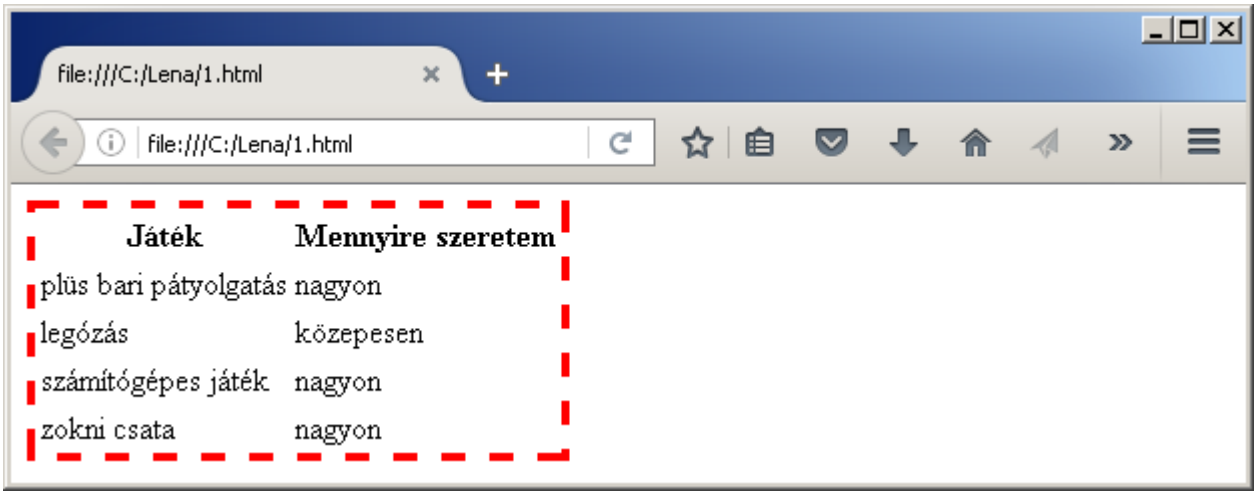
A **border**-rel csinálhatunk egy keretet a táblázat köré.

```
table { border: solid 1px black }
```

A **solid** azt jelenti, hogy folytonos vonal, **1px** a vonalvastagság, a **black** pedig a vonal színe.

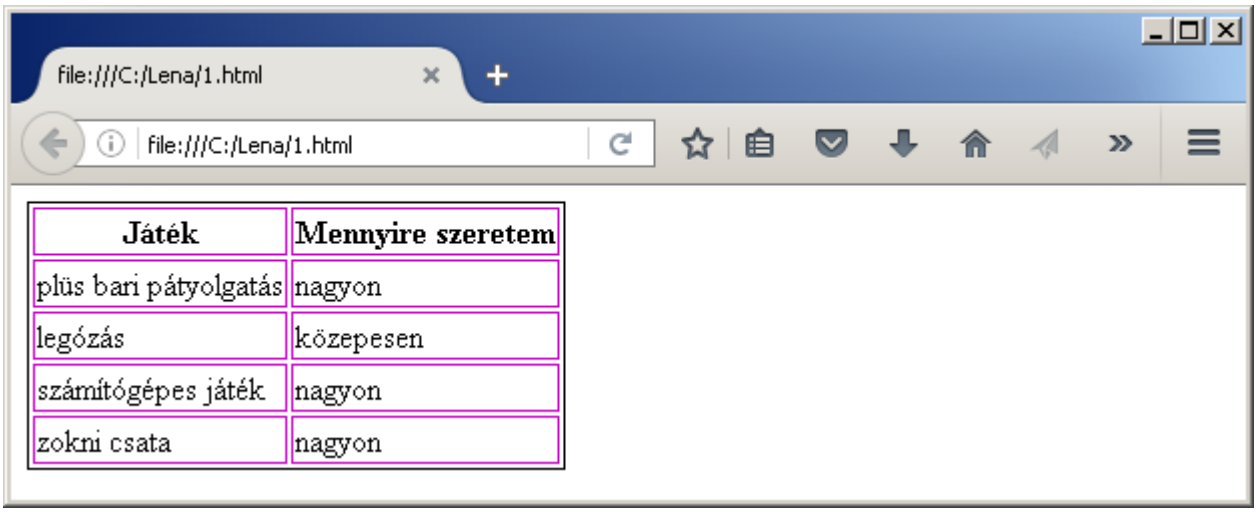
A **solid** helyett az alábbiakkal próbálkozhatunk még: **none**, **dotted**, **dashed**, **groove**, **ridge**, **inset**, **outset**.

```
table { border: dashed 4px red }
```



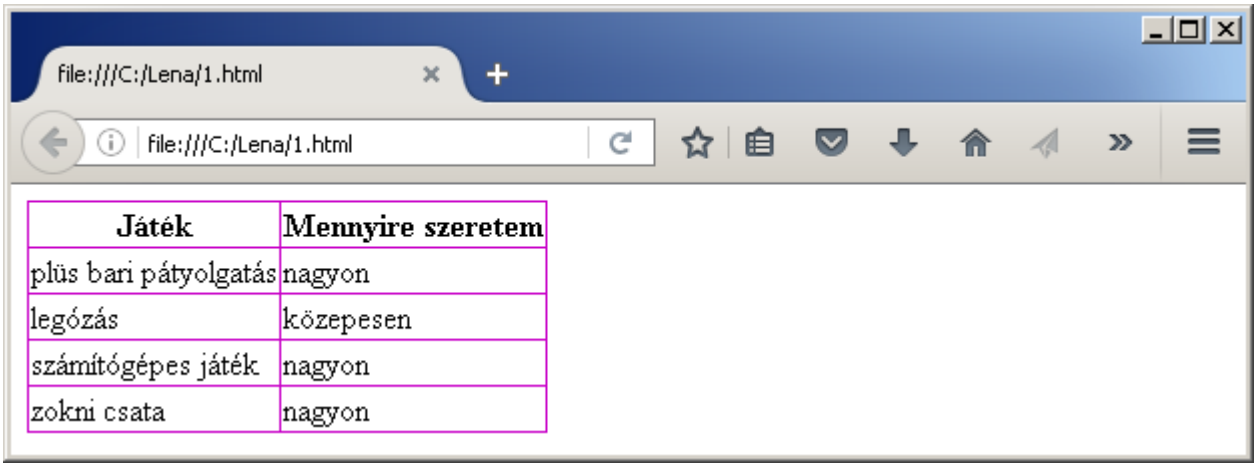
Térjünk inkább vissza a sima fekete vonalhoz, és a táblázat mezőinek is adjunk stílust. A táblázat mezőit celláknak is szokták nevezni.

```
table { border: solid 1px black }
th, td { border: solid 1px rgb(200,0,200) }
```



Elég furcsán kereteződtek be a mezők. Ha azt akarjuk, hogy csak egy vonal legyen köztük, azt a **border-collapse** segítségével érhetjük el. A **border-collapse**-ot a **table**-re kell rakni.

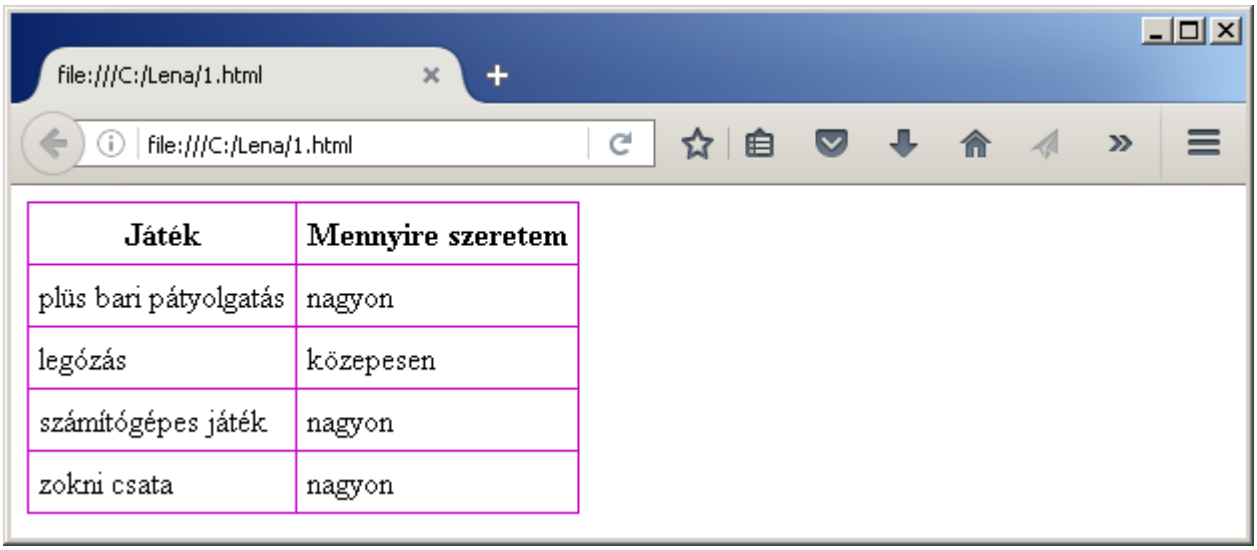
```
table { border: solid 1px black; border-collapse: collapse }
th, td { border: solid 1px rgb(200,0,200) }
```



Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

Szebb lenne, ha a mezők tartalma, vagyis a betűk és a rácsvonalak között egy kicsit több hely lenne. Ezt a **padding** –gal érhetjük el.

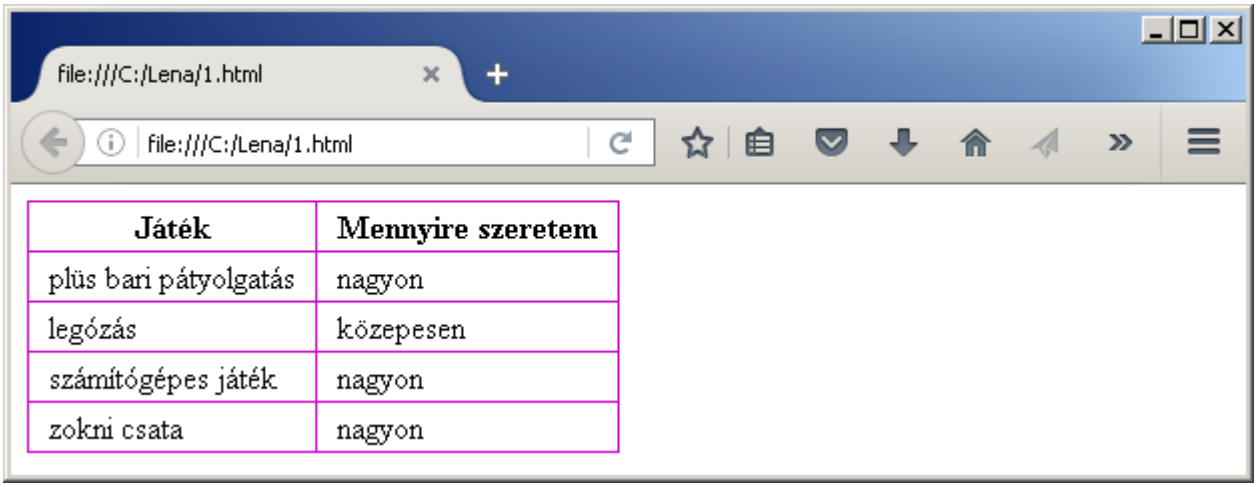
```
table { border: solid 1px black; border-collapse: collapse }
th, td { border: solid 1px rgb(200,0,200); padding: 5px }
```



Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

Vízszintesen legyen inkább még több hely, 10px, függőlegesen viszont elég 2px.

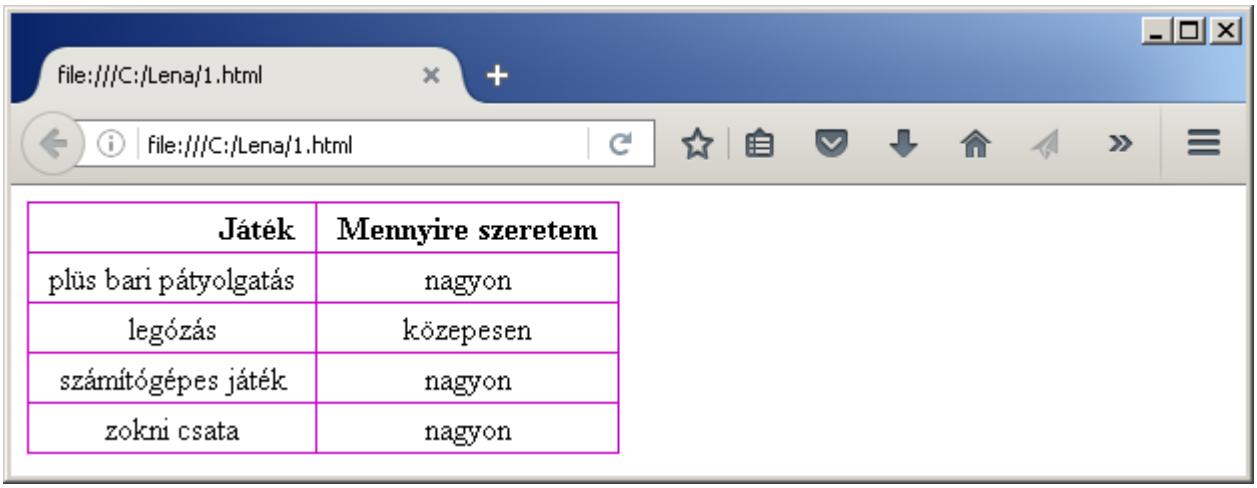
```
table { border: solid 1px black; border-collapse: collapse }
th, td { border: solid 1px rgb(200,0,200); padding: 2px 10px }
```



Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

A **text-align** segítségével balra, középre vagy jobbra igazíthatjuk a mezők tartalmát, vagyis a szöveget. Lehetséges értékei: **left**, **center**, **right**.

```
table { border: solid 1px black; border-collapse: collapse }
th, td { border: solid 1px rgb(200,0,200); padding: 2px 10px }
th { text-align: right }
td { text-align: center }
```



Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

A **vertical-align** segítségével függőlegesen rendezhetjük a mező tartalmát felfelé, középre vagy lefelé. Lehetséges értékei: **top**, **middle**, **bottom**.

Viszont a szöveg a táblázatban csak annyi helyet foglal el amennyi szükséges, és a **padding**-ra nem mehet rá, ezért alapesetben a **vertical-align** hatása nem látszik. Ezért a **height** segítségével meg kell növelnünk a mezők magasságát.

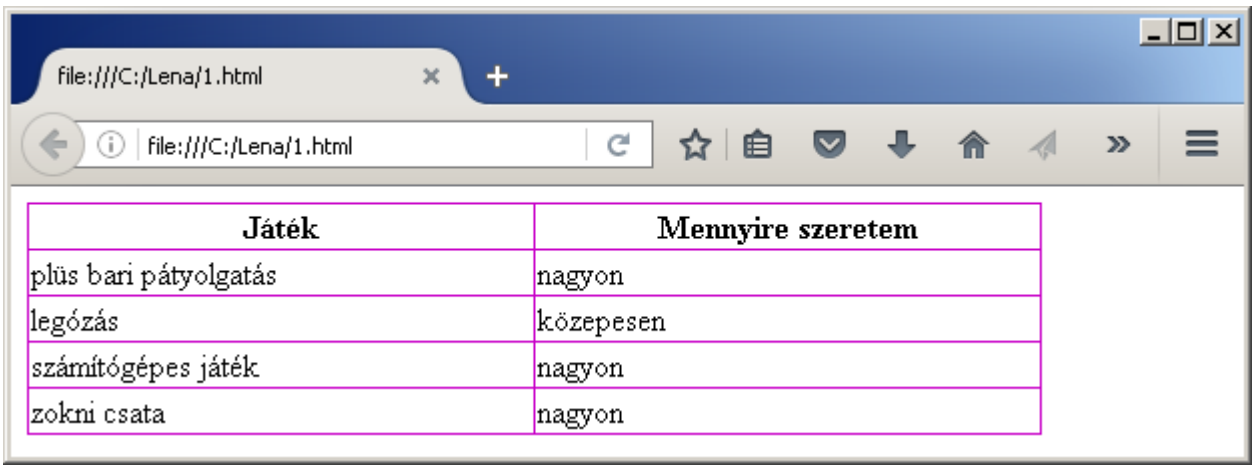
```
table { border: solid 1px black; border-collapse: collapse }
th, td { border: solid 1px rgb(200,0,200); padding: 2px 10px; height: 40px }
th { text-align: right; vertical-align: top }
td { text-align: center; vertical-align: bottom }
```

Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

A **height** tulajdonságot az alábbi HTML elemekre alkalmazhatjuk: **table**, **tr**, **th**, **td**.

A **width** segítségével szélességet állíthatunk be, ezekre az elemekre alkalmazhatjuk: **table**, **td**, **th**.

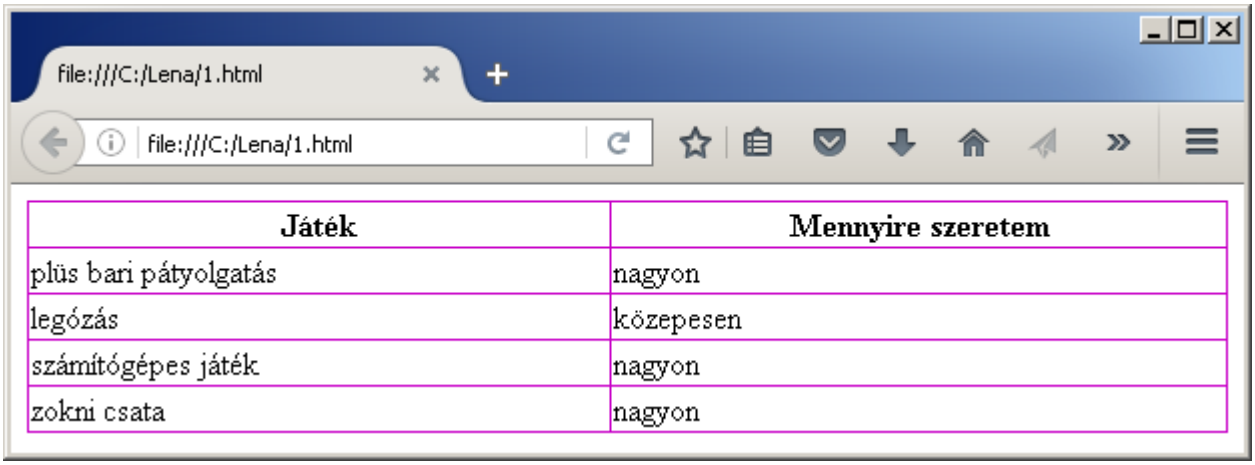
```
table { border: solid 1px black; border-collapse: collapse }
th, td { border: solid 1px rgb(200,0,200); width: 250px }
```



Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

Ha a **width**-et a **table** elemen állítjuk be, megmondhatjuk azt is, hogy a táblázat a rendelkezésre álló hely hány százalékát foglalja el.

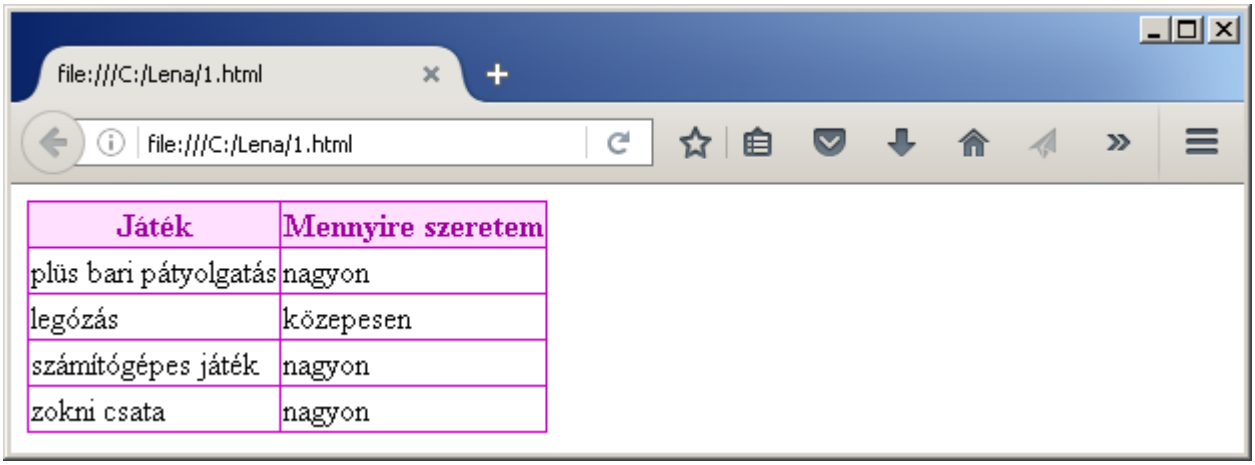
```
table { border: solid 1px black; border-collapse: collapse; width:100% }
th, td { border: solid 1px rgb(200,0,200) }
```



Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

A már ismert **color** és a **background** a táblázat mezőinél is működik.

```
table { border: solid 1px black; border-collapse: collapse }
th, td { border: solid 1px rgb(200,0,200) }
th { color: #A000A0; background: #FFE0FF}
```



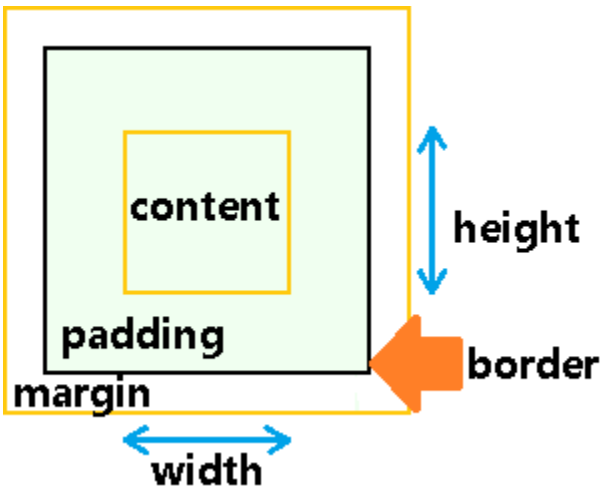
Játék	Mennyire szeretem
plüs bari pátyolgatás	nagyon
legózás	közepesen
számítógépes játék	nagyon
zokni csata	nagyon

A DOBOZ MODELL

Kétféle HTML elem van.

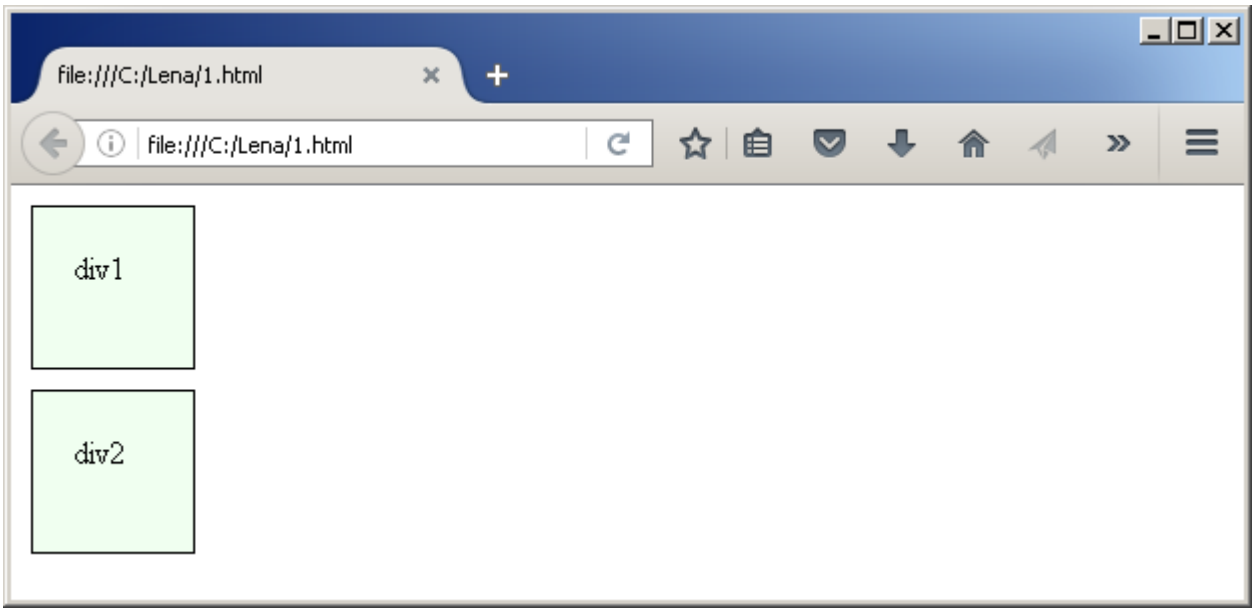
- Az egyik, ami új sorba kerül, pl.: **div**, **h1**, **p**, **table**. Ezeket **block** elemnek is hívják.
- A másik, ami nem kerül új sorba, pl.: **span**, **a**, **img**. Ezeket **inline** elemeknek is hívják.

A doboz modell a **block** típusú elemekre vonatkozik. A doboz modell arról szól, hogy belül van a **content**, magyarul tartalom, ami lehet szöveg, kép, bármi. A **content** körül van egy üres rész, aminek lehet háttérszíne, ez a **padding**. A **padding** körül van a **border**, ami egy látható keret. A **border**-en kívül pedig a **margin** van, ami átlátszó, de helyet foglal el, így távolságot tart a befoglaló elemtől és a szomszédos elemtől.



Nézzünk egy példát!

```
<html>
  <body>
    <div>div1</div>
    <div>div2</div>
    <style type="text/css">
      div {
        background: rgb(240,255,240);
        width: 40px; height: 40px;
        border: solid 1px black;
        padding: 20px;
        margin: 10px;
      }
      body {
        margin: 0
      }
    </style>
  </body>
</html>
```



A **background** mindig a **content**-re és a **padding**-ra vonatkozik.

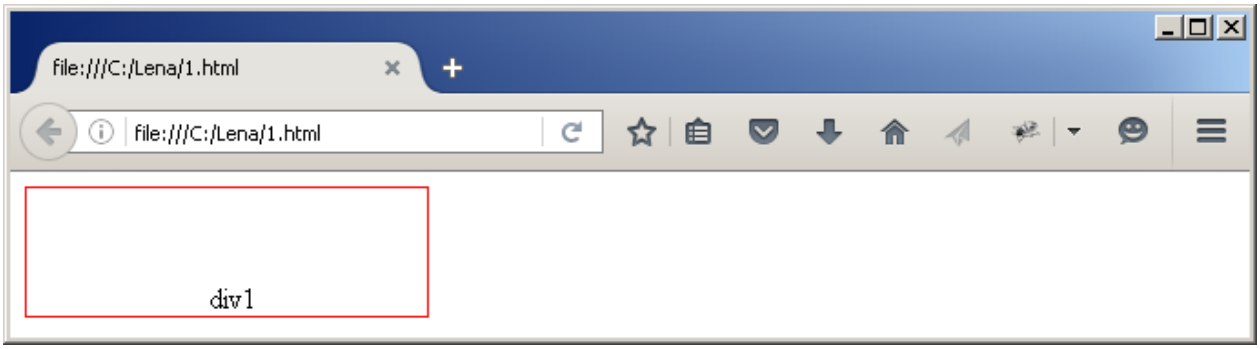
A **width** és a **height** (szélesség, magasság) csak a **content**-re vonatkozik.

A fenti négyzetek szélessége így áll össze: **border** + **padding** + **content width** + **padding** + **border** = 1 + 20 + 40 +20 + 1 = 82px. A magasság szintén 82px lesz.

Az négyzetek távolsága az oldal széleitől és egymástól pedig 10px. Ezt a **margin** állítja be. A **body** elemnek alapból be van állítva valami **margin**, ezt lenulláztam, hogy ne zavarjon be.

A **padding**-nak és a **margin**-nak adhatunk négy értéket is. Ekkor az első a **div** tetejére, a második a jobboldalára, a harmadik az aljára, a negyedik a baloldalára fog vonatkozni. Tehát fentről indulunk és az óramutató járása szerint megyünk körbe.

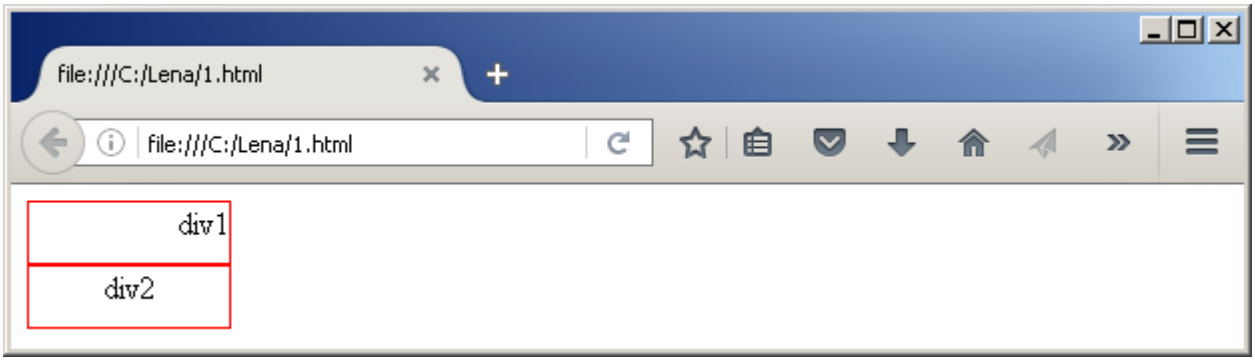
```
<html>
  <body>
    <div>div1</div>
    <style type="text/css">
      div { width: 20px; height: 20px; padding: 50px 100px 0 100px; border: solid 1px red }
    </style>
  </body>
</html>
```



A fenti példában a baloldali és a jobboldali padding terület 100 pont széles, és a fenti padding terület 50 pont magas.

A **text-align** segítségével balra, középre vagy jobbra igazíthatjuk a block típusú elem tartalmát, vagyis a szöveget. Lehetséges értékei: **left**, **center**, **right**. Ez a rendezés csak a content területen belül történik.

```
<html>
  <body>
    <div class="div1">div1</div>
    <div class="div2">div2</div>
    <style type="text/css">
      div { width: 100px; height: 30px; border: solid 1px red }
      div.div1 { text-align: right }
      div.div2 { text-align: center }
    </style>
  </body>
</html>
```

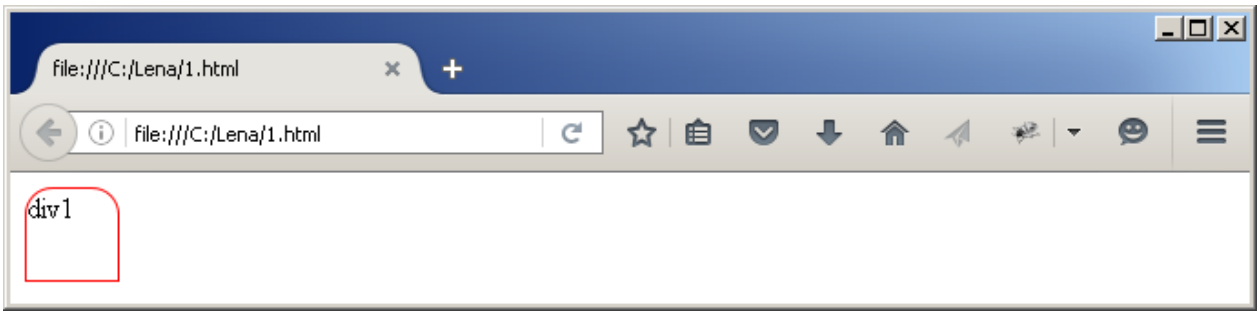


A **border**-nek adhatunk átlátszó színt is, ekkor az előző példában a **red** helyére azt kell írni, hogy **transparent**:

```
div { width: 100px; height: 30px; border: solid 1px transparent }
```

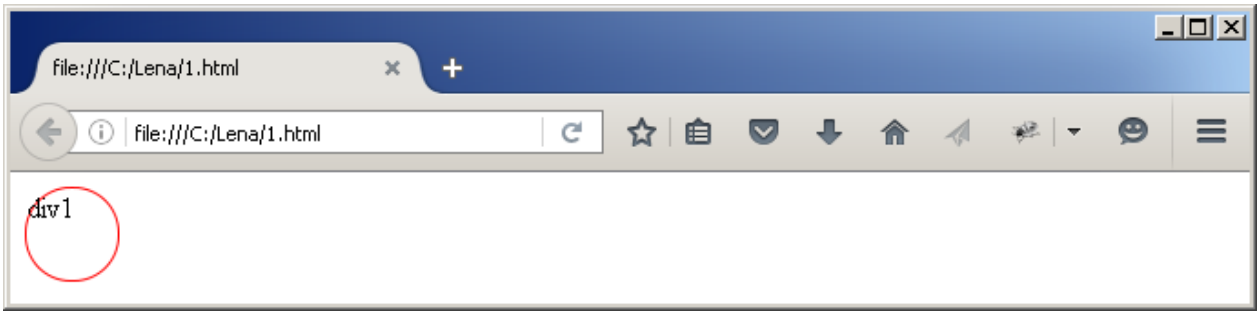
A **border-radius** segítségével lekerekíthetjük a border sarkait. Ha egy számot adunk meg, akkor az mind a négy sarokra vonatkozik, ha négyet, akkor az első a bal felső sarok, a többi pedig körbe az óramutató járása szerint. Minél nagyobb számot adunk meg annál jobban lekerekíti a sarkot.

```
<html>
  <body>
    <div>div1</div>
    <style type="text/css">
      div { width: 50px; height: 50px; border: solid 1px red; border-radius: 15px 15px 0 0 }
    </style>
  </body>
</html>
```



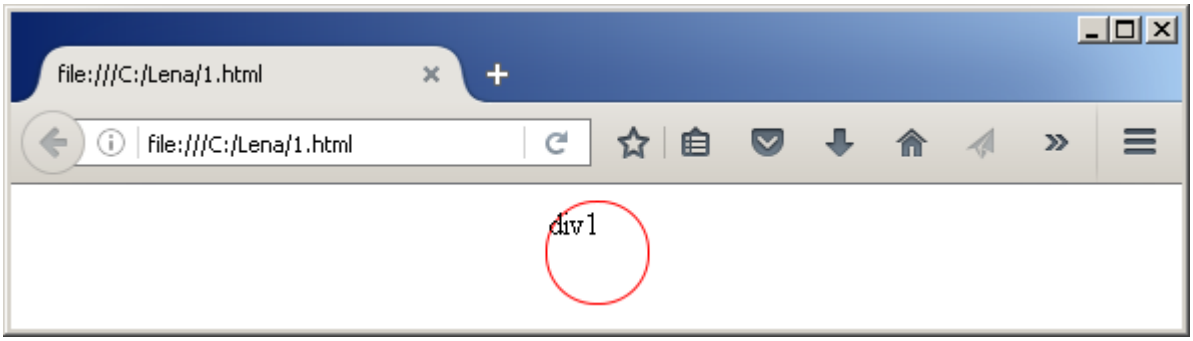
A **border-radius** segítségével kört is csinálhatunk. Első lépésben legyen a **width** és a **height** egyforma, hogy négyzetet kapjunk. Ezután a **border-radius**-t állítsuk a **width** vagy a **height** felére.

```
<html>
  <body>
    <div>div1</div>
    <style type="text/css">
      div { width: 50px; height: 50px; border: solid 1px red; border-radius: 25px }
    </style>
  </body>
</html>
```



Az utolsó trükk ebben a részben a **margin: auto**. Ez úgy állítja be a **margin** területet, hogy felül és alul 0 legyen, bal és jobb oldalt pedig annyi, hogy az elem pont középre kerüljön a befoglaló elemen belül, vagy ha nincs befoglaló elem, akkor a képernyőn belül.

```
<html>
  <body>
    <div>div1</div>
    <style type="text/css">
      div { width: 50px; height: 50px; border: solid 1px red; border-radius: 25px; margin: auto }
    </style>
  </body>
</html>
```

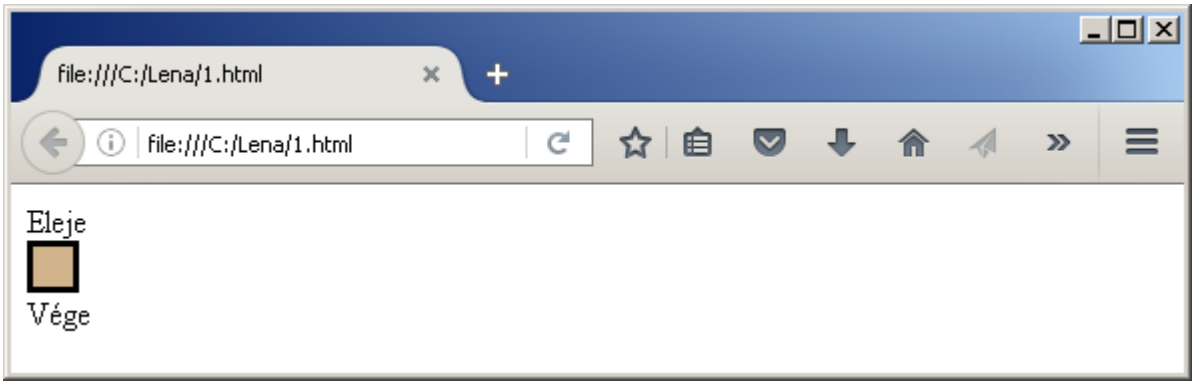


BLOCK TÍPUSÚ ELEMEK ELHELYEZÉSE

Block típusú elemeket háromféleképpen is elhelyezhetünk a képernyőn: **relative**, **absolute**, **fixed**.

Ha a három elhelyezési mód közül egyiket sem adjuk meg, akkor a blokk típusú elemünk, vagyis a bőrszínű doboz az előtte és az utána jövő tartalom közé kerül egy új sorba.

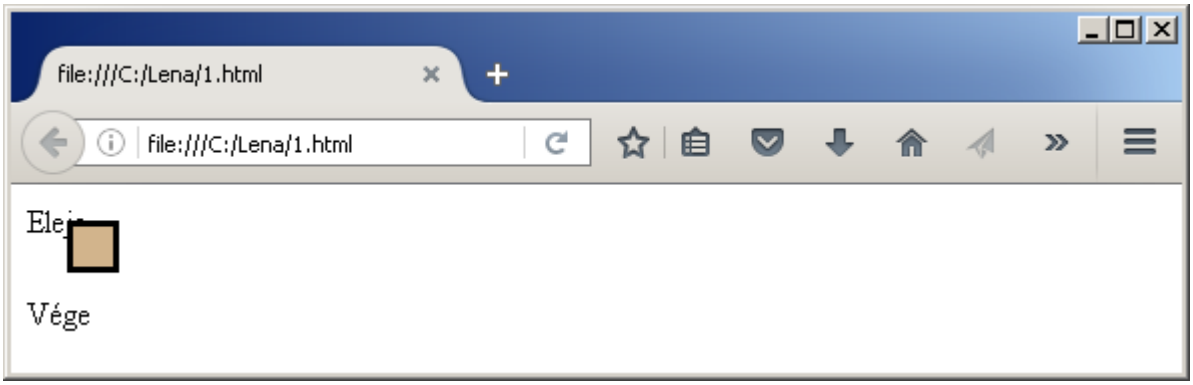
```
Eleje
<div style="background-color:tan; height:20; width:20; border-style:solid"></div>
Vége
```



A **position:relative** segítségével meg lehet mondani, hogy a **block** típusú elem a normál helyéhez képest mennyivel kerüljön lejjebb és jobbra.

A **top** mondja meg, mennyivel kerüljön lejjebb, vagy ha negatív számot adunk meg, akkor feljebb. A **left** mondja meg, mennyivel kerüljön jobbra, vagy negatív szám esetében balra a normális helyétől.

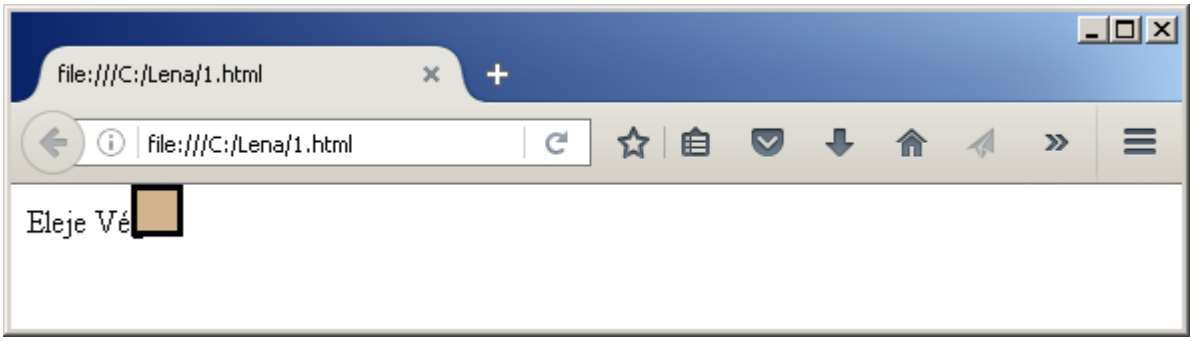
```
Eleje
<div style="position:relative; top:-10; left:20; background-color:tan; height:20; width:20; border-style:solid"></div>
Vége
```



A **position:relative** úgy helyezi át az elemet, hogy az elem eredeti helyét megtartja üresen. Vagyis ennek a beállításnak nincs túl sok haszna.

Ezeknél jóval hasznosabb a **position:absolute**, amivel kerek perec megmondhatjuk, hova kerüljön az elem a képernyőn belül. Pontosabban az elem bal felső sarka hova kerüljön a képernyő bal felső sarkához képest.

```
Eleje
<div style="position:absolute; top:0; left:60; background-color:tan; height:20; width:20; border-style:solid"></div>
Vége
```

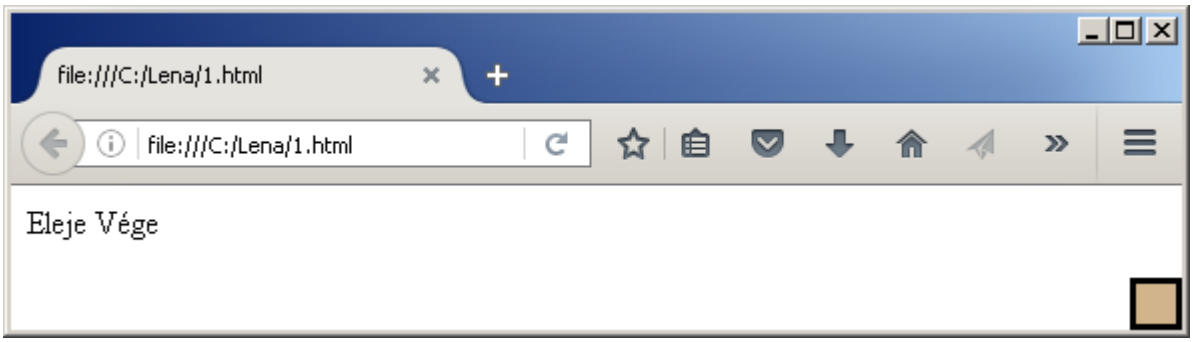


Mint láthatjuk, a **position:absolute** nem tartja meg az elem eredeti helyét.

Ha az elem egy másik olyan elemben van benne, aminek szintén adtunk meg **position**-t, akkor nem a képernyőhöz, hanem a külső elemhez képest számolódik a **top** és a **left**.

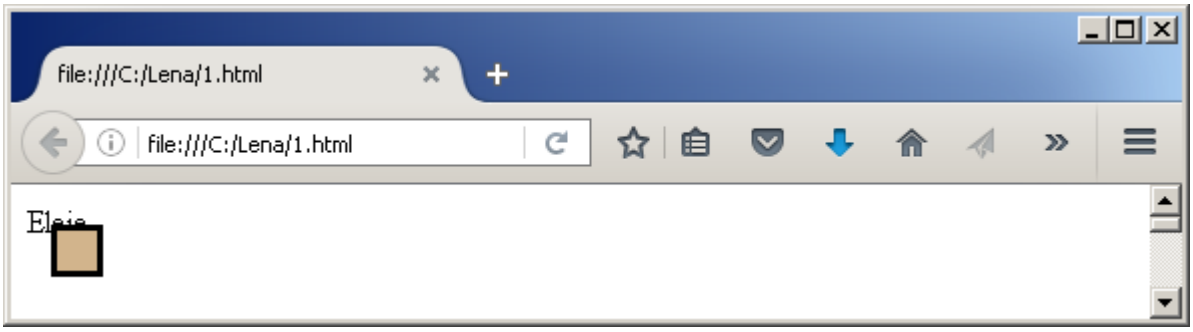
Ha nem szimpatikus a **top** és a **left**, használhatjuk a **right** és a **bottom** tulajdonságokat is. Ekkor a jobb alsó sarokhoz képest adhatjuk meg, hova kerüljön az elem.

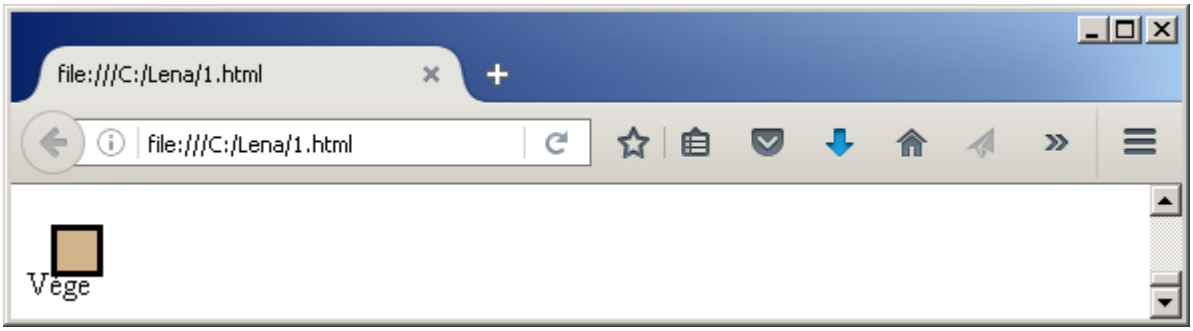
```
Eleje
<div style="position:absolute; bottom:0; right:0; background-color:tan; height:20; width:20;
border-style:solid"></div>
Vége
```



A **position:fixed** ugyanaz, mint a **position:absolute**, csak az a különbség, hogy ha a képernyőt lejjebb görgetjük,akkor is megtartja az elem a helyét.

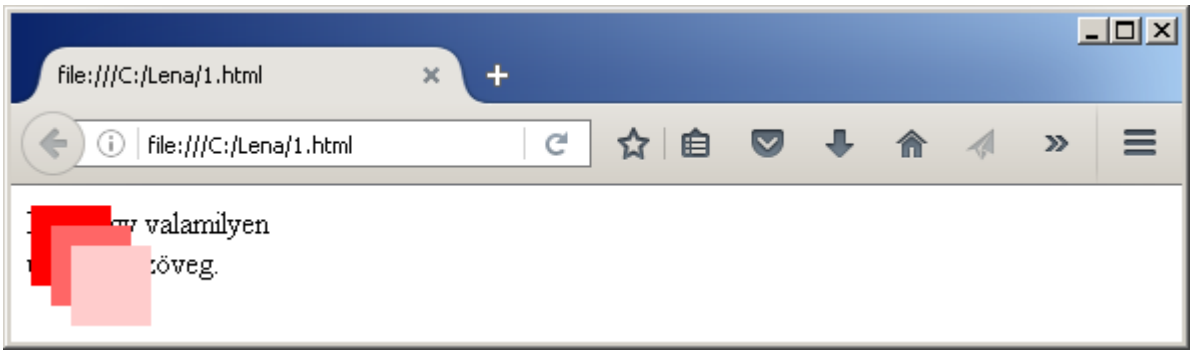
```
Eleje
<div style="position:fixed; top:20; left:20; background-color:tan; height:20; width:20; border-
style:solid"></div>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
Vége
```





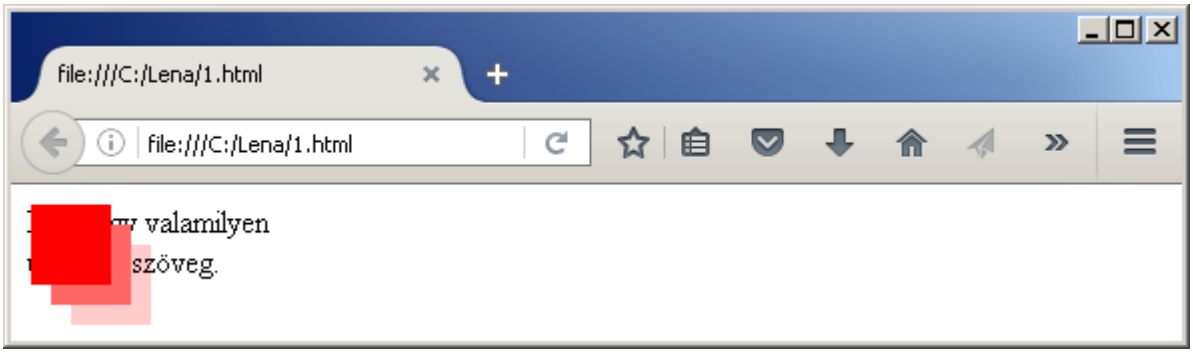
Ha több elemet úgy helyezek el, hogy egymást fedjék, akkor az első kerül alulra és az utolsó felülre. De mindegyik az oldal többi tartalma (az unalmas szöveg) fölé fog kerülni.

```
Ez itt egy valamilyen<br>
unalmas szöveg.
<div style="position:absolute; top:10; left:10; background-color:#FF0000; height:40;
width:40"></div>
<div style="position:absolute; top:20; left:20; background-color:#FF6666; height:40;
width:40"></div>
<div style="position:absolute; top:30; left:30; background-color:#FFCCCC; height:40;
width:40"></div>
```



Ezen a **z-index** segítségével változtathatunk. Minél nagyobb az értéke, annál feljebb kerül az elem. Ha negatív értéket adunk neki, akkor az elem az oldal többi tartalmánál is mélyebbre kerül.

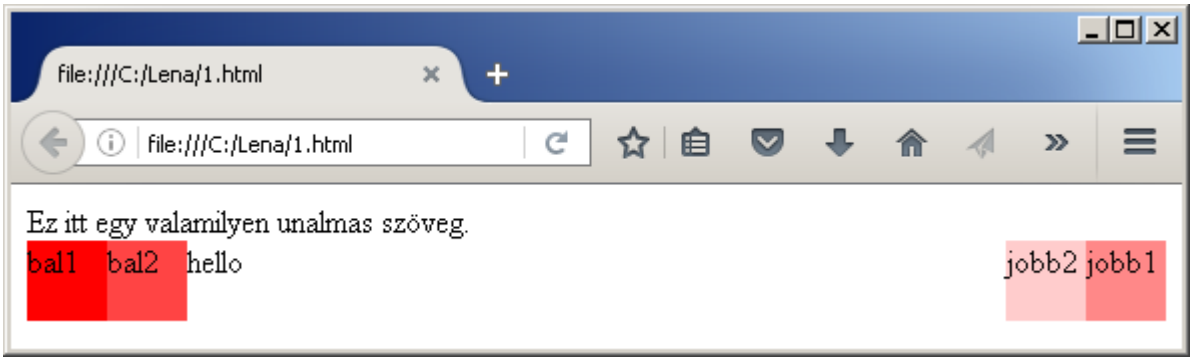
```
Ez itt egy valamilyen<br>
unalmas szöveg.
<div style="position:absolute; top:10; left:10; z-index:4; background-color:#FF0000; height:40;
width:40"></div>
<div style="position:absolute; top:20; left:20; z-index:2; background-color:#FF6666; height:40;
width:40"></div>
<div style="position:absolute; top:30; left:30; z-index:-2; background-color:#FFCCCC; height:40;
width:40"></div>
```



Az utolsó trükk a **float:lef**t és a **float:right**. Ezzel egy **block** típusú elemen belül tudunk úgy elhelyezni **block** típusú elemeket, hogy ne egymás alá kerüljenek, hanem egymás mellé. A **float:left** elemek baloldalra, a **float:right** elemek jobboldalra.

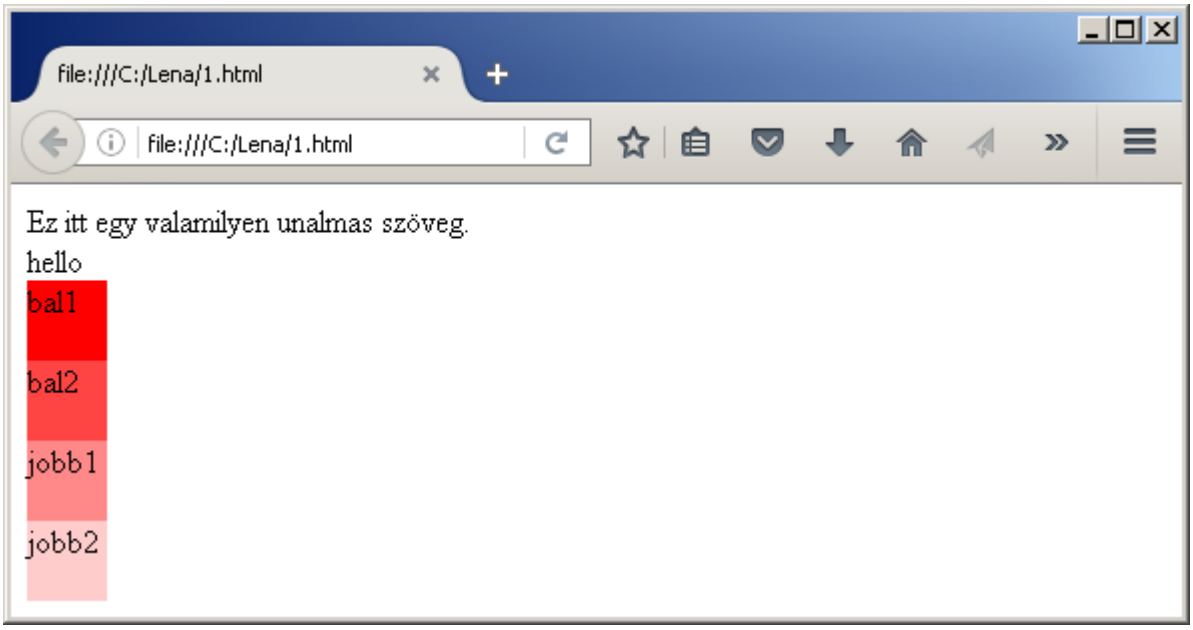
Ez itt egy valamilyen unalmas szöveg.

```
<div>
   hello
   <div style="float:left; background-color:#FF0000; height:40; width:40">bal1</div>
   <div style="float:left; background-color:#FF4444; height:40; width:40">bal2</div>
   <div style="float:right; background-color:#FF8888; height:40; width:40">jobb1</div>
   <div style="float:right; background-color:#FFCCCC; height:40; width:40">jobb2</div>
</div>
```



Ha a **float**-okat kihagynánk, egymás alá kerülnének az elemek, mint minden normális **block** típusú elem.

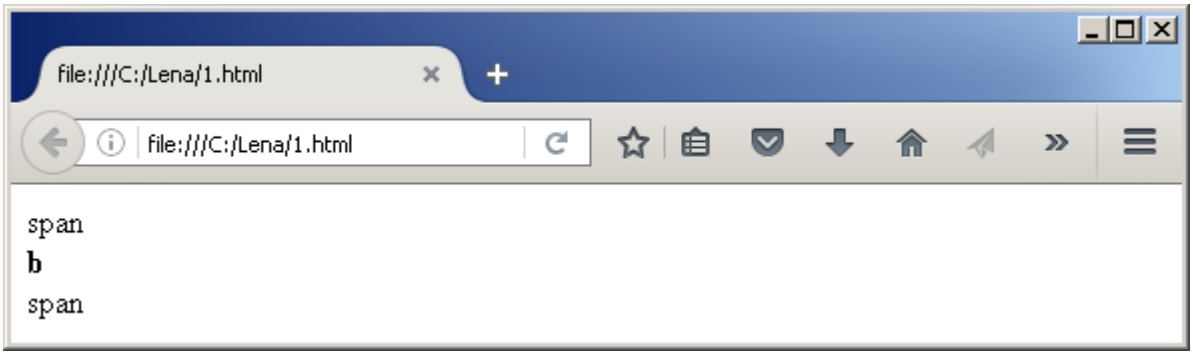
```
Ez itt egy valamilyen unalmas szöveg.
<div>
   hello
   <div style="background-color:#FF0000; height:40; width:40">bal1</div>
   <div style="background-color:#FF4444; height:40; width:40">bal2</div>
   <div style="background-color:#FF8888; height:40; width:40">jobb1</div>
   <div style="background-color:#FFCCCC; height:40; width:40">jobb2</div>
</div>
```



ELEMEK ÁTVÁLTÁSA INLINE-RÓL BLOCK TÍPUSÚRA ÉS VISSZA

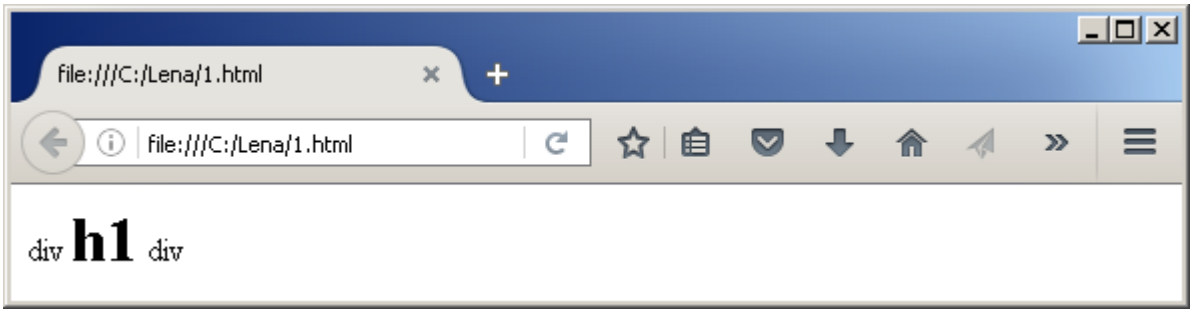
Inline típusú elemeket a **display:block** segítségével alakíthatunk **block** típusú elemmé. A **block** típusú elemeknél width, height megadható, előtte és utána új sor lesz.

```
<span style="display:block">span</span>
<b style="display:block">b</b>
<span style="display:block">span</span>
```



Block típusú elemeket a **display: inline** segítségével alakíthatunk **inline** típusú elemekké, vagyis egy sorba fognak kerülni.

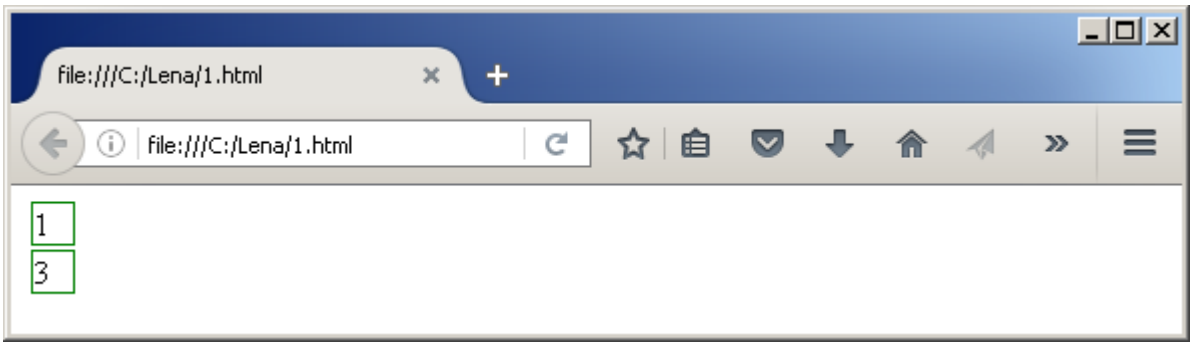
```
<div style="display:inline">div</div>
<h1 style="display:inline">h1</h1>
<div style="display:inline">div</div>
```



BLOCK TÍPUSÚ ELEMEK ELTŰNTETÉSE

A **display:none** segítségével úgy eltűntethetünk egy elemet, mintha ott se lenne.

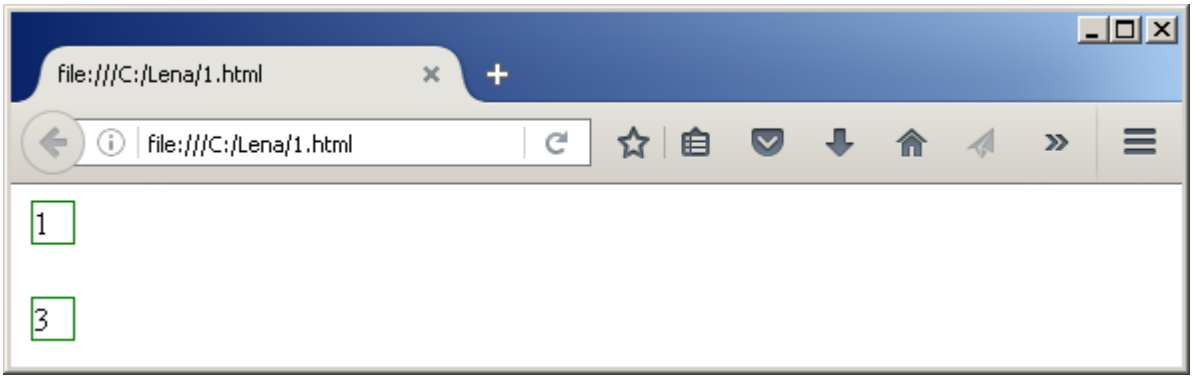
```
<html>
  <body>
    <div>1</div>
    <div class="nemlatszik">2</div>
    <div class="latszik">3</div>
    <style type="text/css">
      div { width: 20px; height: 20px; border: solid 1px green; margin: 2px }
      div.nemlatszik { display: none}
      div.latszik { display: block}
    </style>
  </body>
</html>
```



A második **div**-et tűntettük el. Visszakapcsolás: **display:block** vagy **display: inline**.

Az eltűntetés másik módja a **visibility:hidden**. Ez úgy tűnteti el az elemet, hogy a helyét megtartja üresen.

```
<html>
  <body>
    <div>1</div>
    <div class="nemlatszik">2</div>
    <div class="latszik">3</div>
    <style type="text/css">
      div { width: 20px; height: 20px; border: solid 1px green; margin: 2px }
      div.nemlatszik { visibility:hidden }
      div.latszik { visibility:visible }
    </style>
  </body>
</html>
```

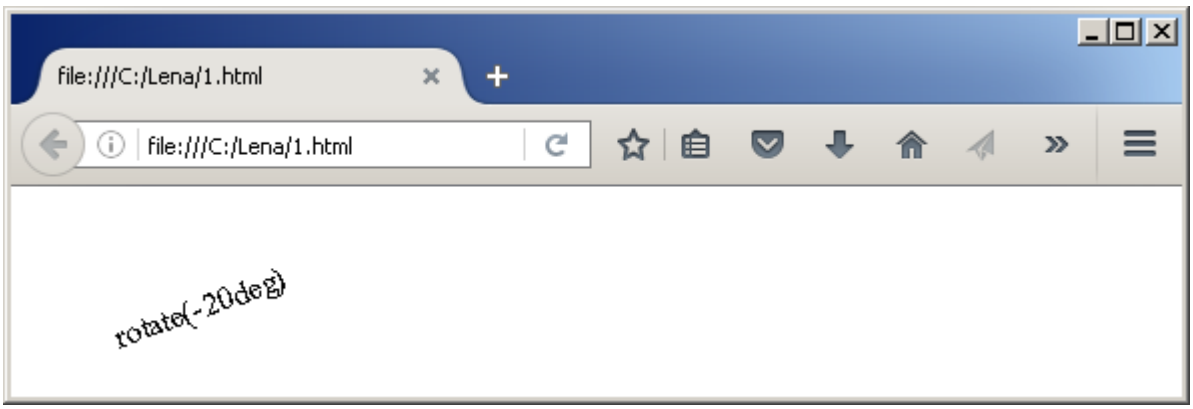


Visszakapcsolás: **visibility:visible**.

TRANSZFORMÁCIÓK

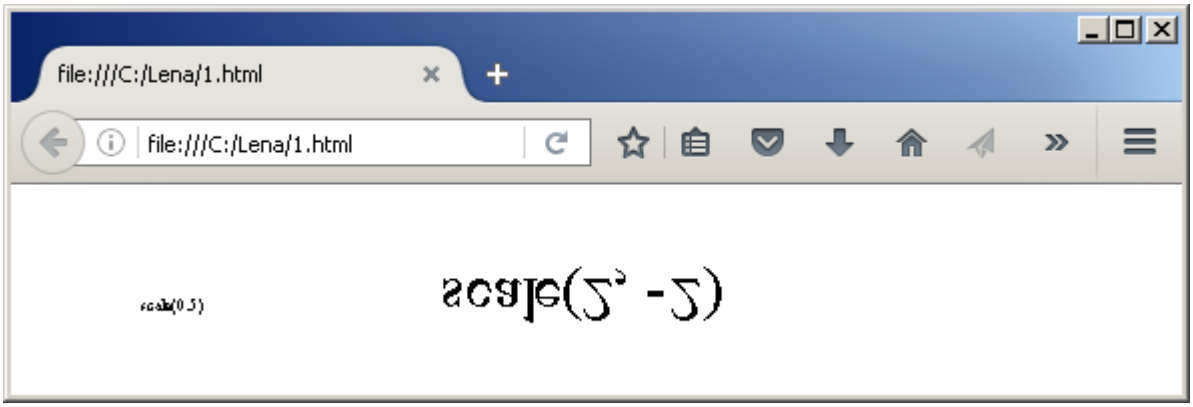
A **transform: rotate** segítségével elforgathatunk egy elemet. Meg kell adni, hogy hány fokkal akarjuk elforgatni. A forgatás óramutató járásával megegyezően történik. Viszont, ha negatív számot adunk meg, akkor óramutató járásával ellentétesen.

```
<html><body>
  <div class="c1">rotate(-20deg)</div>
  <style>
    .c1 { position: absolute; top: 50; left: 50; transform: rotate(-20deg) }
  </style>
</body></html>
```



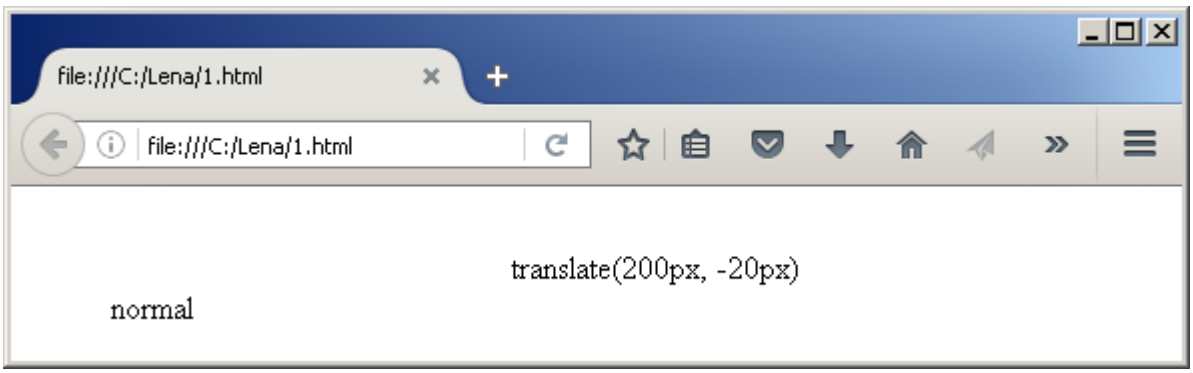
A **scale** segítségével kicsinyíthetünk és nagyíthatunk. Az egynél kisebb pozitív szám kicsinyítést jelent, az egynél nagyobb nagyítást, a negatív szám pedig tükrözést. Ha két számot adunk meg, akkor az első a vízszintes, a második pedig a függőleges irányba történő nagyítást adja meg.

```
<html><body>
  <div class="c1">scale(0.5)</div>
  <div class="c2">scale(2, -2)</div>
  <style>
    .c1 { position: absolute; top: 50; left: 50; transform: scale(0.5) }
    .c2 { position: absolute; top: 50; left: 250; transform: scale(2, -2) }
  </style>
</body></html>
```



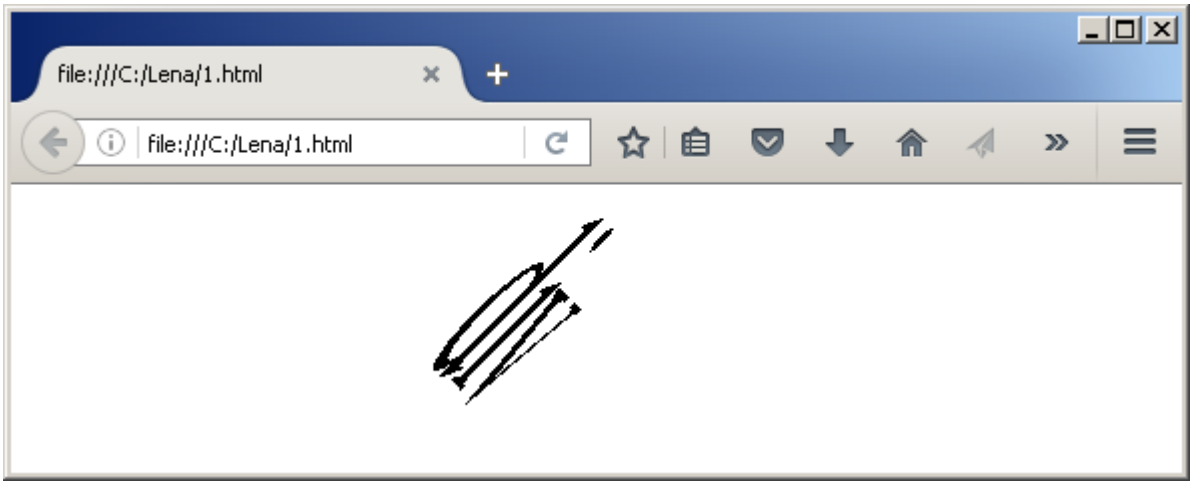
A **translate** segítségével elmozdíthatunk egy elemet a normál helyéről. Az első szám a vízszintes mozgatás, pozitív szám balra, negatív jobbra. A második szám a függőleges mozgatás, pozitív lefelé, negatív felfelé.

```
<html><body>
  <div class="c1">normal</div>
  <div class="c2">translate(200px, -20px)</div>
  <style>
    .c1 { position: absolute; top: 50; left: 50 }
    .c2 { position: absolute; top: 50; left: 50; transform: translate(200px, -20px) }
  </style>
</body></html>
```



Használhatunk egy elemen többféle transzformációt is egyszerre:

```
<html><body>
  <div class="c1">div</div>
  <style>
    .c1 {
      position: absolute; top: 50; left: 50;
      transform: translate(200px, 0) rotate(45deg) scale(2,10)
    }
  </style>
</body></html>
```

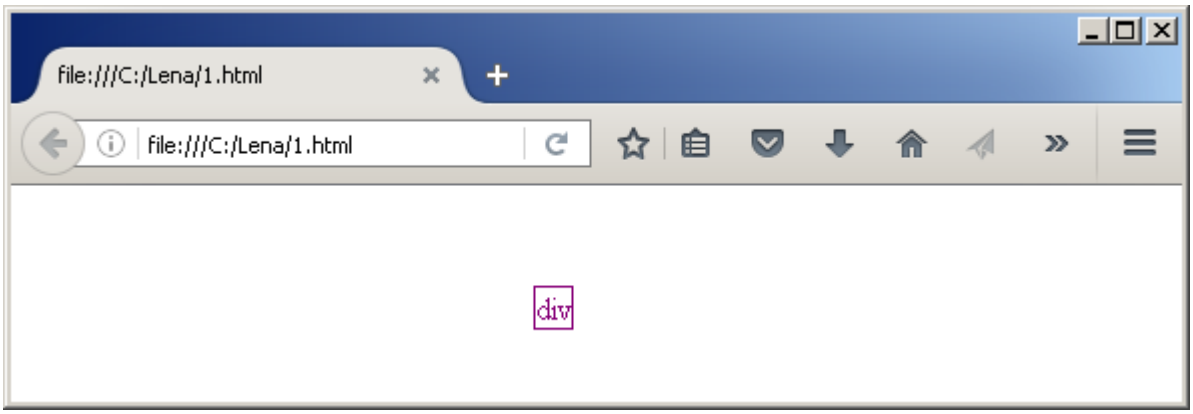


ANIMÁCIÓ

Ha mozgóképet akarunk csinálni, először meg kell adnunk egy **@keyframes**-ben, hogy a HTML elem melyik tulajdonságai változzanak, azoknak a tulajdonságoknak mi legyen a kezdeti és végső értéke. A lenti példában a **div** 50px-től mozog 500px-ig jobbra, közben fokozatosan átszíneződik pirosról kékre. A **@keyframes**-nek adunk egy nevet is, ami most ezmozog.

Az ezmozog **@keyframes**-t az **animation** tulajdonság segítségével használhatjuk fel egy vagy több HTML elemnél. A 3s azt jelenti, hogy 3 másodperc alatt történjen meg az átváltozás. Az **alternate** azt jelenti, hogy először oda, aztán visszafelé is megtörténjen a változás. Az **infinite** pedig azt jelenti, hogy végtelen ideig ismételgesse az animációt.

```
<html><body>
  <div class="c1">div</div>
  <style>
    @keyframes ezmozog { from { left:50px; color:red} to {left:500px; color:blue}}
    .c1 {
      border: 1px solid;
      position:absolute; top:50; left:50;
      animation: ezmozog 3s alternate infinite
    }
  </style>
</body></html>
```

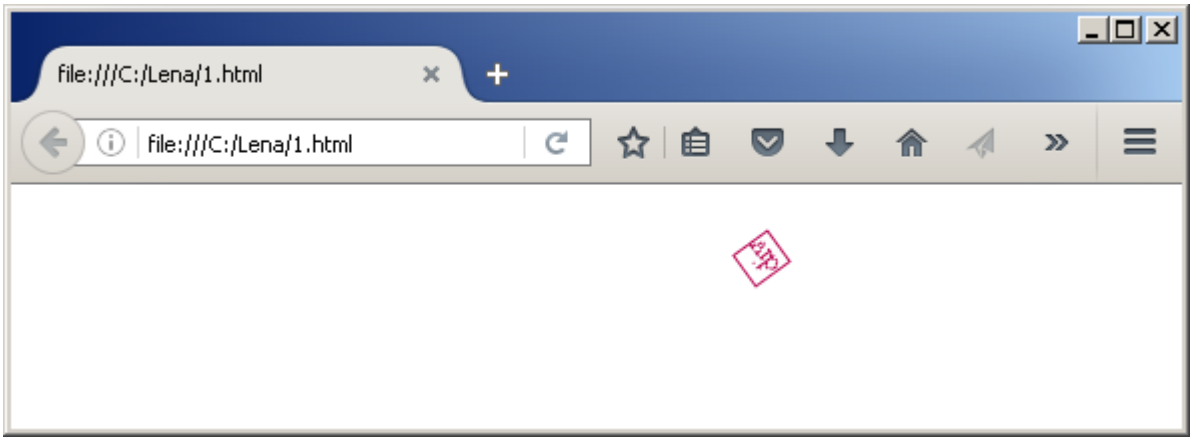


Az **animation**-on belül az **alternate** helyett írhatjuk az is, hogy **normal**, ami azt jelenti, hogy csak oda. A **reverse** pedig azt jelenti, hogy csak visszafele animáljon.

Az **animation**-on belül az **infinite** helyett megmondhatjuk, hogy hányszor ismétlődjön az animáció.

A **@keyframes**-en belül a **from** helyére írhatunk **0%**-ot is, a **to** helyére **100%**-ot, és közbenső lépéseket is felvehetünk, pl. 50%.

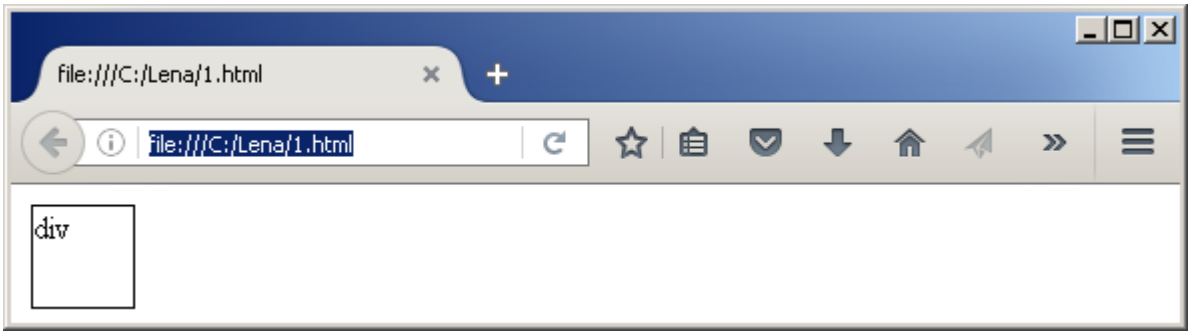
```
<html><body>
  <div class="c1">div</div>
  <style>
    @keyframes ezmozog {
      0%   { left:50px;  top:50px; color:black;  transform: rotate(0deg) }
      25%  { left:275px; top:90px; color:brown;  transform: rotate(90deg) }
      50%  { left:500px; top:50px; color:red;    transform: rotate(180deg) }
      75%  { left:275px; top:10px; color:purple; transform: rotate(270deg) }
      100% { left:50px;  top:50px; color:black;  transform: rotate(360deg) }}
    .c1 {
      border: 1px solid;
      position:absolute; top:50; left:50;
      animation: ezmozog 5s normal 3
    }
  </style>
</body></html>
```



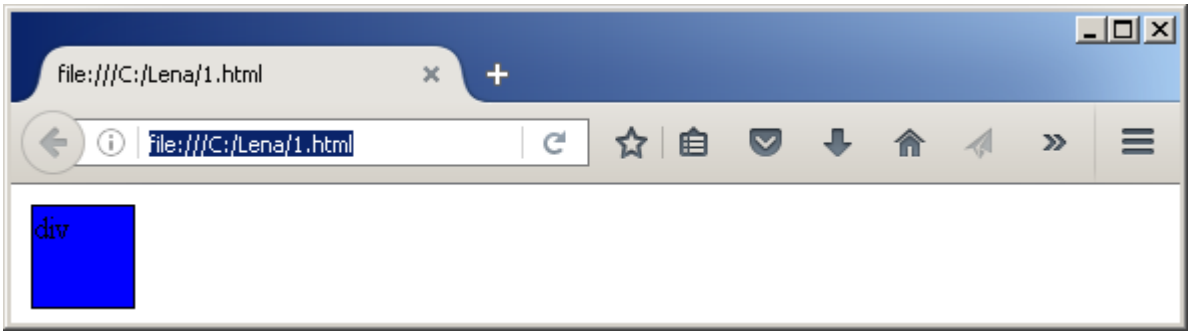
PSZEUDO OSZTÁLYOK

A pszeudo osztályokat kettősponttal kezdődnek. Az egyik ilyen a **:hover**, ami azt írja le, hogy milyen legyen a HTML elem, ha fölé megyünk az egérrel. A **:hover**-t bármilyen elemre rá lehet tenni.

```
<html><body>
  <div class="c1">div</div>
  <style>
    .c1 { border: 1px solid; position:absolute; top:10; left:10; width:50; height:50 }
    .c1:hover { background: blue }
  </style>
</body></html>
```

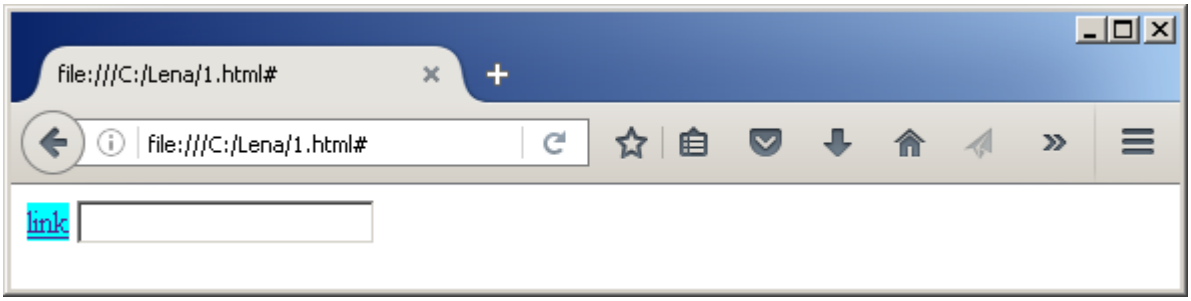


Ha a **div** fölé húzzuk az egeret, a háttere kék lesz:



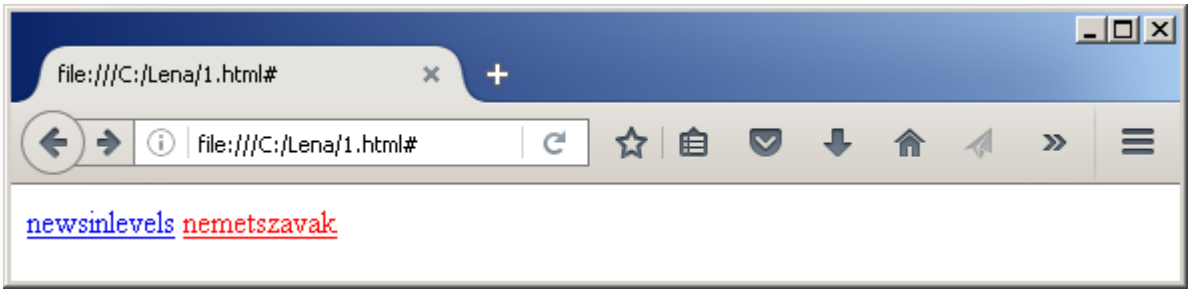
A **:focus** csak olyan elemeknél használható, amikbe bele lehet egérrel klikkelni. És akkor történik valami, ha beléjük klikkelünk.

```
<html><body>
   <a href="#">link</a>
   <input type="text"></input>
   <style>
      input:focus { background:lime }
      a:focus { background:aqua }
   </style>
</body></html>
```



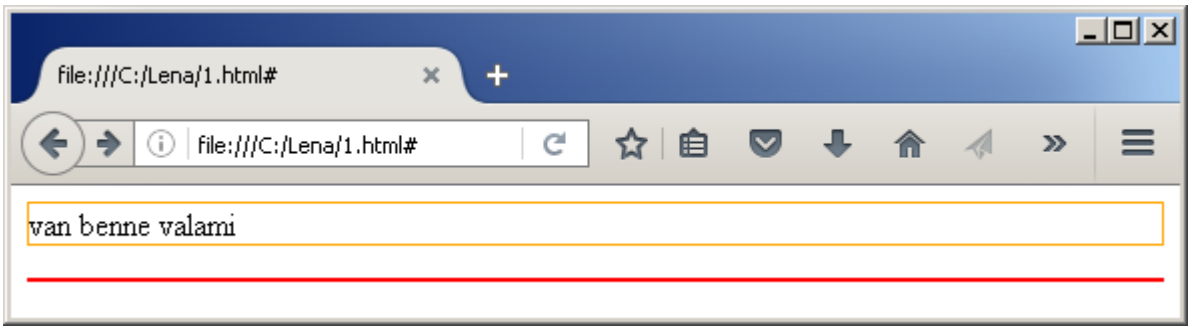
A **:visited** linkekre vonatkozik. A sikeresen meglátogatott linkeknek adhatunk vele tulajdonságokat.

```
<html><body>
   <a href="http://www.newsinlevels.com">newsinlevels</a>
   <a href="http://nemetszavak.hu">nemetszavak</a>
   <style>
      a: { color: blue}
      a:visited { color:red }
   </style>
</body></html>
```



Az **:empty** segítségével olyan elemeket választhatunk ki, amikben nincs semmi, pl.: ** **.

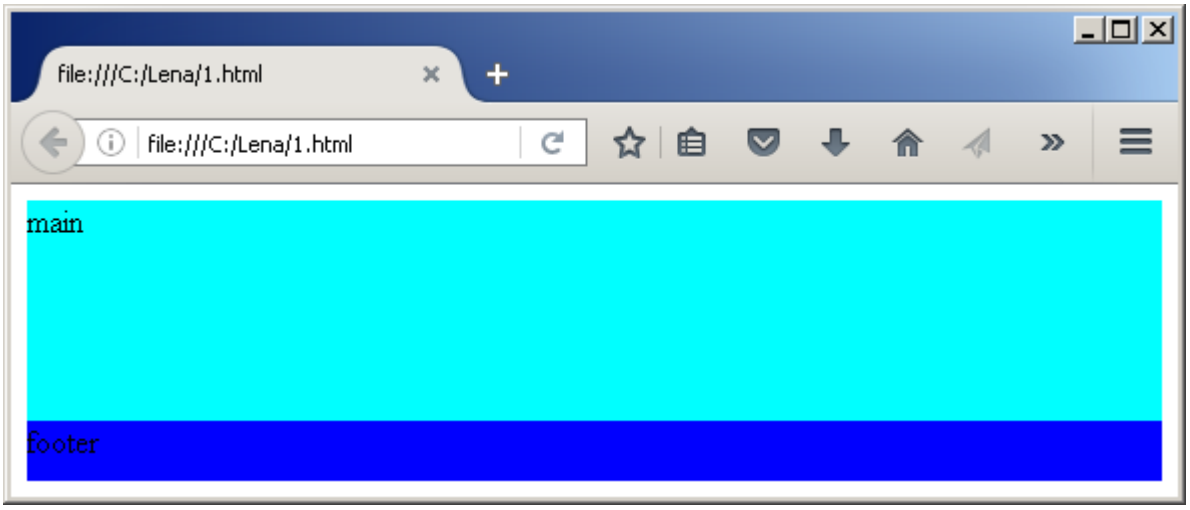
```
<html><body>
  <p>van benne valami</p>
  <p></p>
  <style>
    p { border: 1px solid orange }
    p:empty { border: 1px solid red }
  </style>
</body></html>
```



NÉHÁNY HALADÓ TRÜKK

Ha egy olyan oldalt szeretnénk készíteni, aminek az alján van egy csík, amiben valamilyen szöveget akarunk megjeleníteni, akkor a **sticky footer**, magyarul ragadós lábjegyzet trükköt használhatjuk. Azért hívják így, mert hiába méretezzük át az oldalt, az alsó csík mindig az oldal alján marad.

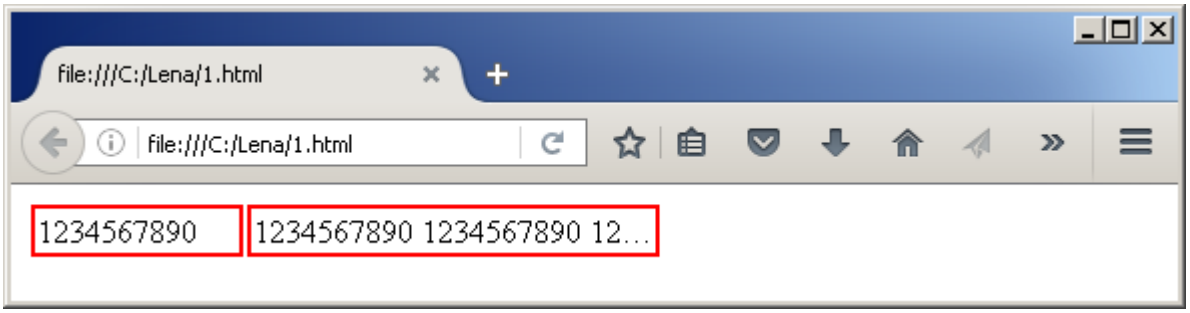
```
<html><body>
 <div id="main">main</div>
 <div id="footer">footer</div>
 <style>
  #main { background: cyan; height: 100%; margin-bottom: -30px }
  #footer { background: blue; height: 30px }
 </style>
</body></html>
```



A trükk lényege, hogy két **div** elemet veszünk fel, amik maguktól is egymás alatt jelennek meg. Az alsó lesz a **sticky footer**, aminek a magasságát 30px-re állítottuk. Viszont a felső **div** magasságát 100%-ra állítottuk, ezért az alsónak pont nem maradna már semmi hely a képernyőn. Ezért azt mondjuk, hogy a felső **div** alatt maradjon ki -30px hely, és ezért az alsó **div** rá fog lógni a felső aljára.

A következő trükk azt mutatja meg, hogy tehetünk automatikusan …-ot egy szöveg mögé, ha az nem fér el a helyén. A **min-width** és a **max-width** azt jelenti, hogy a **td** minimum és maximum milyen széles lehet. A **text-overflow: ellipsis** azt jelenti, hogy ha a szöveg nem fér el, tegyen …-ot a végére. A …-ot angolul ellipsis-nek hívják. Az **overflow: hidden** gondoskodik róla, hogy a túlnyúló szövegrész ne lógjon ki a **td**-ből, hanem inkább ne látszódjon. A **white-space: nowrap** arra jó, hogy az üres helyek mentén ne törjön a szöveg több sorba.

```
<html><body>
 <table>
  <td>1234567890</td>
  <td> 1234567890 1234567890 1234567890 1234567890 1234567890</td>
 </table>
 <style>
  td {
   border: 2px solid red;
   min-width: 100px;
   max-width: 200px;
   text-overflow: ellipsis;
   overflow: hidden;
   white-space: nowrap
  }
 </style>
</body></html>
```



Feladat: Nézd meg, hogy változik a táblázat, ha az **overflow: hidden** vagy a **white-space: nowrap** sort kikommentezed. Ezt úgy teheted, meg, hogy **/*** -ot teszel elé és ***/** -ot mögé:

```
/*    white-space: nowrap */
```

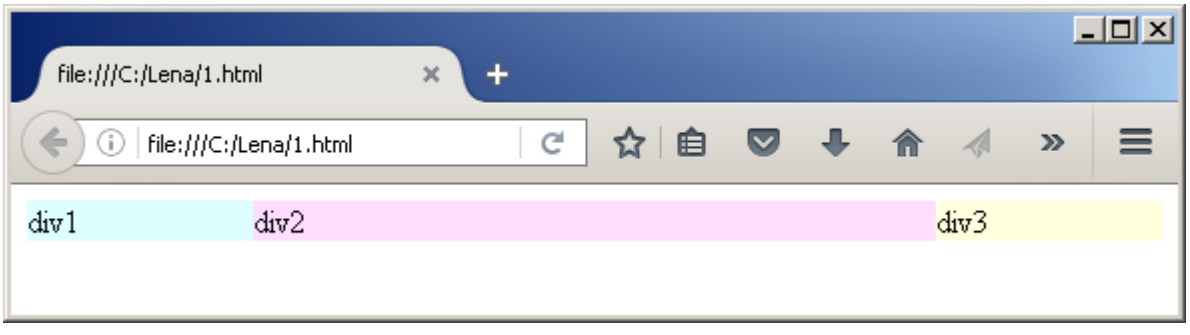
MOBILTELEFON, NYOMTATÓ KOMPATIBILITÁS

Ha a weboldalt mobiltelefonon akarjuk megjeleníteni, számítani kell arra, hogy a böngésző szélessége kisebb lesz, mint számítógépen. Erre a problémára megoldás lehet, ha máshogy jelenítjük meg a weboldalt keskeny és máshogy széles képernyőn.

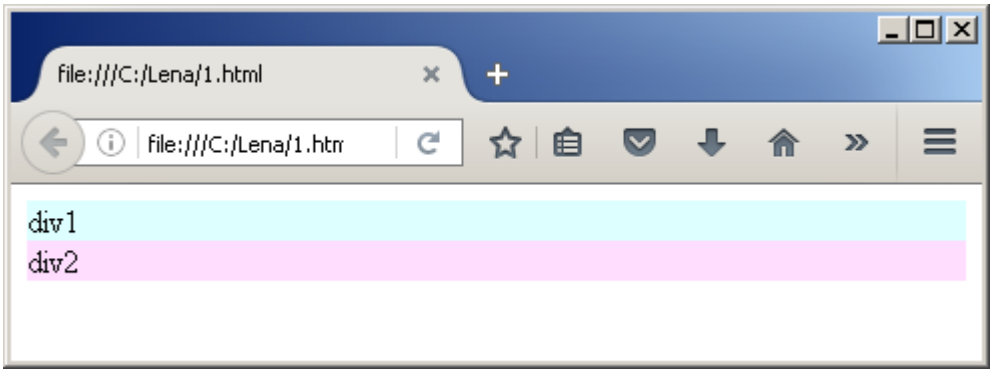
A **@media** segítségével megadhatunk olyan tulajdonságokat, amik csak bizonyos képernyőszélesség alatt vagy fölött kapcsolnak be. Azt is megmond hatjuk, hogy csak a képernyőn vagy csak nyomtatáskor kapcsoljanak be.

Az alábbi példa 500 pontnál szélesebb böngésző ablak esetén a **float:left** segítségével egymás mellé helyezi el a három **div**-et. Ha 500 pontnál kisebb a hely, akkor nem kapcsolja be a **float:left** -et, vagyis egymás alá kerülnek a **div**-ek, és a div3 el fog tűnni. Nyomtatáskor pedig egymás alatt jelenik meg a három div, mert az **only screen** (csak képernyő) miatt egyik **@media** sem fog bekapcsolni.

```
<html><body>
 <div id="div1">div1</div>
 <div id="div2">div2</div>
 <div id="div3">div3</div>
 <style>
  #div1 { background: #DDFFFF }
  #div2 { background: #FFDDFF }
  #div3 { background: #FFFFDD }
  @media only screen and (min-width: 501px) {
   div { float:left }
   #div1 { width:20% }
   #div2 { width:60% }
   #div3 { width:20% }
  }
  @media only screen and (max-width: 500px) {
   #div3 { display:none }
  }
 </style>
</body></html>
```



Ha egy kicsit keskenyebbre húzzuk a böngésző ablakot, így fog kinézni a weboldal:



A **@media** után **only screen** (csak képernyő) helyett írhatunk **only print** (csak nyomtatás) értéket is.

Az **and** előtti és az **and** utáni rész is kihagyható:

```
@media (max-width: 500px) { …

@media screen { …
```

Ha külső CSS fájlokat használunk, megmondhatjuk melyik CSS fájl töltődjön be széles és keskeny képernyő esetében:

```
<link rel="stylesheet" type="text/css" media="only screen and (min-width: 501px)"
href="szeles.css" />
<link rel="stylesheet" type="text/css" media="only screen and (max-width: 500px)"
href="keskeny.css" />
```

Jó tanács: Nyomtatáskor a **@media print** segítségével érdemes a háttérszíneket, különösen a **body background**-ot fehérre állítani.

Ha azt akarjuk, hogy mobilon is jól nézzen ki az oldalunk, a fentieken kívül egy ilyen meta beállítást is kell tennünk a **<head>**-be:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Ez egyrészt beállítja, hogy az oldal szélessége annyi legyen, hogy ne kelljen jobbra-balra görgetni a böngészőben. Másrészt kicsinyítés és nagyítás nélkül jeleníti meg az oldalt.

3 Weboldal publikálása

Tárhely és Domain név

Ahhoz, hogy az oldalunkat más is láthassa az interneten tárhelyre és domain névre van szükségünk.

A tárhely tipikusan egy távoli számítógépen van, ami valószínűleg nem is Windows, hanem valamilyen Unix alapú operációs rendszert használ. A weboldal html, css, jpg, stb. fájljait általában egy **FTP kliens** program segítségével tudjuk feltölteni a tárhelyre.

A domain név annak az URL-nek az eleje, amit a böngészőbe beírva megjelenik a weboldalunk. Ha az URL végén nem adunk meg fájlnevet, általában az **index.html** az alapértelmezett fájlnév. Ezért célszerű, ha mi is **index.html**-nek nevezzük el azt a html fájlunkat, amit kezdőoldalnak szánunk.

TÁRHELY ÉS DOMAIN NÉV SZOLGÁLTATÓK

Ha tárhelyre vagy domain névre vágyunk, akkor egy tárhely vagy domain név szolgáltató céghez kell fordulnunk. Az alábbi kettőt ajánlom:

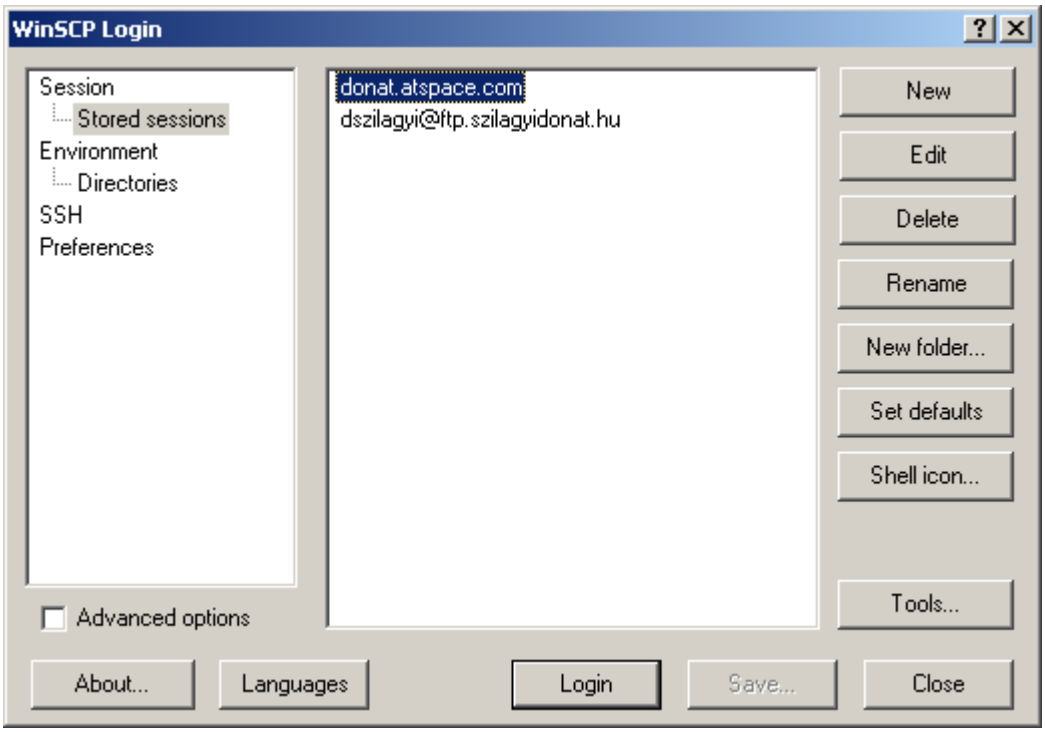
http://www.atspace.com	Angol nyelvű, de német cég. Van ingyenes tárhely és domain név szolgáltatásuk. Az ingyenes domain névnek csak egy részét választhatjuk meg mi. Számítani kell rá, hogy az ilyen ingyenes domain nevek tiltólistán lehetnek a Facebook-on vagy a LinkedIn-in. De indulásnak nagyon jó.
http://manhertz.hu	Magyar, fizetős, de olcsó. Akkor érdemes választani, ha már van egy kész weboldalunk, amit reklámozni is akarunk, például Startlap-on vagy Facebook-on. Külön kell fizetni a tárhelyért és a domain névért, de egy tárhely sok domain nevet is ki tud szolgálni.

WINSCP 

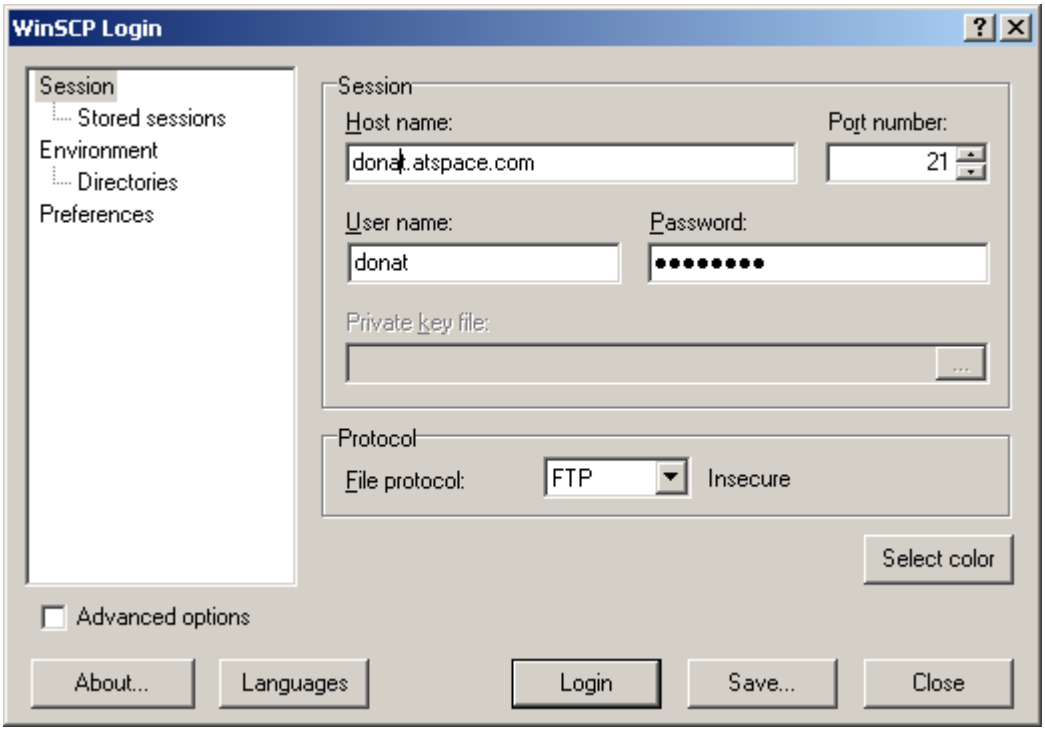
Windows alatt a leggyakrabban WinSCP-t szoktunk használni FTP kliensnek a tárhely eléréséhez.

A WinSCP innen tölthető le: http://winscp.net/eng/index.php

Indítás után kb. így néz ki:

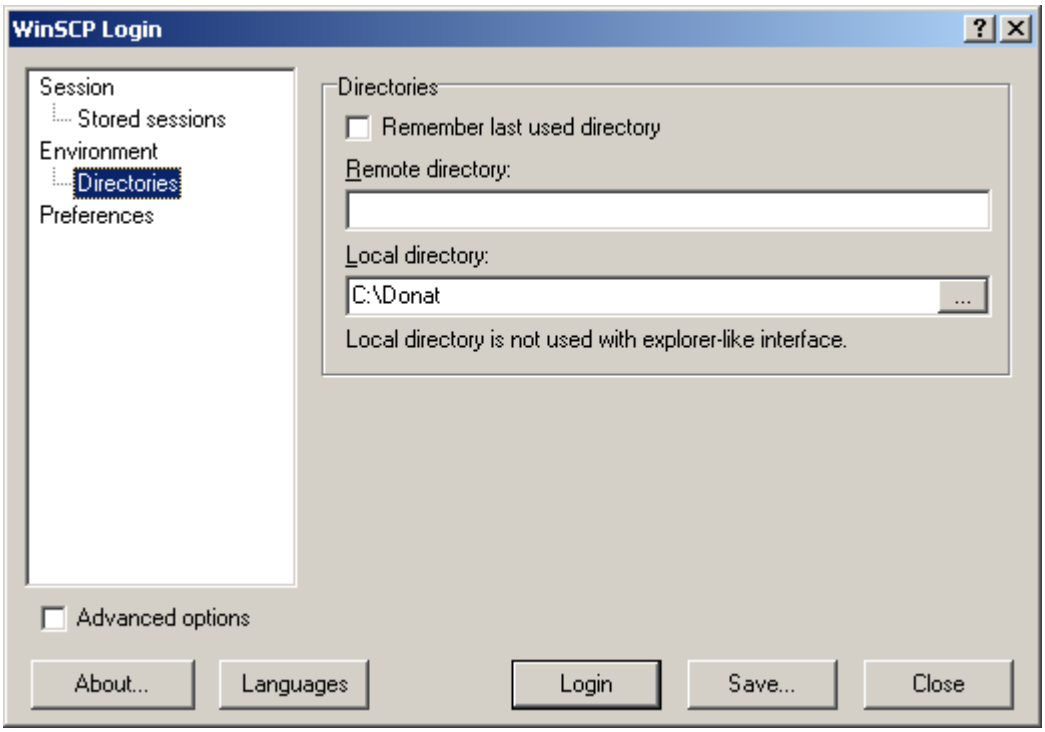


A **New**-val tudunk új kapcsolatot létrehozni. Később pedig az **Edit**-tel tudjuk azt szerkeszteni:



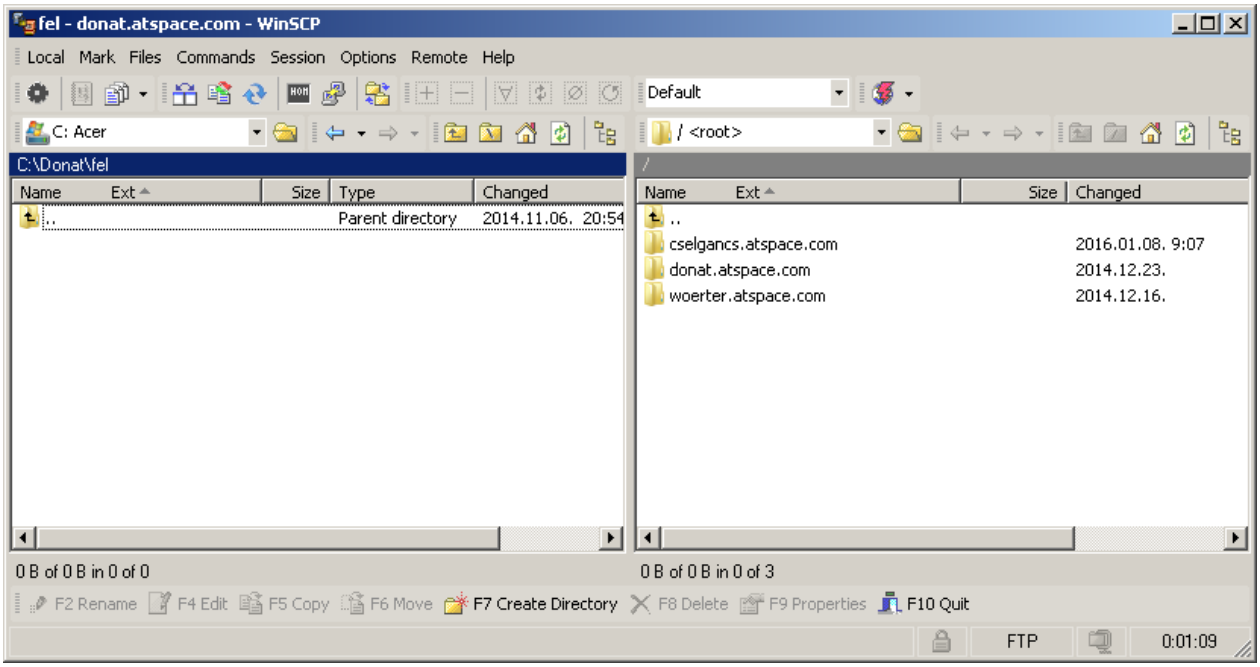
A kitöltendő adatokat a tárhely szolgáltatótól kapjuk meg.

Az Environment / Directories alatt, ha akarjuk, megadhatjuk, hogy alapból melyik könyvtárat nyissa meg a saját gépünkön:



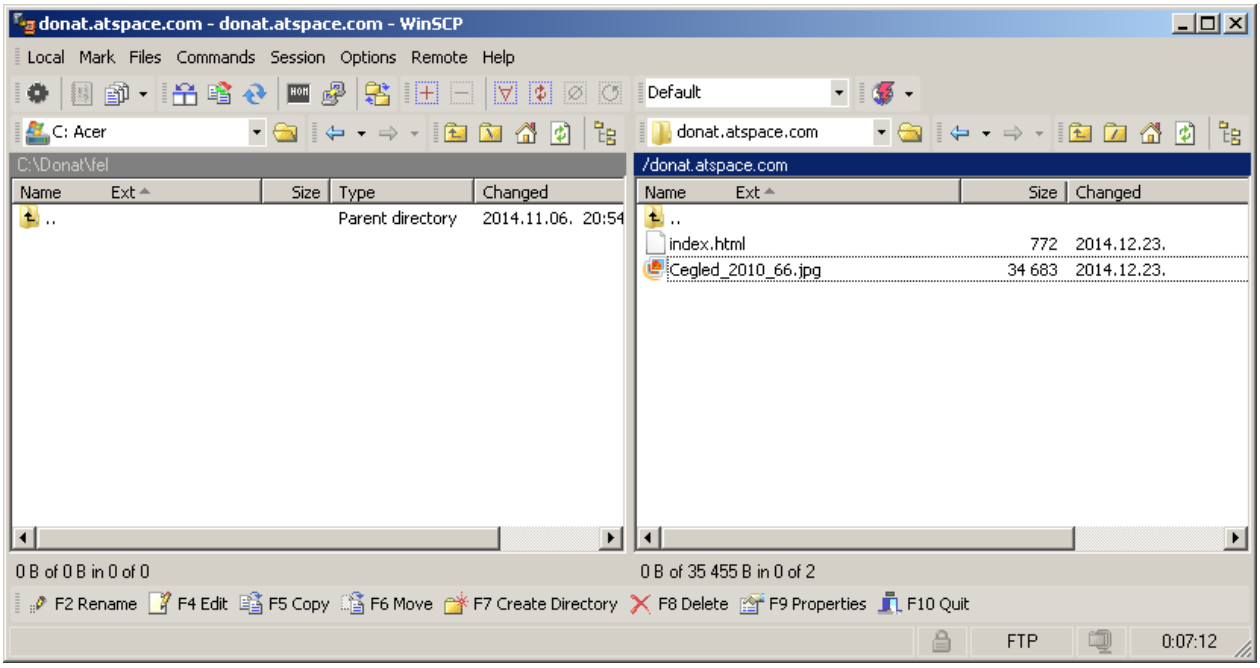
A **Save…** segítségével tudjuk elmenteni a beírt adatokat.

A **Login** segítségével pedig kapcsolódhatunk a tárhelyünkhöz. Utána a **Total Commander**-hez hasonló ablak jelenik meg:



Baloldalt a saját gépünk könyvtárszerkezetét láthatjuk, jobb oldalt pedig a tárhelyünkhöz tartozó domain-ek könyvtárait, amikbe a **Total Commander**-hez hasonlóan bele tudunk menni.

Ha a baloldalon kiválasztunk egy fájlt vagy könyvtárat, azt **F5**-tel tudjuk felmásolni, más néven feltölteni a domain-ünkbe. Feltöltés után a böngészőbe beírva az URL-t, látszódnia is kell a fájlnak.



Például a fenti fájl az alábbi címen érhető el a böngészőben:

http://donat.atspace.com/Cegled_2010_66.jpg

Érdemes egy **index.html** fájllal kezdeni, az akkor is látszik, ha a böngészőbe csak a domain nevet írjuk be:

http://donat.atspace.com

Biztonsági okok miatt inkább csak a saját gépünkről másoljunk a tárhelyre (feltöltés). Visszafelé (letöltés) csak ha nagyon muszáj. Letöltéskor, bár kicsi az esély rá, akár egy vírust is bekaphatunk. Jobb hozzászokni, hogy az interneten ne bízzunk meg senkiben, még a szolgáltatónkban sem.

A WEBOLDAL HIBÁINAK ELLENŐRZÉSE

A W3C nevű szervezet kezeli a HTML és a CSS szabványt. Van egy weboldaluk, ahova feltölthetjük a HTML fájlunkat, és ellenőrzi, hogy van-e hiba benne.

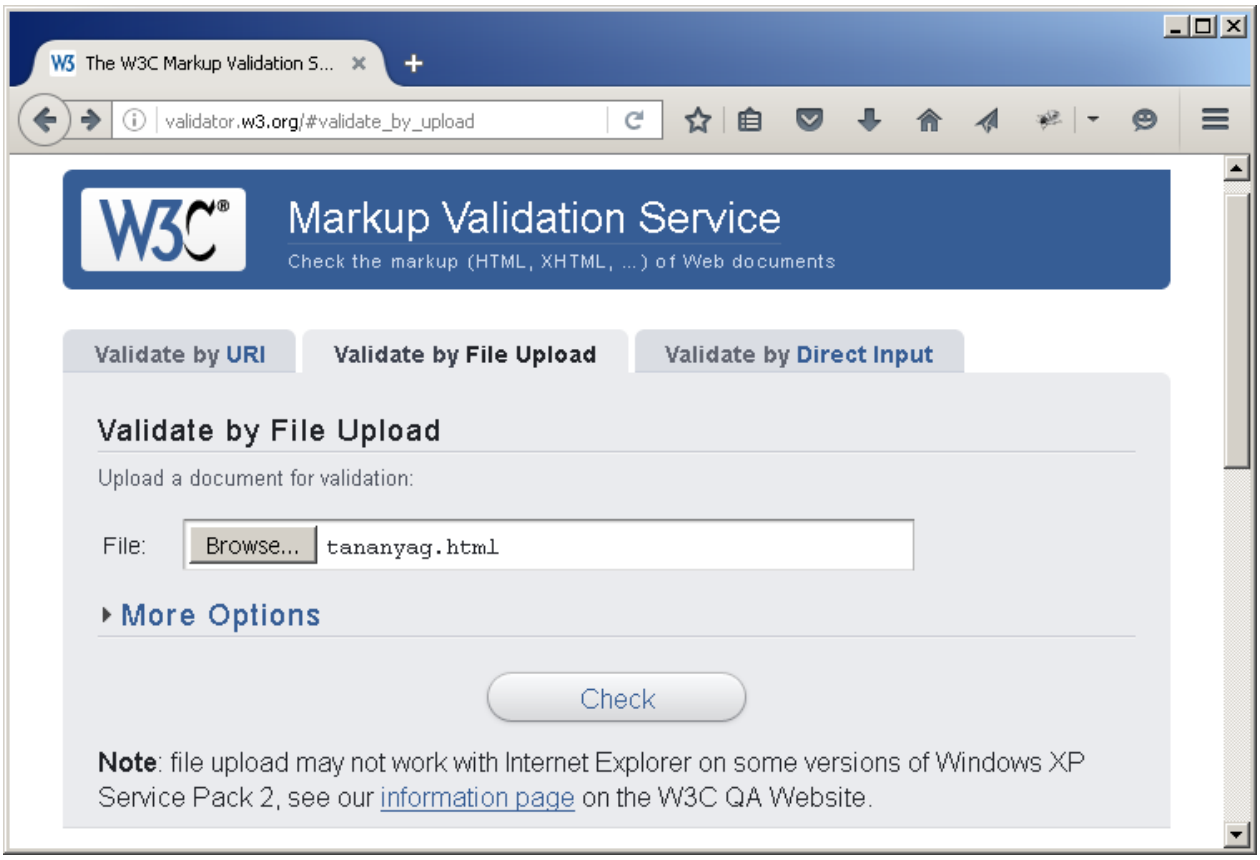
Böngészőbe írjuk be ezt a címet

http://validator.w3.org

Válasszuk ki a **Validate by File Upload** fület.

A **Browse** gomb megnyomása után keressük meg a HTML fájlt, amit ellenőrizni akarunk.

Nyomjuk meg a **Check** gombot.

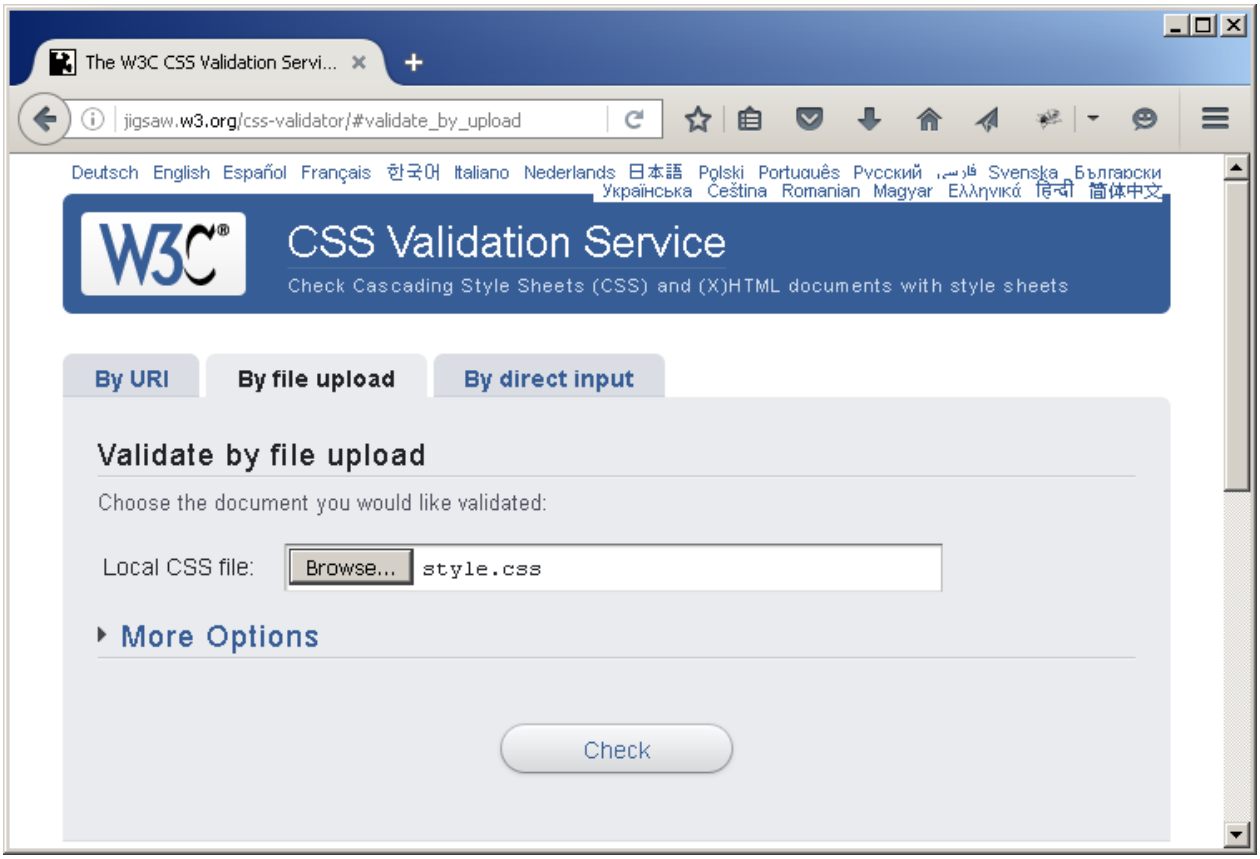


Ezután egy részletes listát kapunk a hibákról.

A W3C-nek van egy CSS ellenőrző oldala is:

http://jigsaw.w3.org/css-validator

Itt válasszuk ki a **By file upload** -ot, **Browse** gomb, CSS fájl kiválaszt, **Check**.



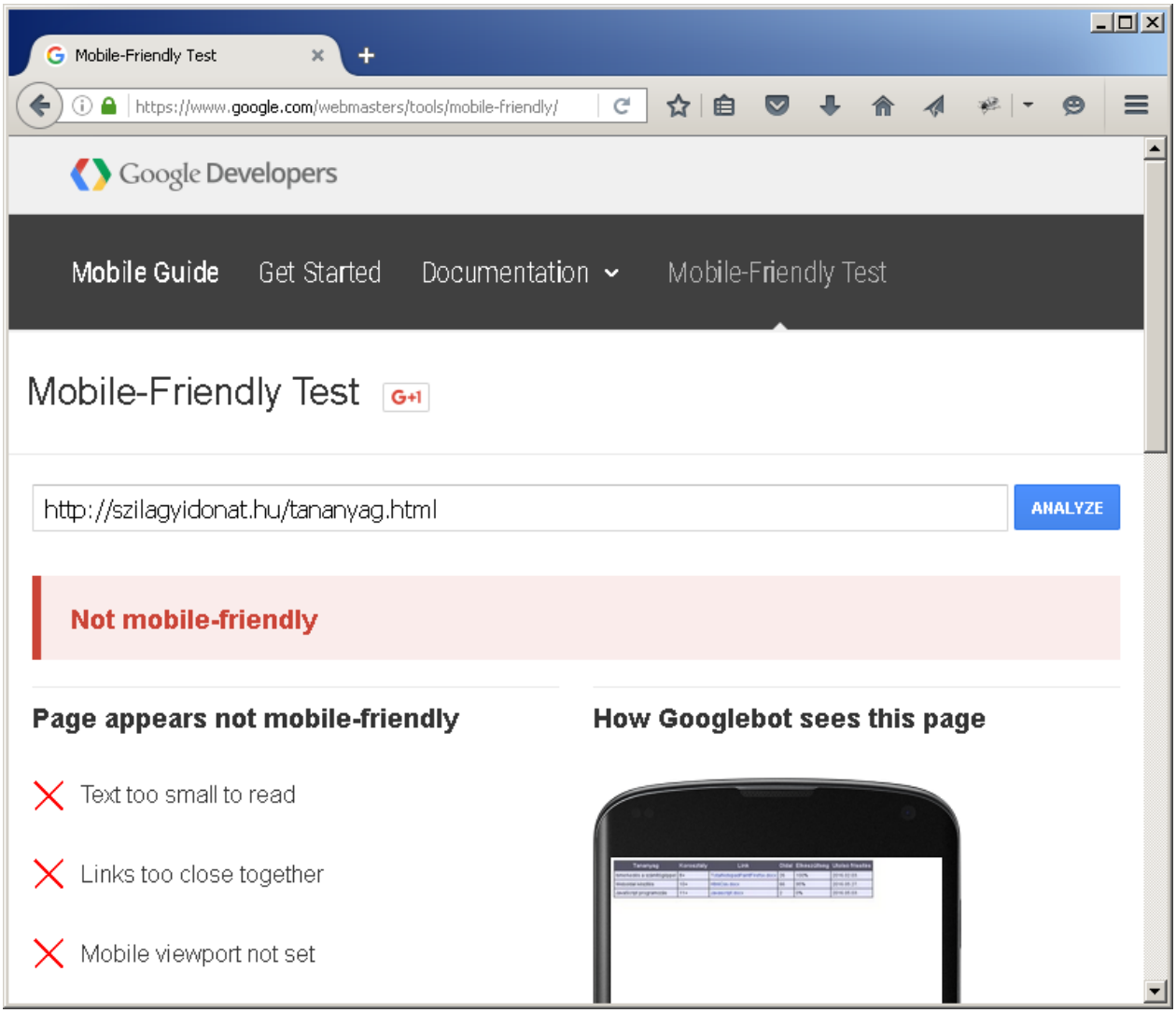
Ezután szintén egy részletes hibalistát kapunk.

Fontos, hogy mindig először a HTML-t kell ellenőrizni, utána a CSS-t.

Célszerű azt is ellenőrizni, hogy néz ki a weboldalunk mobiltelefonon. Ehhez a **Google Mobil-Friendly Test** oldalát használhatjuk:

https://www.google.com/webmasters/tools/mobile-friendly/

Itt egy már élő weboldalt lehet ellenőrizni az URL megadásával. És az **Analyze** gomb megnyomásával.



Minél több hiba van a weboldalunkon, a keresők, pl. Google, annál hátrább teszik a találatok között.

EGYÉB BEÁLLÍTÁSOK A WEBOLDALON

Fontos, hogy a weboldalunknak legyen mindig címe (**title**), leírása (**description**), adjuk meg, milyen betűkészletet használ (**content-type**), milyen nyelvű a weboldal (**content-language**), ki a szerzője (**author**), milyen e-mail címen lehet kapcsolatba lépni velünk (**reply-to**), hány naponta érdemes újra meglátogatni a weboldalt (**revisit-after**), mi a weboldalunk elsődleges URL-je (**canonical**). És ne felejtsük el a mobiltelefon kompatibilitásnál említett **viewport** beállítást se. Itt egy minta:

```
<head>
        <title>Top skills: about the most wanted IT skills</title>
        <meta name="description" content="A page about the most wanted IT skills." />
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <meta http-equiv="content-language" content="en" />
        <meta name="author" content="Szilágyi Donát">
        <meta name="reply-to" content="donat_szilagyi@hotmail.com">
        <meta name="revisit-after" content="15 days" />
        <link rel="canonical" href="http://topskills.eu" />
        <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
```

A **canonical** azért kell, hogy a keresők, pl. a Google észrevegye, hogy ugyanarról a weboldalról van szó akkor is, ha az URL-t **www.** kezdettel gépeli be valaki.

Magyar nyelvű weboldalnál a betűkészlet általában **iso-8859-2**:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

A keresők, pl. Google büntetnek, ha ezek a beállítások hiányoznak.

Van egy **keywords** nevű meta is, azt semmiképp se használjuk, mert ott meg azért büntetnek a keresők, ha van.

Célszerű arra törekedni, hogy a weboldalunk minél kevesebb fájlból álljon. Például a JavaScript-eket és a CSS-eket is érdemes belenyomni a HTML-be. Gyorsabban fog betöltődni a weboldal, és a keresők is szeretik.

A HTML fájlban ne legyen sok fölösleges üres hely, az is lassítja a betöltődést.

A képek mérete fontos, hogy mindig kicsi legyen.

A képekhez mindig adjunk meg címet (**title**), és leírást (**alt**):

```
<img src="cimer/Austria.png" title="Austria" alt="Ausztria">
```

A WEBOLDALUNK REKLÁMOZÁSA

Ha elkészült a weboldalunk, osszuk meg másokkal is az URL-jét, például Facebook-on.

Hirdetőoldalakon is reklámozhatjuk:

http://www.startlap.hu

A Google AdSense oldalát arra is használhatjuk, hogy a saját weboldalunkat reklámozzuk mások weboldalán, jó drágán. Vagy mások weboldalát is reklámozhatjuk a mi oldalunkon, amiért klikkelésenként kaphatunk pár forintot (vagy fillért).

https://www.google.com/adsense

VÉGE