

## Tartalomjegyzék

1. Bevezető foglalkozás.....	3
2. Alapismeretek.....	4
2.1. Alapfogalmak.....	4
2.2. Weboldalak megjelenítési folyamata.....	5
2.3. Ellenőrző kérdések.....	6
3. HTML fájlok és készítésük.....	6
3.1. A HTML fájl.....	6
3.2. HTML fájl készítése.....	7
3.3. HTML fájlok tartalma.....	7
3.4. HTML fájlok megjelenítése.....	8
3.5. Weboldalak forrása.....	9
3.6. Ellenőrző kérdések.....	9
4. HTML alapelemei.....	10
4.1. A HTML nyelv alapelemei.....	10
4.2. A TAG.....	10
4.3. HTML fájl szabványos felépítése.....	10
4.4. HTML fájl kódolása.....	13
4.5. Ellenőrző kérdések.....	14
5. Szabványok.....	14
5.1. Szabványok.....	14
5.2. Tartalom és kinézet.....	15
5.3. Kódjavítás.....	15
5.4. Validálás, érvényesítés.....	16
5.5. Alkalmazott alapelvek.....	16
5.6. Ellenőrző kérdések.....	17
6. Tárhelyek és kezelésük.....	17
6.1. Tárolási lehetőségek.....	17
6.2. Saját tárhely létrehozása.....	19
6.3. Saját tárhely kezelése.....	20
6.4. Ellenőrző kérdések.....	20
7. Karakterformázás.....	20
7.1. A karakter és formázása.....	20
7.2. Karakterformázási lehetőségek.....	21
7.3. Ellenőrző kérdések.....	22
8. Bekezdésformázás.....	23
8.1. Bekezdések.....	23
8.2. HTML kód tördelése.....	24
8.3. Címsorok.....	25
8.4. Szakaszok a weblapon.....	26
8.5. Ellenőrző kérdések.....	26
9. Színek és képek.....	27
9.1. Színek kezelése.....	27
9.2. Képek kezelése.....	28
9.3. Ellenőrző kérdések.....	30
10. Komplex feladat.....	31
10.1. Minta.....	31
10.2. Leírás.....	31
11. Web szerkesztők.....	32

11.1. Web lapok készítésének lehetőségei.....	32
11.2. NotePad++.....	32
11.3. PsPad editor.....	33
11.4. NVU, KompoZer, SeaMonkey webszerkesztő.....	34
11.5. Dreamweaver.....	36
12. Web szerkesztők használata.....	36
12.1. NotePad++.....	36
12.2. PsPad editor.....	38
12.3. NVU, KompoZer, SeaMonkey webszerkesztő.....	39
12.4. Dreamweaver.....	41
13. Hivatkozások.....	44
13.1. Hivatkozások másik oldalra.....	44
13.2. Hivatkozás az aktuális oldalon belül.....	45
13.3. Hivatkozás egy másik oldalon belülrre.....	45
13.4. Képes hivatkozás.....	45
13.5. Ellenőrző kérdések.....	46
14. Felsorolás, számozás.....	46
14.1. Számozott lista.....	46
14.2. Számozatlan lista.....	46
14.3. Többszörös és vegyes listák.....	47
14.4. Ellenőrző kérdések.....	48
15. Teszt.....	48
16. Táblázatok.....	48
16.1. Táblázatok alapjai.....	48
16.2. A táblázat címe.....	49
16.3. Cellák összevonása.....	49
16.4. Táblázatok alkalmazása.....	52
16.5. Ellenőrző kérdések.....	52
17. Táblázatok formázása.....	53
17.1. Teljes táblázatra vonatkozó formázási lehetőségek.....	53
17.2. Táblázatok sorainak formázása.....	53
17.3. Táblázat celláinak formázása.....	54
17.4. Ellenőrző kérdések.....	55
18. Speciális karakterek.....	55
18.1. Alapok.....	55
18.2. Ékezetes betűk kódjai.....	56
18.3. Fontosabb szimbólumok.....	56
18.4. Ellenőrző kérdések.....	57
19. Multimédia.....	57
19.1. Multimédia alapok.....	57
19.2. Multimedia szabványosan, az <object>...</object> elem.....	60
19.3. Ellenőrző kérdések.....	65
20. Komplex feladat.....	66
20.1. A feladat.....	66
20.2. Minta.....	67
20.3. Hozzávalók.....	67
21. A fejléc elemei.....	67
21.1. A fejléc jelentősége.....	67
21.2. Lehetőségek.....	68
21.3. Ellenőrző kérdések.....	73

22. Térképek.....	74
22.1. Mi az a térkép?.....	74
22.2. Térkép img elemmel.....	74
22.3. Ellenőrző kérdések.....	76
23. Keretek.....	76
23.1. Keretek fogalma.....	76
23.2. Kétszlopos minta.....	77
23.3. Keretek meghatározása.....	78
23.4. Beillesztett keretek.....	78
23.5. Ellenőrző kérdések.....	79
24. Keretek alkalmazása.....	80
24.1. Lehetőségek.....	80
24.2. A gyakorlat.....	83
24.3. Ellenőrző kérdések.....	84
25. Űrlapok 1.....	85
25.1. Alapelemek.....	85
25.2. Az űrlap (form) működése.....	85
25.3. A form TAG.....	86
25.4. Űrlapkezelés JavaScript-el.....	88
25.5. Az input elem.....	89
25.6. Ellenőrző kérdések.....	96
26. Űrlapok 2.....	96
26.1. Az űrlap elemei.....	96
26.2. A button elem.....	97
26.3. A select elem.....	97
26.4. A textarea elem.....	99
26.5. Űrlap elemek elhelyezése.....	100
26.6. Ellenőrző kérdések.....	102
27. Teszt.....	103
28. Project feladat.....	103
28.1. Témaválasztás.....	103
28.2. Információgyűjtés.....	103
28.3. Tervezés.....	104
28.4. Elkészítés.....	104
28.5. Beüzemelés.....	105

## 1. Bevezető foglalkozás

Meg kell ismerkedni a Moodle e-learning keretrendszer használatával, lehetőségeivel, valamint az oktatás során használt számítógépes környezettel. Ezen kívül tisztázni kell, hogy otthon hogyan tud mindenki hasonló munkakörnyezetet teremteni magának ahhoz, hogy hatékonyan tudjon tanulni és gyakorolni.

## 2. Alapismeretek

### 2.1. Alapfogalmak

Ahhoz, hogy a továbbiakban meg lehessen érteni az egyes anyagrészeket, szükség van néhány alapfogalom illetve rövidítés jelentésének tisztázására. Ezek a fogalmak a következők:

<b>HTTP</b>	<b>HTTPS</b>	<b>Protokoll</b>	<b>HTML</b>
<b>XML</b>	<b>XHTML</b>	<b>Weblap</b>	<b>Webszerver</b>
<b>URL</b>	<b>Link</b>	<b>Böngésző</b>	<b>Statikus oldal</b>
<b>Dinamikus oldal</b>	<b>CSS</b>	<b>JavaScript</b>	<b>Java</b>
<b>CGI</b>	<b>ASP</b>	<b>ASP.NET</b>	<b>PHP</b>

A fogalomtárban mindegyikről található egy rövid leírás, de az interneten is meg lehet keresni a teljesebb leírásokat.

### 2.1.1. HTTP

„A **HTTP** ( *HyperText Transfer Protocol*) egy információátviteli protokoll a világhálón. Az eredeti célja a HTML lapok publikálása és fogadása volt.

A HTTP egy kérés-válasz alapú protokoll kliensek és szerverek között. A kommunikációt mindig a kliens kezdeményezi.”<sup>1</sup>

### 2.1.2. HTML

„A **HTML** (angolul: *HyperText Markup Language=hiperszöveges jelölőnyelv*) egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált.

Az aktuális változata a 4.01, mely az SGML általános jelölőnyelv egy konkrét alkalmazása.

HTML általában szöveges állományokban található meg olyan számítógépeken, melyek az internethez kapcsolódnak. Ezek az állományok tartalmazzák azokat a szimbólumokat, amelyek a megjelenítő programnak leírják, hogyan is kell megjeleníteni illetve feldolgozni az adott állomány tartalmát. Megjelenítő program lehet egy webböngésző (angolul: web browser), aural böngésző (olyan, amelyik a felhasználónak felolvassa a megjelenítendő szöveget), braille olvasó, amely konvertálja a szöveget braille "formátumba", levelező program (mint például: Mozilla Thunderbird, Microsoft Outlook, Eudora stb.), valamint egyéb eszközök, például mobiltelefon.”<sup>2</sup>

### 2.1.3. URL

„A **webcím**, más néven **URL** (mely a *Uniform Resource Locator* [egységes erőforrás-azonosító] rövidítése), az interneten megtalálható bizonyos erőforrások (például szövegek, képek) szabványosított címe.

Egyetlen címben összefoglalja a dokumentum megtalálásához szükséges négy alapvető információt:

- a protokollt, amit a célgéppel való kommunikációhoz használunk;
- a szóban forgó gép vagy tartomány nevét;
- a hálózati port számát, amin az igényelt szolgáltatás elérhető a célgépen;
- a fájlhoz vezető elérési utat a célgépen belül.

Egy tipikus, egyszerű webcím így néz ki:

<http://hu.wikipedia.org:80/wiki>

Ennek részei:

- A *http* a használandó protokoll. A protokoll neve után kettőspont (:) írandó.

<sup>1</sup> hu.wikipedia.org [honlap] [2010.03.12] <<http://hu.wikipedia.org/wiki/HTTP>>

<sup>2</sup> hu.wikipedia.org [honlap] [2010.03.12] <<http://hu.wikipedia.org/wiki/HTML>>

- A *hu.wikipedia.org* a célgép tartománya neve. Ez elé két perjel (//) írandó.
- A *80* a célgép azon hálózati portszáma, amin kérésünket várja; ez elé kettőspont (:) írandó. Ezt a részt gyakran teljesen elhagyhatjuk, például esetünkben a http protokoll alapértelmezett portszáma a 80.
- A */wiki* a kért elérési út a célgépen. Ez a rész mindig a perjellel (/) kezdődik.

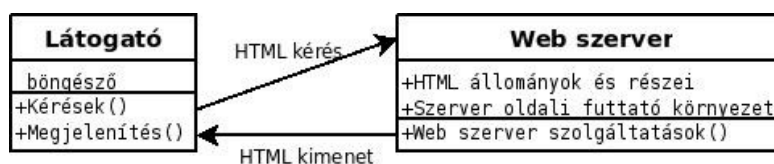
A legtöbb böngésző nem is igényli, hogy a „http://” részt begépeljük egy weblap eléréséhez, hiszen az esetek döntő többségében úgyis ezt használjuk. Egyszerűen begépelhetjük a lap címét, például: „hu.wikipedia.org/wiki/Bit”. A főlap megtekintéséhez általában elég a tartomány nevét beírni, például „hu.wikipedia.org”.

A webcímek egyéb részeket is tartalmazhatnak, http esetében például az elérési út után, egy kérdőjel (?) mögé helyezve keresési kérdés szerepelhet, ami egy *get* metódusú HTML űrlapból származik. Az elérési út után, attól egy kettős kereszttel (#) elválasztva szerepelhet a hiperszöveg egy részére hivatkozó azonosító. Ez az azonosító nem része a webcímnak, de gyakran szerepel vele kapcsolatban.”<sup>3</sup>

## 2.2. Weboldalak megjelenítési folyamata

### 2.2.1. Statikus oldalak

Statikus weboldalak esetén az oldalakhoz tartozó fájlok tartalmát egy az egyben megkapják a böngésző programok. Ha a böngésző kér a kiszolgálótól egy fájlt, akkor azt közvetlenül meg is kapja.

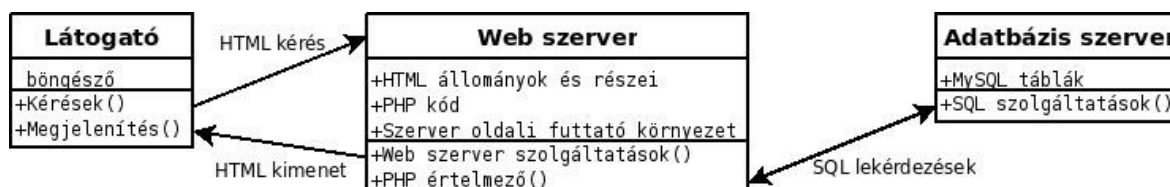


1. ábra Statikus oldalak

### 2.2.2. Dinamikus oldalak

Dinamikus web oldalak esetén a böngésző által kért fájl olyan utasításokat is tartalmaz, amiket a web szerveren kell végrehajtani. A web szerver ilyenkor a kért fájlban lévő utasításokat (scriptek, programok) hajtja először végre. Az ilyen utasítások kimenete általában illeszkedik a HTML nyelvi elemekhez, így a végrehajtás után egy HTML fájlt kap, amit aztán visszaküld a böngészőnek.

A PHP is egy ilyen script nyelv, vagyis, ha a kért oldal fájl tartalmaz PHP kódot, akkor a web szerver meghívja a PHP értelmezőt. Az értelmező végrehajtja az utasításokat, amik akár adatbázis kezelő utasítások is lehetnek, és az eredményt visszaadja a szervernek, aki pedig továbbítja böngésző felé.



2. ábra Dinamikus oldalak

Ebben az esetben a böngésző nem ismerheti meg a PHP kódot, mivel az a szerveren kerül értelmezésre. Ezért is hívják szerver oldali script nyelvnek.

<sup>3</sup> hu.wikipedia.org [honlap] [2010.03.12] <<http://hu.wikipedia.org/wiki/URL>>

## 2.3. Ellenőrző kérdések

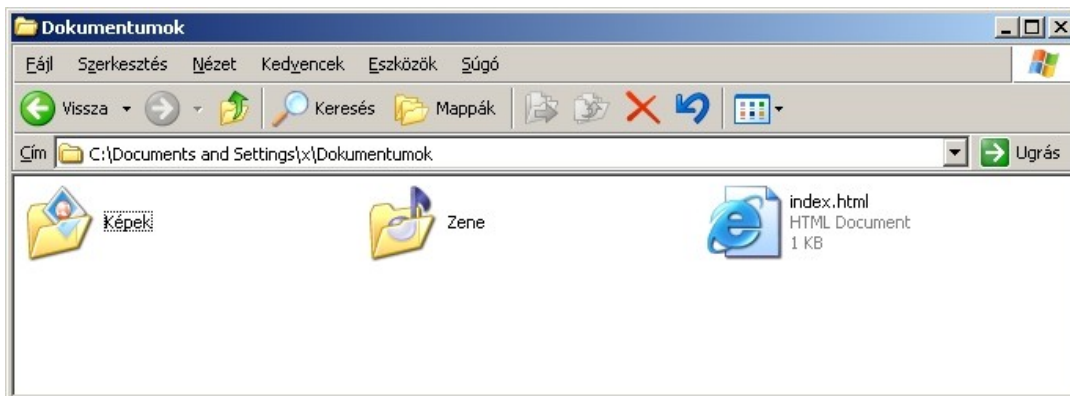
Próbálj meg a lecke megtanulása után önállóan válaszolni az itteni kérdésekre!

1. Mi az azonos a HTML és a HTTP között?
2. Hogyan épül fel egy HTML fájl?
3. Mi a különbség a HTML és a HTTP között?
4. Mi az ASP?
5. Mit jelent, ha egy weboldal dinamikus?
6. Mi a különbség a HTML és a HTTPS között?
7. Szerinted milyen oldalból van több az interneten, HTML vagy PHP?
8. Mi az a JavaScript?
9. Ismertesd az URL-t!
10. Definiáld a protokollt!
11. Sorolj fel néhány olyan protokollt, amik a weboldallal kapcsolatosak!

## 3. HTML fájlok és készítésük

### 3.1. A HTML fájl

A weboldalakat tartalmazó állományok egyszerű szöveges fájlok, amiket bármelyik egyszerű szerkesztő programmal (pl. Notepad) is létre lehet hozni. Az egyetlen megkötés, hogy a fájl kiterjesztése .html vagy .htm legyen. Ma már elsősorban a .html kiterjesztést használják, például: **index.html**.



3. ábra HTML fájlok

Ezek az állományok tartalmazzák azokat a szimbólumokat, amelyek a megjelenítő programnak (böngésző) leírják, hogyan is kell megjeleníteni illetve feldolgozni az adott állomány tartalmát.

### 3.2. HTML fájl készítése

Mivel a HTML fájlok egyszerű szöveges állományok, így Windows alatt a Jegyzetkönyv program segítségével is létre lehet hozni.

Lépések:

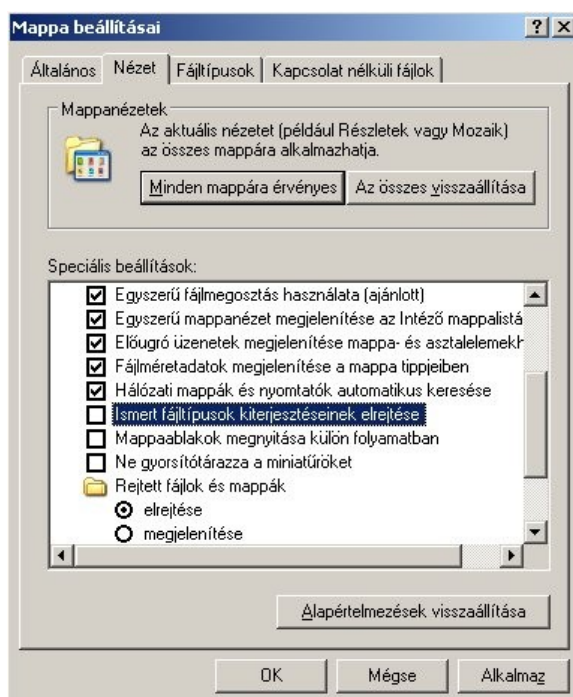
1. Jegyzetkönyv elindítása
2. Mentés másként
3. Hely kiválasztása
4. Név megadása
5. Mentés

### 3.2.1. Munka Windows alatt

Windows operációs rendszer alatt figyelni kell arra, hogy alapértelmezésben a fájlok kiterjesztéseit elrejtí a rendszer. Így ha létrehozunk az asztalon a jobb egérgombbal egy új szöveges dokumentumot **index.html** névvel, akkor annak tényleges neve **index.html.txt** lesz, és sajnos a .txt nem fog látszani.

Az egyik módszer ennek elkerülése érdekében, hogy először elindítjuk a Jegyzetkönyv programot, majd utána mentünk, és a mentésnél átállítjuk a fájlok listázását "*minden fájl*"-ra. Listázáskor azonban így se fog látszani a kiterjesztés.

A másik ajánlott megoldás esetén kikapcsoljuk a Windows elrejtési funkcióját. Ehhez el kell indítani vagy a Sajátgép vagy az Intéző programot. Mindkét esetben az *Eszközök menü*, *Mappa beállításai* menüpontban, a *Nézet* fülön ki kell szedni a pipát az *Ismert fájl típusok kiterjesztésének elrejtése* elől.



4. ábra Ismert fájl típusok kiterjesztésének elrejtése

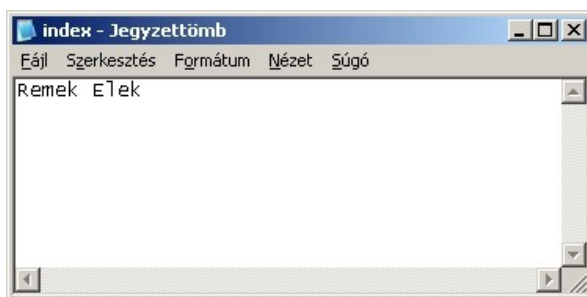
### 3.3. HTML fájlok tartalma

A HTML fájlokban szöveges információk és külső képekre, oldalakra, egyéb elemekre történő hivatkozások, valamint ezek megjelenítését befolyásoló utasítások szerepelhetnek. Ezek mikéntje pontosan szabályozva van, tehát nem lehet akármit akárhogy beleírni ebbe a fájlba. A fájl tartalmát szabványok írják le. A későbbiekben pontosan ezekkel fogunk megismerkedni.

Meg kell azonban jegyezni, hogy a szabványok és a valóság sok esetben nincs szinkronban. Egyrészt több szabvány is készült, másrészt a böngészők nem támogatják a szabványok minden elemét, illetve nem pontosan úgy jelenítik meg az elemeket.

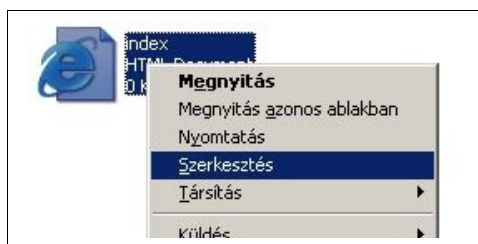
Ki lehet próbálni, hogy az előzőleg létrehozott index.html állományba beleírunk egy tetszőleges szöveget, mondjuk a nevünket. Mentsük is el!





5. ábra index.html jegyzetömbbel

Ha már bezártuk a Jegyzetömböt, akkor újra meg kell nyitni (a fájl nevén jobb egérgomb, szerkesztés).



6. ábra index.html fájl szerkesztése

### 3.4. HTML fájlok megjelenítése

HTML fájlokat böngészők segítségével jelenítünk meg. A legelterjedtebb böngésző programok:

- Internet Explorer 6.0 (IE6)
- Internet Explorer 7.0 (IE7)
- Internet Explorer 8.0 (IE8)
- FireFox 2.x (FF2)
- FireFox 3.x (FF3)
- Opera
- Chrome



IE6



IE7



FireFox



Opera

Egy rendszerben több böngésző is lehet, de mindig van egy kitüntetett, vagyis alapértelmezett. Mindegyik esetében ugyanazok a lehetőségek vannak egy HTML állomány megjelenítésére:

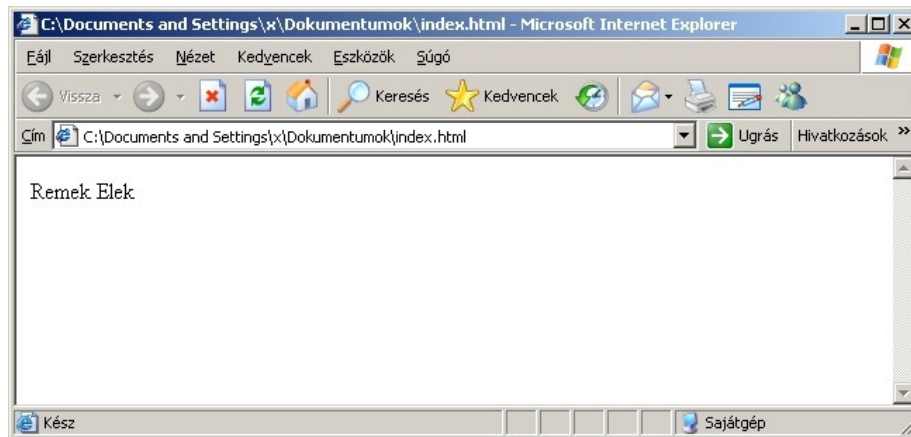
A Sajátgépen duplán kattintani a fájl nevén. Ekkor a beállított alapértelmezett böngésző program elindul, és megjeleníti a fájl tartalmát.

Elindítani a kívánt böngésző programot, majd a *Fájl* menü *Megnyitás* vagy *Fájl megnyitás* menüpontját kiválasztva, meg kell keresni az állományt.

Ha a HTML fájlt már elhelyeztük egy tárhelyen, akkor el kell indítani a böngésző programot, majd a címsorába be kell írni a tárhely címét, és a fájl elérhetőségét.

A nevet tartalmazó **index.html** fájlt IE6-al megjelenítve:



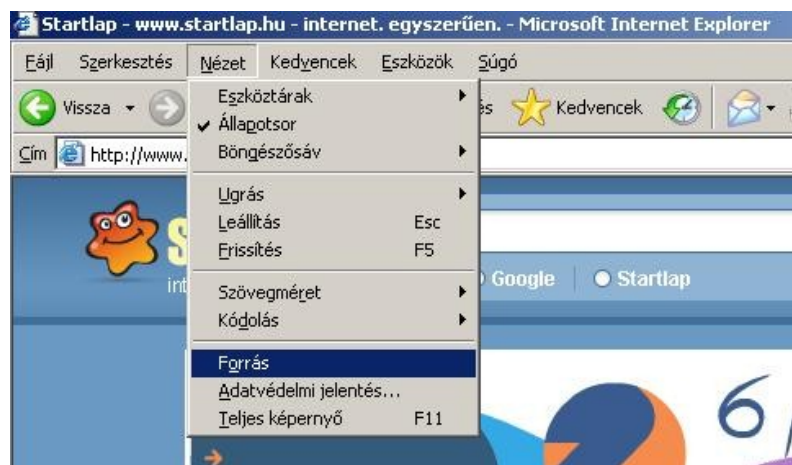


7. ábra index.html IE6-al

### 3.5. Weboldalak forrása

Mindegyik böngészővel lehetőség van arra, hogy a megjelenített oldal HTML forrását is megtekintsük. Firefox esetén a *Nézet* menü *Oldal forrása* menüpont, míg IE6 alatt a *Nézet* menü *Forrás* menüpont jeleníti meg a HTML kódot. Erre nem csak a saját, általunk készített oldal esetén van lehetőség, hanem bármilyen oldal forrását is meg lehet nézni. Ha egy oldal tetszik valamiért, akkor meg lehet nézni, hogy hogyan épül fel, milyen HTML kódot tartalmaz, ezzel is sokat lehet tanulni.

IE6 esetén a startlap.hu oldal forrásának megtekintése:



8. ábra Startlap forrásának megtekintése IE6-al

### 3.6. Ellenőrző kérdések

1. Milyen programot használhatsz web lapok készítésére, ami minden Windows esetén rendelkezésre áll?
2. Mire kell figyelni HTML fájlok készítésénél Windows alatt?
3. Milyen szabványokra kell figyelni web oldalak készítésekor?
4. Hogyan lehet megjeleníteni egy HTML oldalt?
5. Mit tartalmaz egy.html fájl?
6. Lehet-e egy web oldal az **alma.html** fájl?
7. Hogyan lehet megnézni egy internetes weblap forrását?

## 4. HTML alapelemei

### 4.1. A HTML nyelv alapelemei

Ezek az állományok tartalmazzák azokat a szimbólumokat, amelyek a megjelenítő programnak (böngésző) leírják, hogyan is kell megjeleníteni illetve feldolgozni az adott állomány tartalmát.

Négyfajta szimbólum (leíró elem) található meg a HTML-ben:

- *strukturális* elemek, amelyek leírják az adott szöveg "célját" például `<h1>Téma 1</h1>` mint első szintű címsor (alcím).
- *prezentációs* szimbólumok, melyek leírják, az adott szöveg hogy nézzen ki. például `<b>vastag</b>` **vastag** kinézetet eredményez. (Ez a forma azonban ma már *elavultnak* számít, helyette a CSS használata javasolt, ugyanis a legújabb irányelv szerint szét kell választani a tartalmat (amit a HTML kódol) és a formát (amit pedig a CSS).
- *hiperszöveg (hypertext)* elemek, melyek segítségével kapcsolat létesíthető a dokumentum egyes elemei és más dokumentumok között (például a `<a href="http://hu.wikipedia.org/">Wikipedia</a>` a Wikipedia szót mint egy kapcsolatot (angol szóval: link) a megadott URL -hez jeleníti meg)
- *eszköz* elemek, melyek segítségével gombok, listák, beviteli mezők hozhatók létre

### 4.2. A TAG

A HTML elemek a nyelv építőelemei (P=paragrafus; TABLE=táblázat; stb.). Az elemeket úgynevezett **TAG**-ekké (olvasd: teg) alakítjuk a következő jelek segítségével: `<`, `>`, `/`.

A TAG építése roppant egyszerű. Közvetlenül a `<` (TAG kezdő) jel után írjuk az elem nevét, majd a `>` (TAG fejező) jellel lezárjuk. Például: `<elem>` és `</elem>`. Az első a nyitó, a második a záró TAG.

A HTML szabvány keretén belül léteznek olyan TAG-ek, amelyeknek nincs, vagy nem kötelező megadni a záró párját. Az **XHTML** szabvány viszont már ezt is szabályozza.

A TAG-eknek lehetnek jellemzői is, amit a nyitó TAG-en belül lehet megadni, egyszerre akár többet is, szóközzel elválasztva egymástól. Az egyes jellemzők értékeit a jellemző nevét követő = jel után idézőjelek között kell megadni. Például:

```
<p align="justify">A bekezdés szövege</p>
```

Ebben a példában a TAG neve **p**, a beállított jellemző neve **align**, és az **align** jellemző értéke *justify*.

### 4.3. HTML fájl szabványos felépítése

Egy HTML állomány három fő részre bontható:

1. A Dokumentum Típus Definíció az állomány legelején, ami a használni kívánt dokumentum típust (DTD) adja meg, pl:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

A DocType-ot egy sorba kell írni, csak a böngésző, illetve a szerkesztő törli meg a sort. A két idézőjel (") között szóköz van!

2. A HTML fejrész `<head>...</head>` TAG-ek között helyezkedik el, ami technikai és dokumentációs adatokat tartalmaz, melyeket az internet böngésző nem jelenít meg, tehát átlag felhasználó ezeket nem látja és
3. a HTML törzs a `<body>...</body>` TAG-ek között, amely a tényleges, megjelenítendő információkat tartalmazza.

A fejrészt és a tartalmat fogják körbe a `<html>...</html>` TAG-ek.

Tehát egy internetes oldal alapszerkezete a következőképpen nézhet ki:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html lang="hu">
<head>
  <title>Az oldal címe</title>
  <meta http-equiv="Content-type"
content="text/html; charset=iso-8859-2" />
</head>
<body>
  Ide kerül a tartalom, ami megjelenik a böngészőben, mint
jelen esetben ez a sor.
</body>
</html>
```

**Ne felejtjük el, hogy minden HTML fájlnak a fenti szerkezetűnek kell lennie a helyes megjelenítéshez!**

#### 4.3.1. A doctype

A DTD a „Document Type Definition”, azaz a dokumentum típus definíció rövidítése, és sok más dolog mellett ez adja meg azt, hogy milyen elemeket és attribútumokat használhatunk a HTML egyes részeiben. A mai weben több különböző HTML verzió is használatban van. Az aktuális (4.01) szerint a következő három dokumentum típus értelmezett:

##### Strict

A legszigorúbb típus, nem engedi meg, hogy eltérjünk a szabványban leírtaktól. Ebben az esetben nem használhatók a **HTML** nyelv formázó elemei. A fájlban csak *"tisztá"* **HTML** kód van, a formázást stíluslapok (**CSS**) segítségével végezhetjük el.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

##### Transitional

Az előzőhöz képest sokkal engedékenyebb típus. Megengedi a HTML nyelv formázó elmeinek használatát is, és a stíluslapokat is támogatja. Átmenetet képez a hagyományos és a modern HTML kódolás között.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

### Frameset

Keretes oldalak készítésénél használható ez a típus.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"  
"http://www.w3.org/TR/html4/frameset.dtd">
```

**A továbbiakban a Transitional típust fogjuk használni, vagyis minden elkészítendő HTML fájl első sora a Transitional doctype legyen!**

### 4.3.2. A nyelv megadása

A dokumentum nyelvét a `<html>` TAG **lang** attribútumával határozhatjuk meg. Ez segít a megjelenítő programoknak (böngészők) a szövegek helyes megjelenítésében, így ajánlott használni.

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

A **lang** attribútum máshol is használható, nemcsak a `<html>` TAG esetén. Akár bekezdésenként különböző nyelvet lehet beállítani.

A használható nyelvek kódjait a következő címen lehet megtalálni:

[http://www.w3schools.com/tags/ref\\_language\\_codes.asp](http://www.w3schools.com/tags/ref_language_codes.asp)

### 4.3.3. A fejrész

A `<head>` TAG-ekkel határolt dokumentumrészt hívjuk fejrésznek.

A fejrész tartalmazza az oldallal kapcsolatos információkat. A `<head>` tag-ek közötti fejrész a dokumentummal kapcsolatos adatokat (pl. a weboldal címe, nyelve, leírása, kulcsszavak stb.) tartalmazza. A keresők szempontjából kiemelt fontosságú. Hét lehetséges gyermekelemet tartalmazhat. Ezek közül a legfontosabbak:

1. `<title>...</title>`

A böngésző címsorában megjelenő címet adhatjuk meg itt. Pl:

```
<title>A weblap címe</title>
```

2. `<meta ... />`

A `<meta ... />` elemekkel a dokumentum egészére vonatkozó tulajdonságokat állíthatjuk be. Számos további információt is tartalmazhatnak melyeket párokban kell megadnunk, de az oldal ezek legnagyobb része nélkül is hibátlanul működik. Kizárólag a fejlécben engedélyezettek. Ezen elemeknek nincsen zárórésze, de annál hasznosabbak. Több beállítási lehetőség is van. Ezek a közül a legfontosabb a karakterkészlet megadására szolgáló elem:

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

Ezzel állíthatjuk be, hogy milyen karakterkészletet használjon a kliensgép böngészője az oldal megjelenítésekor. Ha az adott karakterkészlet nincs a gépen telepítve, az alapértelmezett karakterkészletét fogja használni. Csak egy karakterkészletet lehet kijelölni.

Magyar oldalakon leggyakrabban az *ISO-8859-2* (közép-európai) és UTF-8-al találkozhatunk. Itt kell még definiálni a böngészőnek azt is hogy egy html dokumentummal van dolga.

#### 4.4. HTML fájl kódolása

Egy szöveges fájl készítésekor nem mindegy, hogy milyen környezetben (milyen operációs rendszer alatt), és milyen programmal készül. Ez ugyanis befolyásolja, hogy a billentyűkhöz milyen kódtábla alapján rendelődnek hozzá a kódok.

Magyar nyelvű Windows operációs rendszerek esetén a *windows-1250* kódtábla az alapértelmezett a különböző programoknál. Vagyis ha a Jegyzetömbbel készítünk egy egyszerű szöveges fájlt, akkor a fájlban a karaktereket a *windows-1250* kódtábla alapján kódolja a program. Ez a kódolás sok hasonlóságot mutat az *ISO-8859-2* táblával, de nem egyezik meg vele teljesen, viszont a magyar ékezetes karakterek ugyanott vannak. Ezért a *windows-1250*-el kódolt szöveg helyesen jelenik meg akkor is, ha *ISO-8859-2* tábla alapján dekódolunk.

Egy böngészőhöz eljutott HTML dokumentum alapértelmezésként *ISO-8859-1*, azaz nyugat-európai kódolást használ. Gyakran előforduló hiba szokott lenni, hogy nincs beállítva a **charset** paraméter a fejléc **content** attribútumában, annak ellenére, hogy a dokumentum nem nyugat-európai kódolású szöveget tartalmaz. Magyar nyelvű oldalak esetén ilyenkor gyakran az „ő” és „ű” betűk helyett „õ” és „ű” karaktereket látunk (ekkor valószínűleg "*ISO-8852-2*" vagy "*windows-1250*" volna a helyes **charset** érték), de az is lehet, hogy minden ékezetes betű teljesen olvashatatlaná válik (ha például a megjelenítendő szöveg UTF-8 kódolású). A **charset** paraméter értékeként több száz kódolás és érvényes alternatív név (alias) megadható.

Az érvényes karakter kódolások listája:

<http://www.iana.org/assignments/character-sets>

##### 4.4.1. Érdekességek a kódlapokkal kapcsolatban

Akiket egy kicsit is érdekel a kódlapok működése, problémái, azok a következő oldalakon olvashatnak utána:

- [Elektronikus szövegtárolás, szövegkódolás](#)
- [Ékezetes betűk: Latin-2 kontra UTF-8](#)
- [Karakterkódolás](#)
- [ISO/IEC 8859-2](#)
- [Windows-1250](#)

#### 4.5. Ellenőrző kérdések

1. Milyen alapelemei vannak a HTML nyelvnek?
2. Mi az a TAG?
3. Hány részből áll egy HTML dokumentum?

4. Hogyan épülnek fel a TAG-ek?
5. Mire szolgál a DOCTYPE?
6. Milyen HTML dokumentumtípusok vannak?
7. Írd le egy üres, tartalom nélküli HTML fájl tartalmát!
8. Miket írhatunk a HTML fájlba a következő elemek közé a pontok helyére?  
/ <head> . . . . </head> /
9. Melyik TAG-ek közé kell elhelyezni az oldalon megjelenő szöveget?
10. Mi a fejrész?
11. Mik azok a kódlapok?
12. Milyen esetben fordulhat elő, hogy a megjelenített HTML oldalon az ő és az ű betű, hullámos?
13. Milyen kódlapok fordulnak elő magyarországon?
14. Melyik az ajánlott kódlap?

## 5. Szabványok

### 5.1. Szabványok

#### 5.1.1. HTML

A HTML-t eredetileg Tim Berners-Lee, a svájci CERN munkatársa fejlesztette ki, tudományos munkák, leírások platformfüggetlen terjesztése céljából. Azóta az internet szabványos leírónyelvévé vált a W3C (World Wide Web Consortium) hathatós közreműködésével. A W3C a világhálóra vonatkozó szabványi előírások létrehozásával és közzétételével foglalkozó szervezet, amely a <http://www.w3.org> címen érhető el.

Jelenleg az aktuális HTML specifikáció a 4.01 verzió, amit a következő címen lehet megtekinteni:

<http://www.w3.org/TR/html401/>.

A HTML 5 munka változata:

<http://www.w3.org/TR/2009/WD-html5-20090825/>

Eltérések listája a 4.01 és az 5 között:

<http://www.w3.org/TR/2009/WD-html5-diff-20090825/>

#### 5.1.2. XHTML

A HTML mellett meg kell említeni az XHTML-t is, mert ez is aktuálisan elfogadott széles körű szabvány, ami XML alapú HTML-nek is tekinthető. Az aktuális verziója az 1.0, és a <http://www.w3.org/TR/xhtml1/> címen érhető el. A HTML 4.01 leváltását várják tőle.

A HTML és az XHTML között gyakorlatilag csak apróbb formai eltérések vannak, ezért érdemes tartani magunkat a szigorúbb előírásokhoz, amit az XHTML határoz meg. Ezek a megkötések a következők:

- Mindent kisbetűvel kell írni (csak a DOCTYPE-ot nem)!
- Minden elemet le kell zárni! Az üres elemeket önmagukban egy szóközzel és egy / jellel: `<br />`.
- Az elemeket csak egymásba ágyazva lehet használni!  
Rossz: `<b><i>Szöveg</b></i>`  
Jó: `<b><i>szöveg</i></b>`
- A jellemzőket idézőjelek közé írjuk!

Rossz: `<table border=1>`

Jó: `<table border="1">`

- A jellemzőknek legyen értéke!

Rossz: `<input disabled />`

Jó: `<input disabled="disabled" />` vagy  
`<input disabled="yes" />`

### 5.1.3. CSS

Még egy fontos szabványt meg kell említeni, mégpedig a CSS-t ami már 2.1-es verziónál tart. A hivatalos oldal címe: <http://www.w3.org/TR/CSS2/>.

A CSS (angolul *Cascading Style Sheets*) a számítástechnikában egy stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le. Ezen kívül használható bármilyen XML alapú dokumentum stílusának leírására is, mint például az SVG, XUL stb.

A CSS-ről tudni kell, hogy gyakorlatilag a HTML kódolás egyik alap építő köve az oldalak formázását, megjelenítését tekintve. Nélküle éppen lehet HTML oldalakat készíteni, de már a HTML szabvány szerint sem ajánlott. A hagyományos formázó elemek már "érvénytelenítettek" (deprecated) a HTML 4.01 szabványban, ami azt jelenti, hogy vagy támogatottak egy böngésző program által vagy sem. Ennek ellenére jelenleg minden böngésző program értelmezi őket a régebben készült oldalak miatt.

## 5.2. Tartalom és kinézet

A szövegszerkesztés esetén is, külön lehet választani a tartalmat a kinézettől (formázás). HTML-ben ez úgy néz ki, hogy a tartalom a tiszta szöveg, a kinézet pedig a szöveg megjelenésének a módosítása. Szokták ezt még prezentációs lehetőségeknek is nevezni.

A HTML fejlődésével egyre inkább nőtt az igény a tartalom és a kinézet szétválasztására. Az aktuális állapot szerint a HTML kód nem tartalmazhat formázó kódot, minden ilyen, megjelenéssel kapcsolatos beállítást CSS segítségével kell megvalósítani. A CSS kód ráadásul egy külön állományban kerül tárolásra, így a tartalmi elemek jól olvashatóak maradnak, és így könnyű javítani is benne.

## 5.3. Kódjavítás

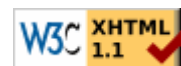
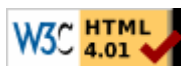
Ha az elkészült kódunk „rendetlen”, vagy sok hibát tartalmaz, akkor jelenthet megoldást a **HTML Tidy**, vagy ha úgy jobban tetszik: a HTML Pucoló. Ez nem más, mint egy letölthető alkalmazás, melyet eredetileg Dave RAGETT fejlesztett ki. A Tidy nem egy szerkesztőeszköz – pusztán dolgunk megkönnyítésére használható, magyarul segít érvényessé tenni oldalainkat. Annyival praktikusabb, mint a W3C HTML kiértékelője, hogy míg ez utóbbi kizárólag a hibák pontos helyét mutatja meg, illetve bizonyos esetekben magyarázó szöveggel segíti a javítást, addig a Tidy bizonyos szintű automatikus javításra képes. A program nemcsak javítani tud, hanem képes különböző kódlapok között is konvertálni, valamint kódtisztításra is alkalmas, illetve HTML kódból jól formázott XHTML kódot is elő tud állítani.

A programot parancssorban használhatjuk "tidy *kapcsoló file kapcsoló file*" formában. Az elérhető lehetőségekről a "tidy -help" parancs kiadásával tájékozódhatunk.



## 5.4. Validálás, érvényesítés

A validátor ellenőrzi a HTML dokumentum szintakszisát, és megnézi, hogy van-e benne valamilyen hiba. A validátor pedig a *doctype* alapján dönti el, hogy milyen szabályokat kell ellenőriznie a dokumentumban. Ez hasonló ahhoz, mint amikor a helyesírás-ellenőrzőnek megmondjuk, hogy milyen nyelven van megírva a dokumentum. Ha nem mondjuk meg neki, akkor nem fogja tudni azt sem, hogy milyen helyesírási és nyelvtani szabályokat kellene ellenőriznie. Ha a HTML dokumentumunk nem tartalmaz *doctype* meghatározást, akkor a böngészőnek kell kitalálnia, hogy mi alapján is értelmezze a tartalmat.



A W3C működtet egy oldalt, ahol ellenőriztetni lehet bármilyen HTML kódot, hogy megfelel-e a szabványoknak. Itt három lehetőség is van:

- megadhatunk egy URL-t
- feltölthetünk egy állományt
- illetve be is lehet másolni egy HTML kódrészletet

Az oldal kiírja a talált problémákat, és még ajánlásokat is tesz. A validátor oldal címe:

<http://validator.w3.org/>

Ha valamelyik szabvány előírásait teljesítette a kód, akkor lehetőség van arra is, hogy különböző képekkel jelezzük a látogatók részére, hogy az oldal szabványos. Ezek az ikonok láthatók a következő oldalon:

<http://www.w3.org/QA/Tools/Icons>

## 5.5. Alkalmazott alapelvek

A modul további részében csak és kizárólag HTML kódokat készítünk, és nem fogunk használni semmilyen CSS vagy egyéb megoldást. Az elemeket tekintve a HTML 4.01 szabvány lesz az alap, szűkítve az XHTML előírásaival. A HTML 4.01 több olyan elemet is tartalmaz, ami *deprecated*-ként (érvénytelenített) van jelölve. Ennek ellenére használni fogjuk ezeket. Emiatt csak a **Transitional doctype** jöhet szóba.

A teljesség kedvéért az alapelvek a következők:

- a doctype HTML 4.01 Transitional a formázásos HTML elemek miatt
- a karakterkészlet ISO-8859-2
- magyar nyelvi beállításokat használunk
- a TAG-eket kis betűvel írjuk, annak ellenére, hogy a HTML 4.01 megengedi a nagybetűs formát is, mivel az XHTML és a HTML 5 is ezt várja el
- minden TAG-et lezárunk
- a zárótag nélküli elemeket is lezárjuk (`<elem />` módon)
- az elemek attribútumainak értéke idézőjelek közé kerül (`<html lang="hu">`)
- az elemek egymásba ágyazásánál vigyázunk a sorrendre (`<x> <y> szöveg </y> </x>`)
- érték nélkül nem használunk attribútumot (`checked="checked"`)
- publikálás előtt ellenőrizzük (validálás) a kódot

## 5.6. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Milyen szabványokra kell figyelni web oldalak készítésekor?
2. Mi a különbség a HTML és az XHTML között?
3. Milyen verziónál tart a HTML szabvány, és mikori ez a szabvány?
4. Mire szolgál a CSS?
5. Milyen program segítségével lehet automatikusan kijavítani a HTML kódban lévő hibákat?
6. Mit értünk validáláson?
7. Hogyan lehet egy elkészült oldalt érvényesíteni?
8. Sorold fel a HTML oldalak készítésekor alkalmazott alapelveket!

## 6. Tárhelyek és kezelésük

### 6.1. Tárolási lehetőségek

Egy weblaphoz több állomány is tartozhat. Ilyen állományok a .html fájlok, az oldalon megjelenő képek, illetve a videók és zenék is. Ezeket együtt kell kezelni a helyes megjelenítéshez. A fájlok elhelyezésére több lehetőségünk is van:

#### 1. Csak saját számítógépen tároljuk

Amíg csak saját szórakoztatásunkra készítünk oldalakat, addig az oldalhoz tartozó állományokat tárolhatjuk a saját számítógépünkön is, a böngésző programokkal ugyanis így is meg lehet jeleníteni őket. Ha azonban a célunk az, hogy mások is elérhessék a művünket, már más megoldás után kell nézni.

#### 2. Ingyenes tárhely szolgáltatónál helyezük el (Free Web hosting)

Ahhoz, hogy az általunk készített oldalak mások számára is hozzáférhetők legyenek, el kell helyeznünk azokat olyan helyen, amit bárhonnán elérhetnek. Ezen kívül szükség van web szerverre is, ami majd ténylegesen szolgáltatja is azokat a HTTP protokollon keresztül.

Egyes tárhelyek fizetők, mások nem. Ez utóbbi ingyenes tárhely szolgáltatók rendszerint reklámokat jelenítenek meg az oldalaink tetején, illetve alján. Ha ez nem zavar minket, szabadon használhatjuk szolgáltatásaikat.

#### 3. Fizetős szolgáltatónál helyezük el (web hosting)

Az ingyenes tárhely szolgáltatókhoz hasonlóan használhatjuk ezeket a tárhelyeket. A különbség az, hogy nincs reklám csík, valamint további kényelmi szolgáltatásokat is szoktak biztosítani. A tárolás díját a szolgáltatótól függően havonta, illetve megegyezés alapján évente is lehet fizetni.

#### 4. Saját szervert üzemeltetünk

##### • Szerver bérlet

Különböző szolgáltatóknál lehetőség van arra, hogy megfelelő díjazás ellenében szervert béreljünk. Ilyenkor a szolgáltató üzemelteti a szervert, nekünk csak a saját szolgáltatásainkra kell figyelni.

##### • Saját szerver elhelyezése egy szolgáltatónál (server hosting)

Ha van felesleges, megfelelő teljesítményű saját számítógépünk, akkor odaszállíthatjuk a választott szolgáltatóhoz, ahol ellenőrzött környezetben tárolhatjuk azt. A szolgáltató csak az áramot és az internet kapcsolatot fogja biztosítani, minden mást nekünk kell megoldani. Persze ezért a szolgáltatásért is fizetni kell.

- **Az otthoni számítógépből készítünk szervert**

A teljesség kedvéért meg kell említeni azt a lehetőséget is, amikor az otthoni saját számítógépen biztosítjuk a web szolgáltatást. Ez azt jelenti, hogy egy szabadon letölthető programot feltelepítve, máris web szerverként használható a számítógép. Ilyen alkalmazás például a Wampserver (<http://www.wampserver.com/en/>). Ahhoz azonban, hogy mások is hozzáférhessenek az oldalainkhoz, további hálózati beállításokra is szükség lehet. Előnye, hogy ingyen van.

A tanulás és gyakorlás során nincs tárhelyre szükség, hiszen az elkészült fájlokat közvetlenül meg lehet nyitni a böngészőkkel. Ennek ellenére mégis azt ajánlom, hogy készítsen mindenki saját magának legalább egy bemutatkozó, nyilvános oldalt. Teljesen más érzés az, amikor bárki megtekintheti az elkészült oldalt, vagy csak saját maga tudja megnézni egy adott számítógépen.

### 6.1.1. Szolgáltatások

A tárhely szolgáltatók azon túl, hogy elhelyezhetjük az általuk üzemeltetett szerveren a fájljainkat, további szolgáltatásokat is nyújtanak.

Ilyen szolgáltatás például az, hogy a saját oldalunkat nem csak IP cím (<http://88.151.96.4>) alapján, hanem Domain névvel (például: <http://feleselek.atw.hu>) is elérhetjük. Ezt hívják DNS (Domain Name Service) szolgáltatásnak.

További szolgáltatás lehet az is, hogy nem csak statikus oldalakat helyezhetünk el náluk, hanem dinamikusakat is. Ehhez többnyire a PHP nyelvű fájlok kezelését és a MySQL adatbázis kiszolgálót biztosítják.

A legtöbb helyen e-mail címet is biztosítanak, valamint levélküldésre is lehetőséget biztosítanak.

A szolgáltatások köre szolgáltatónként különbözhet, így érdemes végignézni a kínálatot több szolgáltató esetén is.

Az ATW szolgáltató például ingyenesen a következő szolgáltatásokat biztosítja:



9. ábra Az ATW ingyenes szolgáltatásai

Ugyanez a szolgáltató fizetős változatban már sokkal többet nyújt.

### 6.1.2. Ingyenes tárhely szolgáltatók

Ingyenes magyar tárhely szolgáltatók például:

- Extra: <http://extra.hu>
- Ultraweb: <http://ultraweb.hu>
- Freeweb: <http://freeweb.hu>

- **Atw:** <http://atw.hu>
- **Fee:** <http://fee.hu>

További ingyenes tárhelyek: <http://www.megaupload.com/portal/index.php?cat=736>

Sajnos 2010 március 31.-től megszűnik az extra ingyenessége. Mondjuk már 2 éve nem lehet oda regisztrálni. A fenti oldalak mindegyike magyar oldal. Azonban léteznek külföldi ingyenes tárhelyek is. Ezek hátránya, hogy mindent, legjobb esetben is, angol nyelven kell megérteni, és a kezelőfelület is angol lesz.

Néhány ingyenes külföldi tárhely szolgáltató:

- **000webhost:** <http://www.000webhost.com/>  
Kell hozzá DNS! Ingyen DNS: <http://www.co.cc/>
- **freeweb7.com:** <http://www.freeweb7.com/>
- **byethost.com:** <http://byethost.com/>

További szolgáltatók listája: <http://www.freewebspace.net/free/Free-webhosting>

### 6.1.3. Fizetős tárhely szolgáltatók

Az ingyenes szolgáltatók többnyire fizetős szolgáltatásokat is biztosítanak, persze ezért többet is nyújtanak. Csak fizetős szolgáltató azonban sokkal több van. Mindegyikük több konstrukciót ajánl, amiből a felhasználók, igényeik alapján választhatnak. A díjak évi 3900 Ft-tól, akár 30000Ft-ig is terjedhetnek.

- Fizetős magyar szolgáltatók:
- Hostingnet: <http://www.hostingnet.hu>
- CyberServer: <http://www.cyberserver.hu>
- Webcode: <http://www.webcode.hu/>
- Maxer: <http://www.maxer.hu/>
- DataGlobe: <http://www.dataglobe.eu/hu>
- TeraCom: <http://teracom.hu/>

Szolgáltató keresése esetén a következő oldalak is segíthetnek:

- <http://www.tarhely.hu/>
- <http://forum.cmsaward.hu/index.php?board=22.0>

## 6.2. Saját tárhely létrehozása

A tárhelyek használatához először regisztrálni kell, ahol meg kell adni egy e-mail címet, amire küldenek egy levelet. A levélben egy aktivációs linkre kell kattintani, ami után be lehet jelentkezni az oldalon. Az elhelyezni kívánt fájlokat fel lehet tölteni FTP-n keresztül is, de van olyan is, hogy mindez webes felületen keresztül is elvégezhető.

A fizetős szolgáltatásoknál is hasonló feladatok vannak, azzal a különbséggel, hogy Domain nevet is regisztrálni kell.

### 6.2.1. Tárhely létrehozás az ATW-n

A fontosabb lépések:

- regisztráció indítása (<http://atw.hu>)
- felhasználói adatok megadása
- regisztráció
- aktiváló levél megnyitása
- aktiválás (levélben lévő linkre kattintva)
- újabb e-mail megnyitása (a tárhely elérésének adatait kapjuk itt meg)

- bejelentkezés az adminisztrációs felületbe az atw oldalán
- az oldal adatainak megadása, módosítása

### 6.3. Saját tárhely kezelése

Az elkészült oldal állományait FTP protokollon keresztül lehet feltölteni a tárhely szerverére. Ehhez szükség van egy olyan programra, ami képes FTP kapcsolatokat kezelni. Ilyen ingyenes program például a FileZilla, ami kifejezetten FTP program. Ezen kívül használható erre a célra a szintén ingyenes Unreal Commander, vagy a nem ingyenes Total Commander is.

A sikeres csatlakozáshoz a programokban meg kell adni a tárhely FTP elérhetőségét, valamint a regisztrált nevet, és a hozzá tartozó jelszót.

Néhány tárhely biztosít saját fájlkezelési lehetőséget is, amivel böngészőn keresztül lehet feltölteni az állományokat. Ilyen az extra is. Ezekben az esetekben azonban az a probléma, hogy a feltöltést fájlként külön kell elvégezni. A használatához be kell jelentkezni a szolgáltató nyitó oldalán.

### 6.4. Ellenőrző kérdések

A lecke tanulmányozása után próbáld meg önállóan válaszolni a következő kérdésekre!

1. Értelmezd a tárhely fogalmát!
2. Milyen lehetőségeink vannak web oldalaink elhelyezésére?
3. Sorolj fel néhány ingyenes tárhely szolgáltatót!
4. Milyen szolgáltatásokat biztosíthatnak a szolgáltatók?
5. Milyen ingyenes megoldást ismersz weblapok tárolására?
6. Hogyan kell az elkészült oldalakat elhelyezni a tárhelyen?

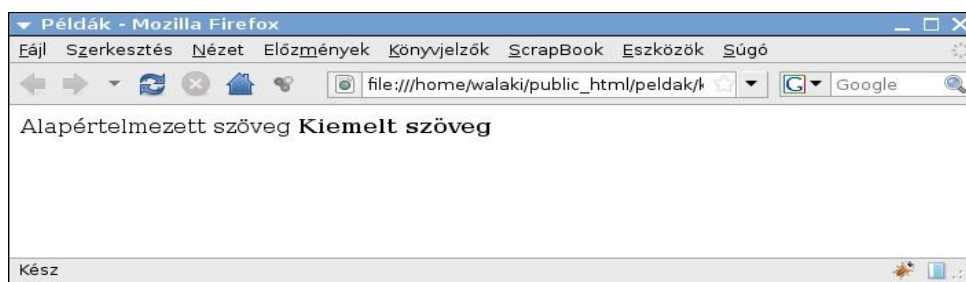
## 7. Karakterformázás

### 7.1. A karakter és formázása

A karakter az oldalon megjelenő szöveg egyetlen betűje, írásjele, vagy speciális jele lehet. A karakterformázás során lehetőségünk van arra, hogy az egyes betűket külön-külön, különbözőképpen formázzuk, vagyis módosítsuk a megjelenés módját.

A formázás során mindig van nyitó és záró TAG is. A kettő között a / jel a különbség. A formázandó karaktert vagy szöveget pedig a nyitó és záró TAG közé kell elhelyezni. Például a következő példában először az alapértelmezett típussal jelenítünk meg két szót, majd a **b** TAG segítségével kiemelten:

```
Alapértelmezett szöveg
<b>Kiemelt szöveg</b>
```



10. ábra Karakterformázás kiemelt szöveg

## 7.2. Karakterformázási lehetőségek

A következő táblázat összefoglalja a karakter formázási lehetőségeket:

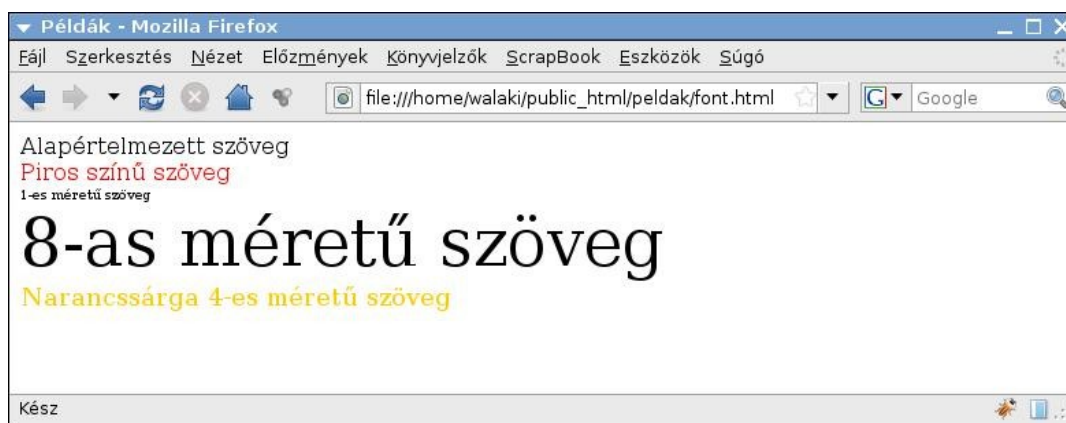
Kezdő elem	Ilyen betűformátumot eredményez	Záró elem
<b>	<b>Félkövér betűformátumot eredményez</b>	</b>
<i>	<i>Dőltbetűs formátumot eredményez</i>	</i>
<u>	<u>Aláhúzott formátumot eredményez</u>	</u>
<s>	<del>Áthúzott formátumot eredményez</del>	</s>
<tt>	Fixpontos betűket eredményez	</tt>
<em>	<i>Kiemeli a szöveget</i>	</em>
<cite>	<i>Idézetekre használható</i>	</cite>
<var>	<i>Változónevet jelöl</i>	</var>
<strong>	<b>Ez is egy kiemelési lehetőség</b>	</strong>
<code>	Kódoknál használjuk	</code>
<kbd>	Billentyűfelirat jelzése	</kbd>
<bq>	Idézet megjelenítése	</bq>
<big>	Nagyméretű betűformátumot eredményez	</big>
<small>	Kisméretű betűformátumot eredményez	</small>
<sub>	Alsóindexet eredményez	</sub>
<sup>	Felsőindexet eredményez	</sup>
<blink>	Villogó szöveget eredményez	</blink>
<dfn>	<i>Melléklet jelzése</i>	</dfn>
<samp>	Program kimenet jelzése	</samp>
<abbr>	Rövidítés jelölése (pl. WWW, HTTP)	</abbr>
<acronym>	Mozaikszó vagy betűszó jelölése (pl. WAC)	</acronym>
<font ...>	<b>A részleteket lásd lentebb</b>	</font>

### 7.2.1. A font TAG

A `<font face="név" color="színkód" size="szám">`, `</font>` utasítaspárral direkt módon előírhatók a megjelenő szöveg betűinek a jellemzői. A **face** opcióval a betűtípust lehet meghatározni (Pl.: Times New Roman, Arial, stb), de nem szokás használni, mert nem valószínű, hogy minden rendszerben rendelkezésre áll az adott betűtípus. A **color** opció pontosan meghatározza a megjelenítendő szöveg színét. A **size** opcióban egy számot megadva a betűméretet határozza meg direkt módon. A **size** opcióban előjeles szám is szerepelhet, ami az alapbetűtípushoz viszonyított méretet jelöl.

A következő példa a `<font>` TAG használatát mutatja be:

```
Alapértelmezett szöveg<br />
<font color="red">Piros színű szöveg</font><br />
<font size="1">1-es méretű szöveg</font><br />
<font size="8">8-as méretű szöveg</font><br /><font
color="#ffcc00" size="4">Narancssárga 4-es méretű
szöveg</font>
```



11. ábra Karakterformázás font

Ebben a példában újdonság a `<br />` elem használata. Ez az elem egy sortörést helyez el, és nincs záró TAG-je. A böngészők ugyanis figyelmen kívül hagyják a forrásfájlban lévő sorvége karaktereket és az ismételt szóközöket. A `<br />` TAG-ek nélkül az előző példa szövegei egymás mellett jelentek volna meg.

Próbáld ki a példát a `<br />` TAG-ek nélkül!

### 7.3. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mit értünk karakter formázáson?
2. Web oldalak esetén hogyan formázhatunk meg egy karaktert? Mivel kell kiegészíteni a szöveget a formázáshoz?
3. Mi a különbség a nyitó és a záró TAG között?
4. Hogyan kell HTML-ben a következő szöveget formázni?  
/ Ez itt a **formázott szöveg**, ennek kell a kódját leírni! /
5. Mit határoz meg a `<tt>` TAG?
6. Melyik a helyes változat?  
`<b>kiemelt szöveg</b>`,  
`<strong>kiemelt szöveg</strong>`,  
`</b>kiemelt szöveg</b>`,  
`<strong>kiemelt szöveg</strong>`
7. Hogyan lehet valamit négyzetre emelni a szövegen belül?
8. Milyen lehetőségeink vannak a font TAG-gel? Miket lehet meghatározni?
9. Írd le a kódját:

/ Ez egy szép színes és érdekes, de annál értelmesebb szöveg! /



10. Keresd meg és javítsd ki a hibát a következő kódrészletben, hogy megfeleljen az XHTML előírásainak!

```
/A mondat <b>itt</i> kezdődik, és <u><b>itt</u></i> <font  
kolor=zöld>végződik. /
```

A / jelek csak a kód elejét és végét jelölik!

## 8. Bekezdésformázás

### 8.1. Bekezdések

Szövegszerkesztő programoknál bekezdésnek nevezik azt az összefüggő többsoros szöveget, amelyet bekezdés jel (ENTER) zár le. A szövegszerkesztésben megkülönböztetjük a sortörést is, ami szintén új sort kezd, de az így elválasztott szövegek egy bekezdésnek számítanak.

HTML-ben a bekezdéseket **p** TAG-ek, nyitó `<p>` és záró TAG `</p>` jelöli. A sortörést pedig a **br** TAG segítségével lehet kikényszeríteni. A **br** TAG-nek nincs záró párja, ezért az XHTML szerint `<br />` módon kell megadni.

```
Bekezdés nélküli sor.  
<p>Az első bekezdés</p>  
<p>Második bekezdés, ami akár több sorosra is tördelhető  
a böngésző program által.</p>  
Következő sor, ami nincs bekezdéshez rendelve.<br />  
Sortörés után egy második sor, még mindig bekezdés  
nélkül.
```

#### 8.1.1. Bekezdés igazítási lehetőségek

A bekezdéseket itt is lehet balra, jobbra, középre és sorkizártan igazítani. Ehhez a `<p>` TAG **align** jellemzőjének kell *left*, *right*, *center* illetve *justify* értéket adni.

```
<p align="left">Balra igazított bekezdés</p>  
<p align="right">Jobbra igazított bekezdés</p>  
<p align="center">Középre igazított bekezdés</p>  
<p align="justify">Mindkét oldalra igazított bekezdés.  
Ehhez viszont olyan sok szövegre van szükség, ami már nem  
fér el egyetlen sorban. Legalább három soros szövegre van  
szükség ahhoz, hogy látható legyen a bal oldali és a jobb  
oldali egymás alá igazítás. Ehhez az is elég, ha a  
böngésző ablak méretét kisebbre állítják. A másik  
megoldás lehet, ha a bekezdés betűméretét  
megnövelnénk.</p>
```

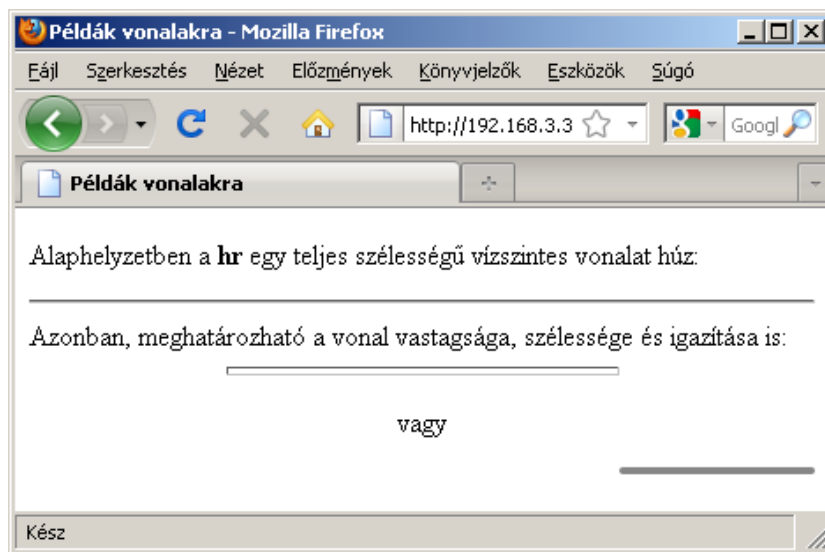
### 8.2. HTML kód tördelése

A HTML oldalak készítésénél meg kell szoknunk azt, hogy a HTML kód tördelése (mennyi üres helyet teszünk, hol nyomunk entert) nem befolyásolja az oldal megjelenését. Ez alól kivételt jelent, ha a `<pre>` taget használjuk. Például:

### 8.2.1. Nem törhető szóköz

hosszt az ablak-szélesség százalékában. A **noshade** pedig térhatást (árnyékoltságot) tiltja le, értéke lehet **true** vagy **false** (igaz vagy hamis).

```
<p align="justify">Alaphelyzetben a <strong>hr</strong>
egy teljes szélességű vízszintes vonalat húz: <hr />
Azonban, meghatározható a vonal vastagsága, szélessége és
igazítása is:</p>
<hr align="center" size="5" width="50%" />
<p align="center">vagy</p>
<hr align="right" size="5" width="25%" noshade="true" />
```



14. ábra Vízszintes vonalak

### 8.3. Címsorok

A címsorok (heading) - mint ahogy a szövegszerkesztésnél is megszokhattuk - az oldalak logikai felosztását teszik lehetővé. A HTML nyelvben ezeket a `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` tagekkel adhatjuk meg. Például `<h1>` az oldal címe, `<h2>` egy alcím, `<h3>` egy mélyebb szintű alcím és így tovább. A HTML oldalak esetén 6 címsort használhatunk. Itt is van kezdő és záró TAG, magát a címet pedig a TAG-ek közé kell elhelyezni.

A címsorokat a bekezdéseknél bemutatott módszerrel igazíthatjuk balra, középre, jobbra, illetve sorkizárttá is tehetjük.

```
Normál szöveg.
<h1 align="center">Első szintű címsor középre
igazítva</h1>
<h2>Második szintű címsor</h2>
<h3>Harmadik szintű címsor</h3>
Ismét normál szöveg.
```

### 8.4. Szakaszok a weblapon

A címek csak a szemlélő számára keltik a tagoltság érzetét, a valóságban nem tagolják fizikailag szakaszokra a dokumentumot. Ezt a tagolást a

```
<div class="osztály" align="hely"> ... </div>
```

utasításokkal lehet meghatározni, ahol a **class** opció sorolja a megfelelő SGML osztályba (lásd CSS) a szakaszt, az **align** pedig a szakasz igazítási formátumát írja elő a bekezdés formázásnak megfelelően (*left, right, center, justify*).

Ennek értelmében a `<div>` TAG-ekkel egyszerre több `<p>`-vel jelölt bekezdést is lehet igazítani. Például:

```
<p>Igazítatlan bekezdés.</p>
<div align="right">
<p>Div tag-en belüli sima bekezdés.</p>
<p>Ez egy másik bekezdés.</p>
<p align="center">Külön középre igazított bekezdés.</p>
</div>
```

## 8.5. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Határozd meg mit nevezünk bekezdésnek!
2. HTML kódban hogyan határozzuk meg a bekezdést?
3. Mi a különbség a `<p>` és a `<br>` TAG között?
4. Miért nem jó a `<br>` ?
5. Mit jelent a sorkizárás?
6. Hogyan kell egy bekezdést középre igazítani?
7. A következő kódban keresd meg a hibát és javítsd is ki!  

```
/<br><p>Ez az első bekezdés!<p><p align="center">Ez már jó lesz ugye?</p>/
```
8. Mire szolgál a `<pre>` TAG?
9. Hogyan lehet ismételt szóközöket írni HTML kódban?
10. Mik azok a címsorok?
11. Hány darab címsor van?
12. Hogyan lehet jobbra igazított címet írni? Írd le a kódot!
13. Szövegszerkesztésben mik azok a szakaszok?
14. Hogyan definiálunk HTML-ben egy szakaszt?
15. Mi lehet egy HTML szakaszon belül?
16. Hogyan lehet egy szakaszt igazítani?
17. Helyes-e a következő kód? Indokold meg!  

```
/<p align="right">Cím</p><div align=justify><p>Szövegelek</p><p align="center">Szöveg helye</div></p>/
```

A / jelek csak a kód elejét és végét jelölik!

## 9. Színek és képek

### 9.1. Színek kezelése

Az eddigiekben csak az oldalon elhelyezett karakterek színének beállítását ismerhettük meg (`<font>` TAG). Azonban további beállításokra is lehetőség van:

- beállítható a teljes oldal esetén az alapértelmezett szöveg színe
- beállítható a teljes oldal háttérszíne
- beállítható az oldal hivatkozásainak színe

Mint látható, ezeket a beállításokat a teljes oldalra egységesen lehet csak beállítani. Nem lehet olyat, hogy csak egy sor vagy egy szó háttérszínét állítsuk be. Ilyen esetekben már a CSS alkalmazására van szükség.

A teljes oldalra vonatkozó beállításokat a `<body>` TAG megfelelő attribútumaival lehet beállítani. A `<body>` TAG fogja körbe az oldal tartalmát. A `<body>` TAG teljes alakja:

```
<body background="fájlnév.kit" bgcolor="színkód"
text="színkód" link="színkód" vlink="színkód"
alink="színkód">
```

**Figyelem! Minden .html fájlban csak egyetlen body TAG lehet!**

### 9.1.1. Alapértelmezett szöveg szín beállítása

Az előzőekben megtanultuk, hogy hogyan lehet a szöveget karakterenként színezni a `<font>` TAG segítségével. Ha egy oldalon elhelyezett minden szöveget ugyanolyan színnel kell megjeleníteni, akkor is használható a `<font>` TAG. Erre a feladatra azonban van egy másik, szebb megoldás is. Mégpedig az oldal tartalmat körbezáró `<body>` TAG-en belüli `text` attribútum.

```
<body text="red">
```

**Figyelem! Minden .html fájlban csak egyetlen body TAG lehet!**

### 9.1.2. Oldal háttérszín beállítás

Az oldal háttérszínét a `<body>` TAG `bgcolor` attribútumával lehet állítani:

```
<body bgcolor="#b9fbfc">
Ebben az esetben az oldalon elhelyezett tartalom világos
kék háttéren fog megjelenni!
</body>
```

### 9.1.3. Hivatkozások színei

A hivatkozások (linkek) a weblapok azon elemei, amelyek egy másik oldalra mutatnak. Alapesetben a link színe világos kék, és alá is van húzva. Ha egyszer már kiválasztottuk, akkor megváltozik a színe lilára. A `<body>` TAG `link` és `vlink` attribútumaival ezeket a színeket lehet módosítani. A következő példa a normál hivatkozások színeit pirosra állítja, míg a már meglátogatott hivatkozások zölden jelennek meg:

```
<body link="red" vlink="#4fb060">
```

## 9.2. Képek kezelése

A HTML oldalakon képeket is elhelyezhetünk. A képek külön állományokban tárolódnak, amire a HTML kódban külön hivatkozni kell. A hivatkozásnál egyszerűen meg kell adni az állomány pontos nevét, kiterjesztéssel együtt.

A böngészők alapvetően háromféle képformátumot támogatnak:

- **jpg**
- **gif**
- **png**

Ezek közül bármelyiket használhatjuk, azonban jó ha tudjuk, hogy:

- A **gif** képek maximum 256 színt tartalmazhatnak, cserébe viszont alkalmasak átlátszó képek készítésére is. A **gif** formátum támogatja az animált képeket is.
- A **png** formátumot régebben nem minden böngésző támogatta, de ma már ez nem probléma. Ez a formátum is lehetőséget biztosít átlátszó képek készítésére.
- A **jpg** formátum veszteséges tömörítést használ, akár 16 millió színnel, de nem támogatja az átlátszóságot.

A web oldalakon a képek elhelyezésére is több lehetőség van:

- Háttérkép teljes lefedettséggel - a képet annyiszor helyezi el a böngésző, ahányszor szükséges, a teljes terület lefedéséhez
- Bármennyi kép az oldal tartalma közé tetszőlegesen elhelyezve
- Háttérkép és egyéb képek egyszerre is elhelyezhetők az oldalon

### 9.2.1. Háttérkép

Az oldal háttérképét a `<body>` TAG **background** paraméterével lehet meghatározni. A képfájl nevét pontosan kell megadni! Itt a pontosság a kis- és nagybetűkre is vonatkozik, mivel Linux alatt az **alma.txt** és az **Alma.txt** különböző fájl!

Arra is figyelni kell, hogy a kép fájl ugyanabban a könyvtárban legyen, mint az oldalt leíró .html fájl. Amikor tárhelyen szeretnénk elhelyezni, akkor is ugyanabba a könyvtárba kell felmásolni a fájlokat. A következő példa almákat jelenít meg a teljes oldalon annyi példányban amennyi az oldal hátterének lefedéséhez szükséges:

```
<body background="alma.gif bgcolor="yellow">
```

A példában a háttérszín is beállításra került ugyan, de a háttérkép eltakarja a teljes hátteret, így sehol nem látszik a sárga háttér!

### 9.2.2. Kép

Képeket az `<img>` TAG segítségével helyezhetünk el az oldalon. A képek elhelyezkedhetnek:

- külön, a szövegtől elkülönülve,
- lehetnek a szöveg mellett balra illetve jobbra is elrendezve,
- valamint a szöveg kezdődhet a kép tetejénél, közepénél és az aljánál.

Az `<img>` TAG teljes alakja a következő:

```

```

Az **align** opció meghatározza a kép igazításának módját, lehetséges értékei: **top**, **middle**, **bottom**, **left**, **right**.

A **hspace** a kép melletti vízszintes térközt, a **vspace** pedig a függőleges térközt (ha úgy tetszik: margókat) határozza meg.

A **width** a szélességét, a **height** pedig a magasságát adja a képnek, az **units** által meghatározott egységben (pixel vagy en).

Az **alt** azt a szöveget adja meg, amely nem grafikus böngészők használata esetén meg fog jelenni a kép helyett.

Az `<img>` TAG-nak nincs záró TAG-je, ezért a következő alakban kell használni:

```

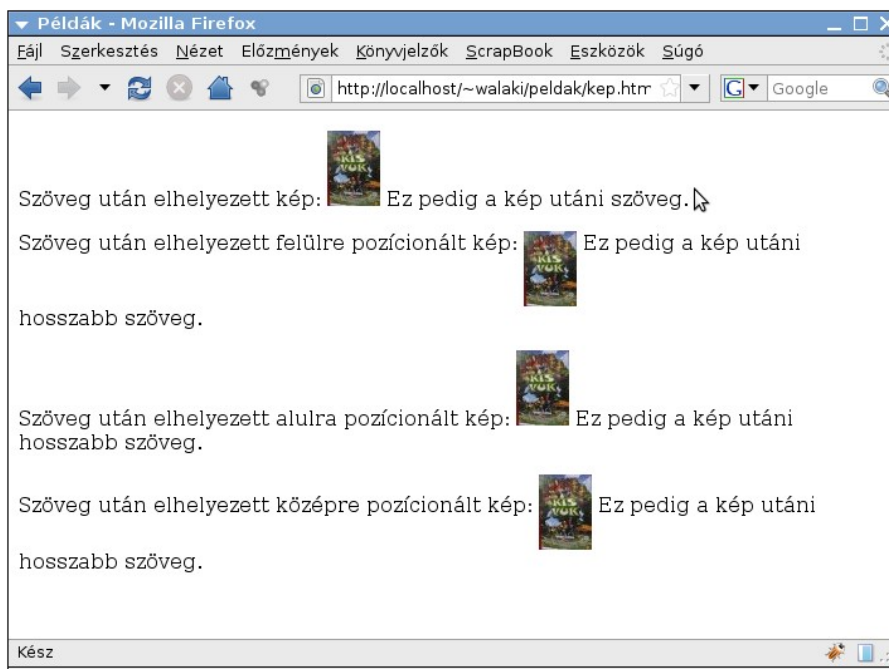
```

Az **alt** attribútumot is használnunk kell minden esetben, ugyanis az XHTML elvárja!

### 9.2.2.1. Függőleges pozícionálás

```
<p>Szöveg után elhelyezett kép: 
Ez pedig a kép utáni szöveg.</p>
<p>Szöveg után elhelyezett felülre pozícionált kép: 
Ez pedig a kép utáni hosszabb szöveg.</p>
<p>Szöveg után elhelyezett alulra pozícionált kép: 
Ez pedig a kép utáni hosszabb szöveg.</p>
<p>Szöveg után elhelyezett középre pozícionált kép: 
Ez pedig a kép utáni hosszabb szöveg.</p>
```





15. ábra Kép függőleges pozícionálása

### 9.2.2.2. Vízszintes pozícionálás

```
<p>Szöveg után elhelyezett balra pozícionált kép:  Ez pedig a kép utáni
hosszabb szöveg, esetleg olyan hosszan, hogy több sorba
tördelődjön.</p>
<p>Ez pedig egy másik bekezdés.</p>
<p>Szöveg után elhelyezett jobbra pozícionált kép:  Ez pedig a kép utáni
hosszabb szöveg, esetleg olyan hosszan, hogy több sorba
tördelődjön.</p>
<p>Ez pedig egy másik bekezdés.</p>
```

## 9.3. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Hogyan lehet beállítani egy web oldal háttérszínét?
2. Mire szolgál a **background** paraméter?
3. Egészítsd ki!  
/`<body ...="ter.jpg" text="...">`/
4. Milyen színeket lehet beállítani a linkekkel kapcsolatban?
5. Mire szolgál az `<img>` TAG?
6. Írd le azt a szabványos kódot, ami az **images** könyvtárban lévő **auto.jpg** nevű képet középre igazítva beilleszti az oldalba úgy, hogy a szöveges böngészők az **auto** szöveget jelenítsék meg!
7. Hogyan lehet egy képet úgy beilleszteni a szövegbe, hogy a szöveg a kép mellett baloldalt több sorban helyezkedjen el?

**A kezdő és záró / jelek csak a kód elejét és végét jelölik!**

## 10. Komplex feladat

### 10.1. Minta

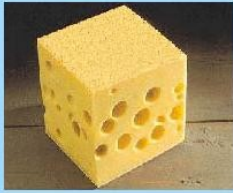
Készítsd el az alábbi mintának megfelelő oldalt! Törekedj a szabványos HTML kód használatára!

## Sajtfélék

Készült: 2009. 11. 09.

### Kemény sajtok


Közös vonásuk a kemény vagy nagyon kemény sajtészta, amely azonban szerkezetében lehet a rugalmas vághatótól a durva-darabosig vagy szemcsésig sokféle. A kemény sajtokat 4 családba lehet osztani: **ementáli**, **parmezán**, **cheddar félek**, **filatasajtok**. Az ementáli jelentős lyukképződés jellemzi. A parmezán nagyon kemény lyuk nélküli, reszelni valók. A cheddar félek főként angliára jellemzők. A filasajtok forrázott és gyúrt sajtok, pl. mozzarella (olasz), kaskaval (Délkelet-Európa).



Az emmentáli sajt

### Vágható, vagy félkemény sajtok

A besorolás nem a sajt állagára vonatkozik, hanem a víz és szárazanyag tartalom arányára. Így a tárolási és érlelési idő előrehaladtával ezekből a sajtokból kemény vagy extra kemény sajtok is lehetnek. Legismertebb tagja a **gouda** (holland). Ma a holland goudában 48% a zsír a szárazanyagban, magas karotin tartalma miatt sárgás színű. Az ugyancsak holland származású **edami** is jeles képviselője ezeknek a sajtoknak. Fűszeres és füstölt sajtspecialitások



16. ábra Komplex feladat minta

### 10.2. Leírás

A feladat részletezése:

1. Az oldalt tartalmazó fájl neve legyen **sajtfelek.html**!
2. Az oldal címe, ami a böngésző címsorában jelenik meg, **Sajtfélék** legyen!
3. Az oldal háttérszíne #A2DAFF, az oldalon lévő alapértelmezett szöveg színe piros legyen!
4. A főcím egyes szintű címsor, míg az alcímek 3-as szintűek.
5. Az oldalon lévő képeket az interneten kell keresni, és le kell menteni a sajtfelek.html fájl mellé!
6. Az oldalon lévő szövegek színei: sárga, zöld, kék, barna.

## 11. Web szerkesztők

### 11.1. Web lapok készítésének lehetőségei

Egy HTML fájl egy egyszerű szöveges fájl, amit minden egyszerű szövegszerkesztővel el lehet készíteni. Az eddigiekben a Windows beépített szerkesztő programját, a Jegyzettömb programot használtuk. Azonban több, kényelmesebben használható program is rendelkezésünkre áll.

Csoportosítva:

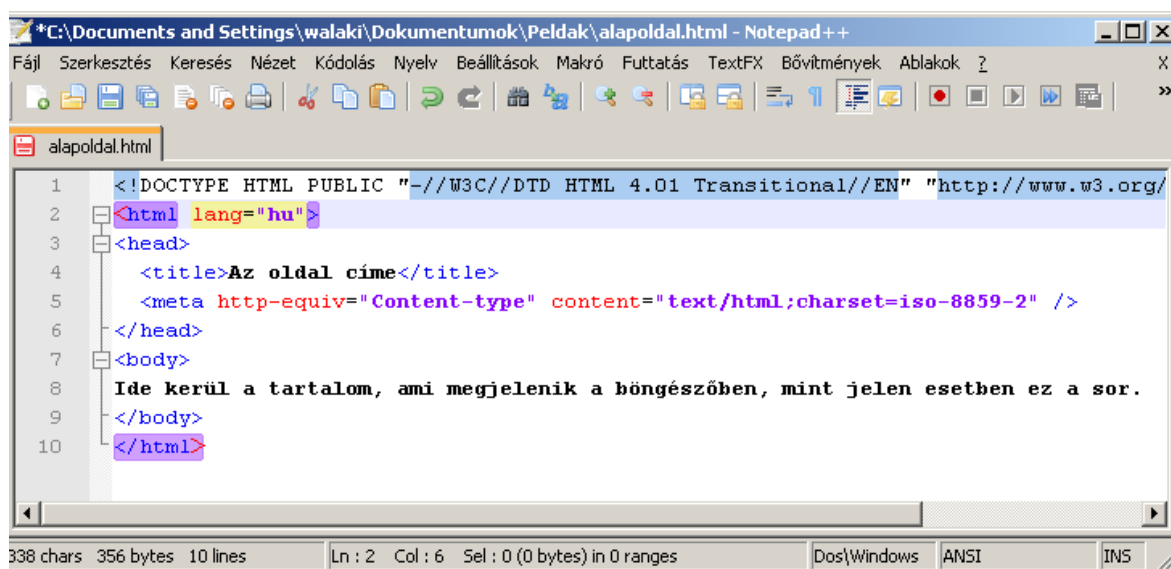
1. Az operációs rendszer beépített szerkesztő programjai, pl.: **Jegyzettömb**
2. *Ingyenes* szerkesztő programok, pl.: **NotePad++**
3. Fizetős szerkesztő programok
4. *Ingyenes professzionális* programok, pl.: **PsPad editor**, **HTML Kit**
5. *Ingyenes WYSIWYG professzionális* programok, pl.: **NVU**, **KompoZer**, **SeaMonkey webszerkesztő**, **Aptana**
6. Fizetős *professzionális* programok, pl.: **Dreamweaver**

Az **ingyenes** kifejezés azt jelenti, hogy a programot bárki letöltheti, és szabadon használhatja.

**Professzionális** program alatt olyan programokat kell érteni, amiket kifejezetten web oldalak készítésére terveztek.

A **WYSIWYG** a web oldalak kialakítását a szövegszerkesztő programokhoz hasonlóan teszik lehetővé. Vagyis nem látszanak a HTML kódok, rögtön az eredményt látjuk, és ezen végezhetünk különböző formázási műveleteket.

### 11.2. NotePad++



17. ábra A NotePad++ program

#### Fontosabb jellemzők:

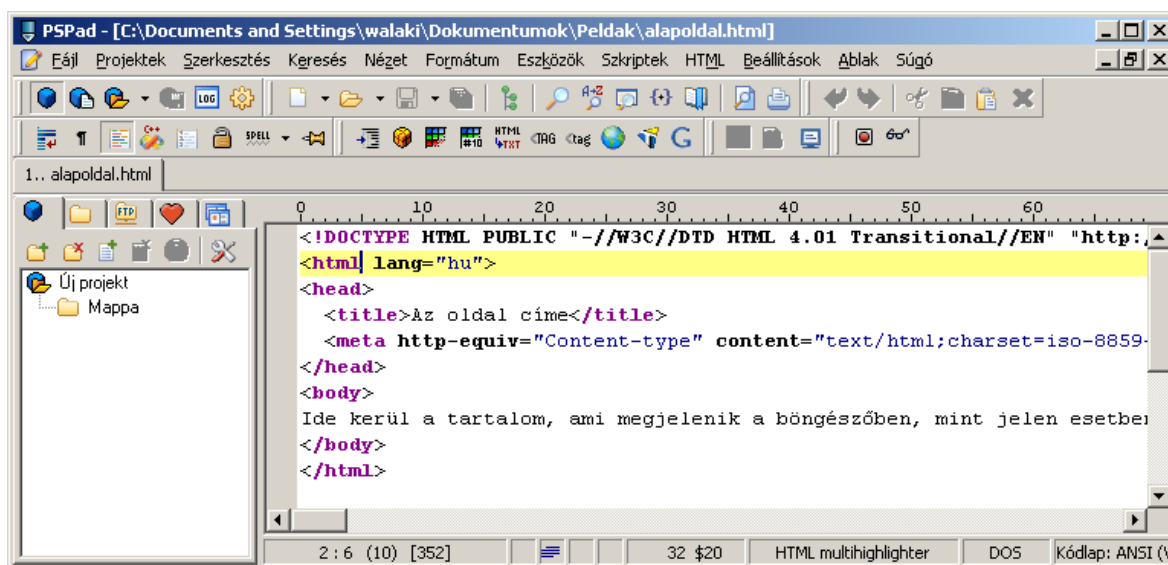
- Ingyenes, nyílt forráskódú, bárki letöltheti és korlátozások nélkül használhatja

- Magyarul is tud
- Egyszerre több fájlt is tud szerkeszteni
- Képes színekkel is kiemelni a különböző nyelvi elemeket (a fájlnev kiterjesztése és a tartalma alapján ez automatikus)
- Sok más programnyelvet is támogat (C, Pascal, C#, PHP, CSS, stb)
- Képes különböző kódlapok közötti átalakításra is
- Csak Windows alatt futtatható

Elérhetőség:

<http://notepad-plus.sourceforge.net/hu/site.htm>

### 11.3. PsPad editor



18. ábra A PsPad editor program

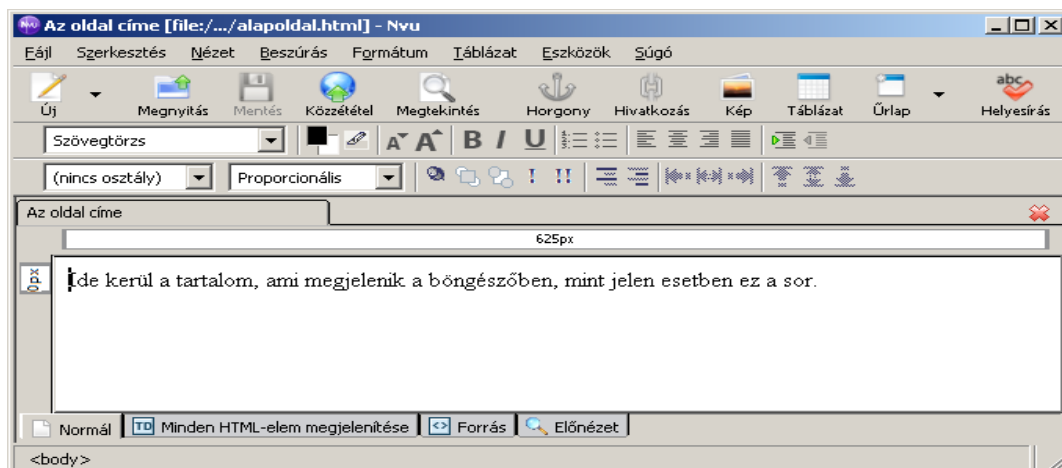
Fontosabb jellemzők:

- Ingyenes, nyílt forráskódú, bárki letöltheti és korlátozások nélkül használhatja
- Magyarítható (külön letölthető a magyar nyelvi fájl)
- Egyszerre több fájlt is tud szerkeszteni
- Képes színekkel is kiemelni a különböző nyelvi elemeket (a fájlnev kiterjesztése és a tartalma alapján ez automatikus)
- Sok más programnyelvet is támogat (C, Pascal, PHP, CSS, stb)
- Beállíthatók különböző kódlapok is
- Sok hasznos kiegészítője van, ami kifejezetten a weboldalak készítésénél előnyös
- Projektek kezelése
- Csak Windows alatt futtatható

Elérhetőség:

<http://www.pspad.com/>

## 11.4. NVU, KompoZer, SeaMonkey webszerkesztő



19. ábra Az NVU program

### Fontosabb jellemzők:

- Ingyenes, nyílt forráskódú, bárki letöltheti és korlátozások nélkül használhatja
- Magyar nyelvi változat is letölthető
- Egyszerre több fájlt is tud szerkeszteni
- WYSIWYG, vagyis a böngészőben megjelenő kinézetet lehet szerkeszteni, szövegszerkesztőkben megszokott funkciókkal
- Lehetőséget biztosít a HTML forrás szerkesztésére is
- Képes színekkel is kiemelni a különböző nyelvi elemeket
- Több operációs rendszer (Windows, Linux, Mac) alatt is futtatható

### Elérhetőség:

<http://mozilla.fsf.hu/nvu-kompozer/>

<http://mozilla.fsf.hu/seamonkey/>

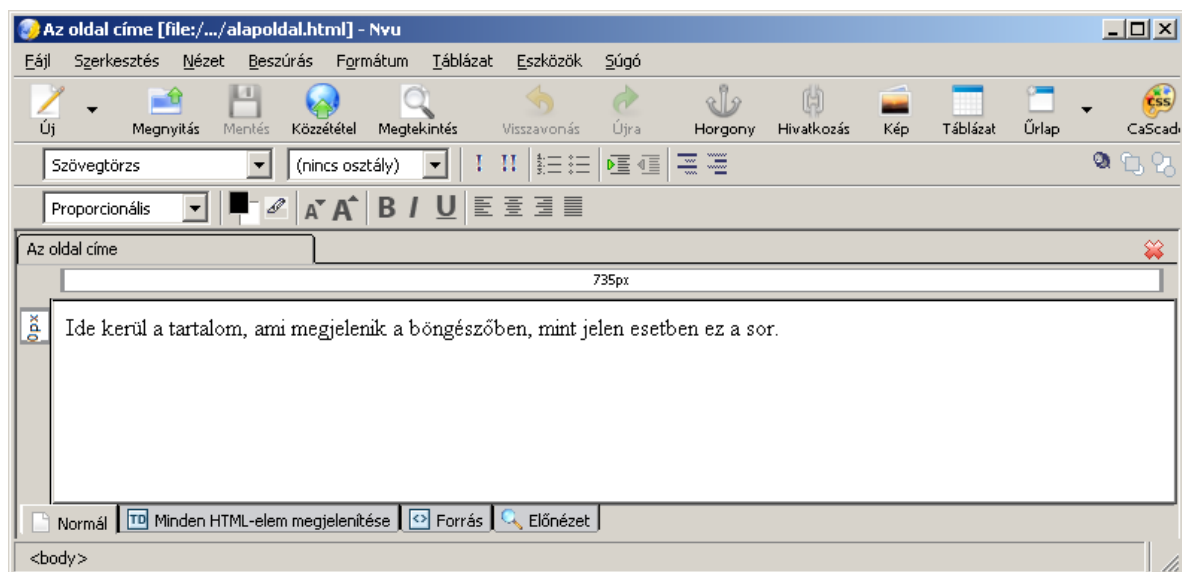
### NVU vagy KompoZer vagy SeaMonkey?

Mindhárom ugyanarra a nyílt forrású program kódra épül, mégpedig a Mozilla programcsomag (Mozilla Application Suite) HTML szerkesztő komponensére. A Mozilla régebben egy komplex programrendszer volt, ami mindent (böngésző, levelező, web szerkesztő) tartalmazott, ami az internet használatához szükséges lehet. A fejlesztő cég (Mozilla Corporation) már leállt ennek a fejlesztésével, ma már csak a különálló programokat (Firefox, ThunderBird) fejleszti.

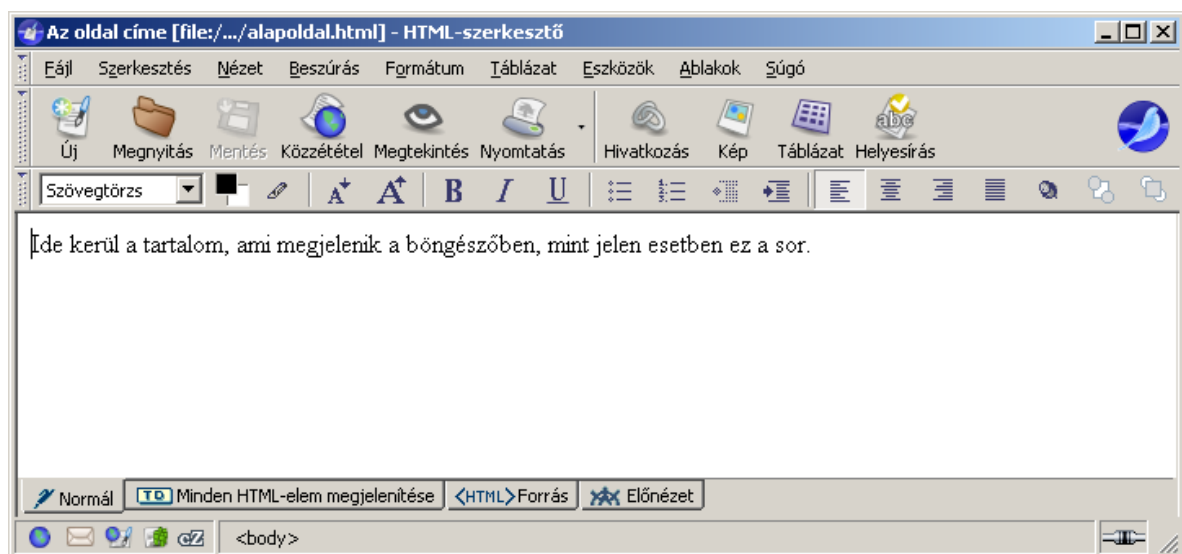
Az NVU program a Mozilla programcsomag webszerkesztő komponensének (Mozilla Composer) tovább fejlesztéseként jelent meg. Az 1.0-s verzió megjelenése után, 2006-ban fejlesztése megállt, és azóta sem fejleszti tovább a készítője.

Egy másik fejlesztő csapat folytatta tovább az NVU-t, de már KompoZer névvel fejlesztik. A fejlesztése jelenleg is folyik. Ma már elsősorban ezt ajánlják WYSIWYG webszerkesztőként.

A SeaMonkey projekt, az eredeti Mozilla elvet követve egy komplex alkalmazás rendszert készít, ami tartalmaz böngészőt, levelezőt, web szerkesztőt és IRC klienst is. Ebben a projektben felhasználják a Firefox, ThunderBird és KompoZer elemeit is. Így a SeaMonkey web szerkesztője majdnem teljesen megegyezik a KompoZer-el.

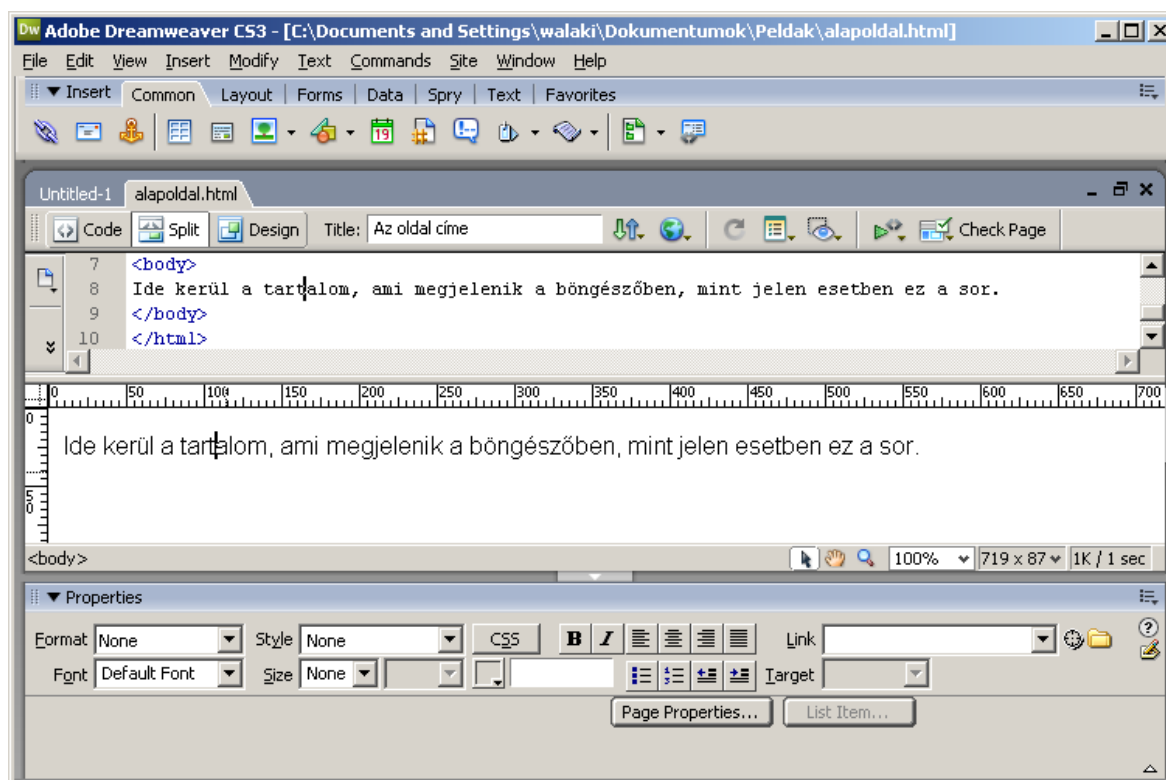


20. ábra A KompoZer program



21. ábra A SeaMonkey program

## 11.5. Dreamweaver



22. ábra A Dreamweaver program

### Fontosabb jellemzők:

- Fizetős, drága
- Nincs magyar nyelvi változat
- Egyszerre több fájlt is tud szerkeszteni
- WYSIWYG, vagyis a böngészőben megjelenő kinézetet lehet szerkeszteni, szövegszerkesztőkben megszokott funkciókkal
- Lehetőséget biztosít a HTML forrás szerkesztésére is
- Képes színekkel is kiemelni a különböző nyelvi elemeket
- Mindent tud, amire bárkinek is szüksége lehet web oldalak készítéséhez
- Csak Windows alatt futtatható

### Elérhetőség:

<http://www.adobe.com/products/dreamweaver/>

## 12. Web szerkesztők használata

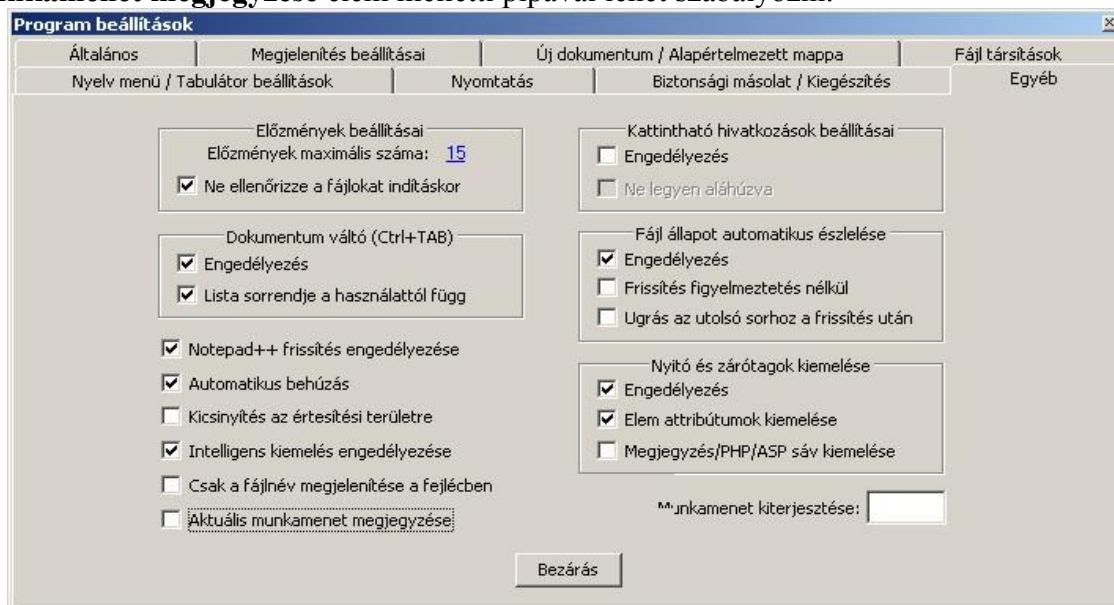
### 12.1. NotePad++

#### 12.1.1. Beállítás

A program nem igényel különösebb beállításokat, nagyon intelligensen ismeri fel a különböző tartalmakat. A Beállítások menüben nagyon sok mindent be lehet állítani, köztük a kezelőfelület kinézetét, gyorsbillentyűket, működéssel kapcsolatos jellemzőket, stb.



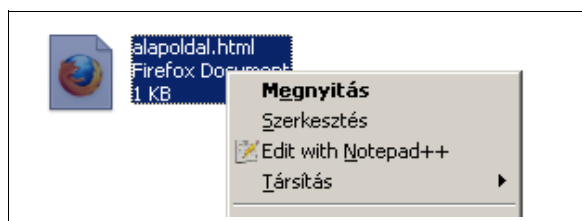
A program régebben mindig megjegyezte az utoljára megnyitott fájlt, és indításkor automatikusan meg is nyitotta. Ez sokszor zavaró tud lenni, ezért ezt a funkciót ki lehet kapcsolni. Az aktuális verzióban már alapban ki van kapcsolva, de ha szükséges, ezt a **Beállítások** főmenüpont, **Program beállítások...** menüpont, **Egyéb** fül **Aktuális munkamenet megjegyzése** elem melletti pipával lehet szabályozni.



23. ábra NotePad++ beállítás

### 12.1.2. Használat

Ha egyszerű szöveges fájlokhoz (text fájlok) szeretnénk használni a NotePad++-t, könnyen megtehetjük, mivel a telepítéskor a rendszerbe integrálta a szerkesztés funkciót. A szerkeszteni kívánt fájl nevére jobb egérrel kattintva, a gyorsmenüben kiválasztható az **Edit with NotePad++** funkció, ami kiválasztása után elindítja a NotePad++ programot, és megnyitja a fájlt is.



24. ábra NotePad++ használata

Windows alatt a fenti példában a Megnyitás gomb hatására a FireFox böngésző programban nyílik meg az alapoldal.html fájl, míg a Szerkesztést kiválasztva Jegyzettömbben válik szerkeszthetővé.

A Társítás menüben módosítható a Windows rendszer úgy, hogy a Megnyitás menüponttal (vagy a dupla kattintással) ne a FireFox nyissa meg a fájlt, hanem a pl. a NotePad++.

A programban nagyon hasznosak a szerkesztési műveletek, amelyeket egyrészt a **Szerkesztés** menüből lehet elérni, másrészt bizonyos funkciók gyorsbillentyűkhöz is hozzá vannak rendelve. Ehhez kapcsolódik a TextFX menü is, amiben nagyon sok kellemes szövegkezelési eszköz van. Ez a rész sajnos nincs magyarítva, de néhányelemére így is szükség lehet.

Szintén érdemes megismerkedni a **Nézet** menü lehetőségeivel is.

Számunkra a két legfontosabb menü a **Kódolás** és a **Nyelv**. Az elsőben beállítható az aktuális kódlap (a *ISO-8859-2* a **Karakterkódolás/Kelet-európai** menü alatt van), valamint átalakításokat is lehet végezni. A másodikban pedig a szerkesztendő fájl programnyelvét lehet beállítani. Erre csak akkor van szükség, ha program valamilyen okból nem ismerné fel azt magától is.

## 12.2. PsPad editor

### 12.2.1. Beállítás

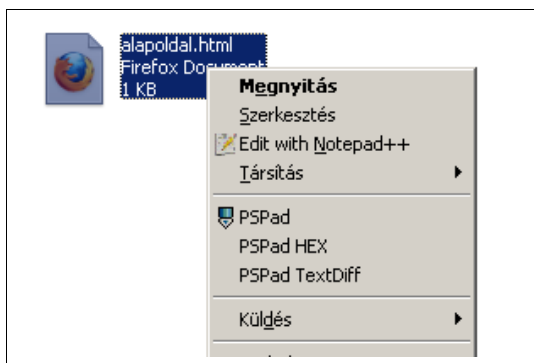
Itt sincs szükség különleges beállításokra, az alapértelmezett beállítások minden szempontból megfelelőek. Ha mégis módosítani szeretnénk valamit, akkor azt a **Beállítások** főmenün belül kell keresni. Egyébként ez a program is teljesen testre szabható, a kinézetet és a működést tekintve is.

Akár a HTML kód színjelöléseit is átállíthatjuk a **Beállítások, Kiemelés beállítások** segítségével, nyelvenként külön-külön.

Ha csak 1 vagy 2 állománnyal dolgozunk egyszerre (egyszerű weblapok készítésekor), akkor a projekt ablakot ki is kapcsolhatjuk a **Nézet** főmenü **Projektszerkezet** menüpontjával vagy a CTRL+F2 billentyű kombinációval:

### 12.2.2. Használat

Ha a PsPad editorral szeretnénk szerkeszteni egy szöveges fájlt, akkor azt a fájl nevén jobb egérgombbal kattintva, és a megjelenő menüből a PsPad elemet kiválasztva tehetjük meg.



25. ábra PsPad használat

A program rengeteg olyan szolgáltatást biztosít, amivel nagyban megkönnyíti a web lapok készítését. Néhány hasznos példa ezek közül:

- Színek listája (**Eszközök** főmenü **Színválasztó** menüpont)
- Szemcseppentő - bármelyik program bármelyik részéből ki lehet emelni egy színt, és rögtön bekerül a szín kódja a fájlba (**Eszközök** főmenü **Szemcseppentő** menüpont)
- A rendezetlen HTML kódot egyetlen mozdulattal jól olvashatóvá lehet tenni (**HTML** főmenü **HTML kód újraformázása** menüpont)
- Előnézet, a beépített böngészővel azonnal ellenőrizni lehet a kinézetet, ráadásul különböző felbontásban is (**HTML / HTML megjelenítés** vagy F10)

- Ellenőrizni lehet a kód helyességét (**HTML / HTML kód ellenőrzése** vagy CTRL+F10)
- Közvetlen validálási lehetőség a Tidy segítségével, ez is kódellenőrzés, de itt már a DTD szerinti szabványosságot is vizsgálják (**HTML / Tidy / Tidy reformat only**)
- Projektek kezelése, ami lehetővé teszi, hogy összetettebb oldalak esetén, minden ehhez tartozó fájlt könnyen lehessen kezelni
- Közvetlen FTP lehetősége a tárhely kezelésére

## 12.3. NVU, KompoZer, SeaMonkey webszerkesztő

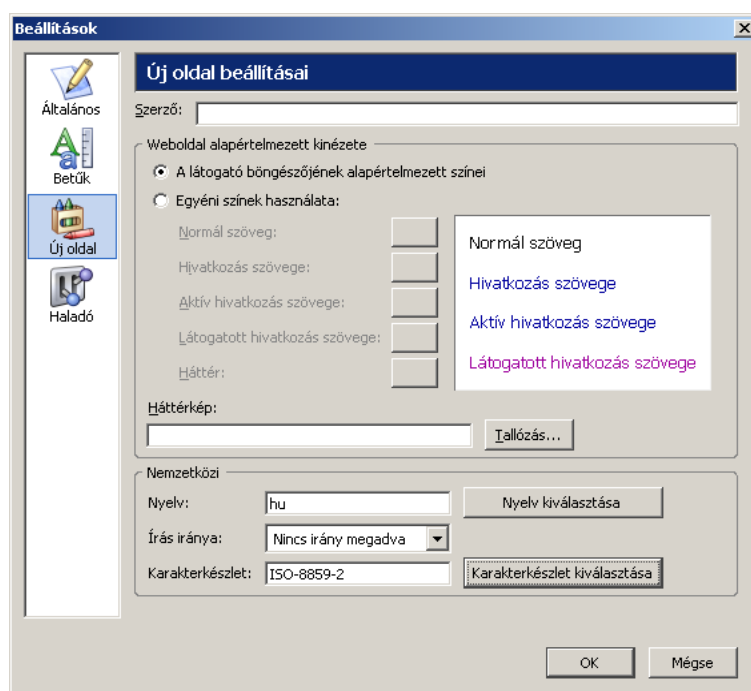
### 12.3.1. Beállítás

A 0.8 verzióval a telepítés után nincsenek további teendők, azonnal használható is.

A 0.7.10-es esetén magyaráítani kell, és be kell állítani a kódlapot is. A magyar kezelőfelülethez:

- Először is le kell tölteni a magyaráítást, pl. [innen](http://innen.kompozer-0.7.10.hu-HU.xpi) (**kompozer-0.7.10.hu-HU.xpi**)
- El kell indítani a programot a **kompozer.exe**-vel
- Ki kell választani a **Tools / Extensions** menüt
- A megjelenő ablakban **Install**
- Majd ki kell választani a magyaráítást tartalmazó fájlt
- Ha lejár a visszaszámlálás, **Install Now**
- Ki kell lépni a KompoZer-ből, és újra kell indítani

Innentől kezdve magyar a kezelőfelület. Sajnos ez a magyaráítás nem állítja be a karakterkészletet és a nyelvet. Az alapértelmezett karakterkészlet és nyelv beállítása az **Eszközök** főmenü **Beállítások** menüpont **Új oldal** fülön az alábbi példának megfelelően:



26. ábra KompoZer alapbeállítások

Ezek a beállítások egy új oldal készítésekor jutnak érvényre. Itt a kódlaapon kívül beállíthatók az új oldal színei is és a háttérkép is.

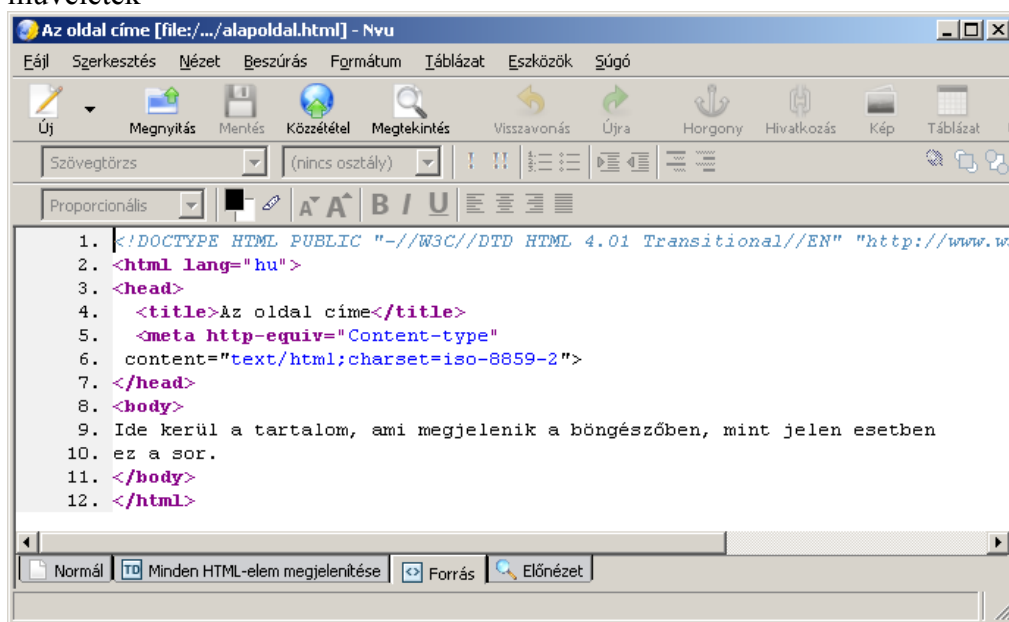
Kezdőknek, akik most ismerkednek a HTML kódolással, érdemes kikapcsolni a stíluslapok használatát a KompoZer-en belül. Ezt az **Eszközök / Beállítások / Általános fül Stílusok használata** elem elől kiszedett pipával lehet megoldani.

### 12.3.2. Használat

A KompoZer használható forrás HTML szerkesztőként is, de az igazi erőssége a WYSIWIG szerkesztés támogatása. Segítségével mindenfajta HTML ismeret nélkül képes bárki HTML oldalakat készíteni. Ehhez mindössze egyszerű szövegszerkesztési ismeretekre van szükség. A karakterformázási és bekezdésformázási műveletek ugyanúgy történik itt is, mint egy szövegszerkesztőben, pl. az eszköztáron lévő gombok segítségével.

A KompoZer-nek 4 fő üzemmódja van, amelyek között gyorsan lehet váltani. Ezek az üzemmódok sorban:

- Normál - nem látszik a forráskód, csak a tényleges tartalom, amelyen különböző formázási műveleteket lehet végrehajtani, ez a WYSIWYG mód
- HTML elemek mutatása - itt se látszik a forráskód, azonban a kód főbb elemeit külön kiemelten mutatja
- Forrás - tisztán HTML kód, itt már nem használhatók a WYSIWYG formázási műveletek



27. ábra HTML forrás a KompoZer-ben

- Előnézet - itt úgy jelenik meg az oldal, ahogyan az egy böngészőben jelenne meg

A normál nézetben történő bármilyen tevékenység azonnali módosulást eredményez a forrásban is, ami annyit jelent, hogy minden esetben újragenerálja a kódot. Sok esetben azonban a generált kód nem a leghatékonyabb.

Bizonyos funkciók használatát meg kell tanulni, de utána könnyen és gyorsan megoldható vele minden. Ezen kívül bármikor átválthatunk a forrás nézetre, ahova bármilyen szabványos HTML kód beírható.

Több oldal is ismerteti az NVU kezelését. Ezek a leírások a KompoZer esetén is érvényesek, hiszen ugyanaz az alapja mindkettőnek.

Egy részletes ismertető található a sulinet oldalán:

<http://www.sulinet.hu/tart/ncikk/Rad/0/29416/index.html>

További leírások:

- [http://www.linuxvilag.hu/content/files/cikk/59/cikk\\_59\\_50\\_55.pdf](http://www.linuxvilag.hu/content/files/cikk/59/cikk_59_50_55.pdf)
- [http://www.hmg.sulinet.hu/vk-szerzok/gaspar\\_bela/webszerkesztes\\_nv.u.pdf](http://www.hmg.sulinet.hu/vk-szerzok/gaspar_bela/webszerkesztes_nv.u.pdf)
- <http://www.studio10.extra.hu/> - itt videók is bemutatják a használatot!

## 12.4. Dreamweaver

### 12.4.1. Telepítés

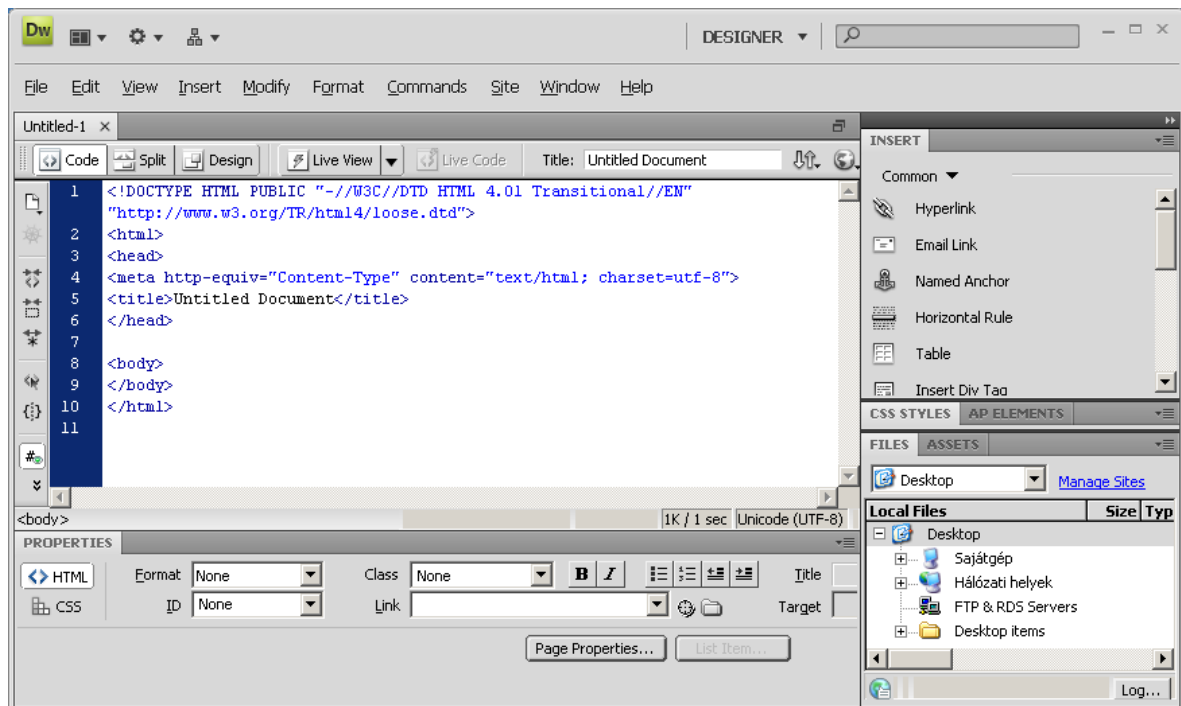
A Dreamweaver fizetős program, amit meg kell vásárolni, mielőtt használnánk. Azonban arra lehetőség van, hogy kipróbáljuk a programot, mielőtt megvásárolnánk. A kipróbálható demó változat a telepítés után 30 napig használható, és közben bármikor aktiválható a megvásárolt program kódjával.

A próbaváltozat letölthető a gyártó oldaláról (<http://www.adobe.com/downloads/>) is, és különböző letöltőközpontokból (pl.: <http://szoftver.hu/download/24>) is. Az aktuális változat a CS4, de a szoftver.hu-ról a régebbi változatok is letölthetők.

A CS4 demó változatának telepítése:

- Az aktuális demó változat egy tömörített .zip fájlban érhető el (**ADBEDRWVCS4\_win.zip**), amit először ki kell csomagolni egy könyvtárba, két fájlt kapunk, egy exe-t és egy .7z tömörített fájlt
- El kell indítani az exe fájlt
- Ki kell választani, hogy hova kerüljön kicsomagolásra a telepítő, majd elindul a kicsomagolás
- A csomagolás után elindul maga a telepítő, ahol első lépésként választhatunk, hogy teljes értékű programként használjuk, amihez eredeti kódra van szükség, vagy csak próbaverzióként, kipróbálási jelleggel
- A következő lépésként el kell fogadnunk a liszenszt
- Választhatunk, hogy mely Adobe termékeket szeretnénk telepíteni még a Dreamweaver-en kívül
- **Telepítés**
- A végén ki kell lépni a telepítőből

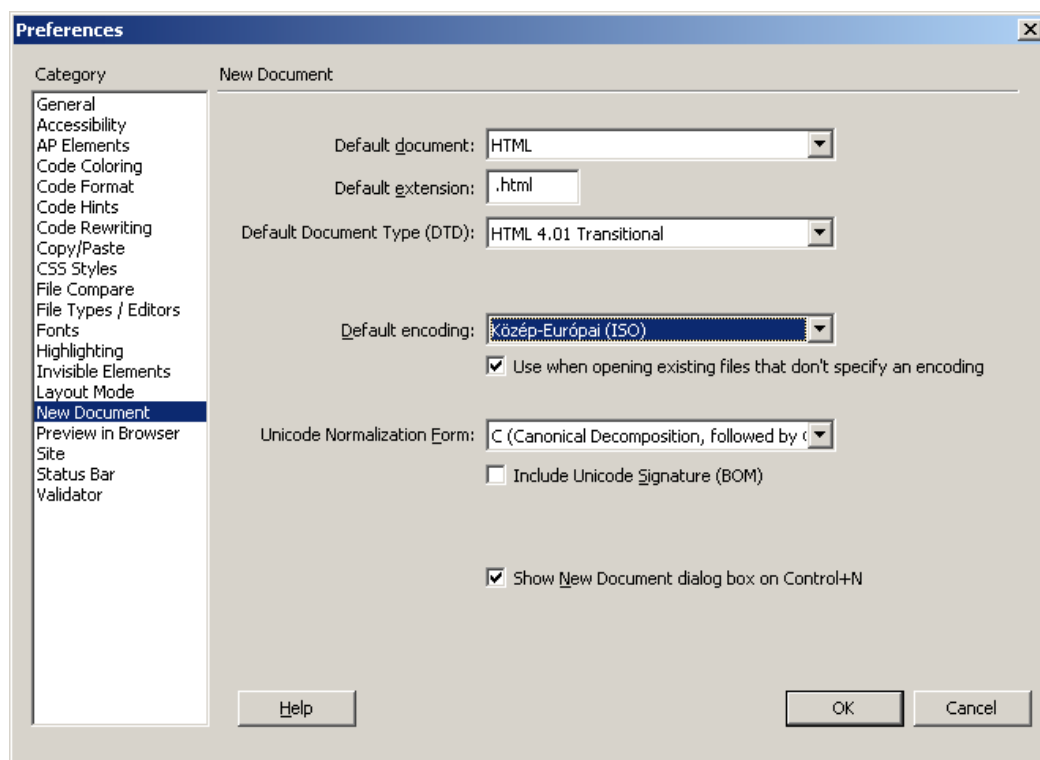
A végeredmény egy teljes tudású, időkorlátos program változat:



28. ábra HTML forrás a Dreamweaver-ben

### 12.4.2. Beállítás

A telepítés után az alapértelmezett karakterkészlet az UTF-8, a DTD pedig az XHTML 1.0 Transitional. Első lépésben érdemes átállítani mindkettőt az igényeinknek megfelelően. Mindkét beállítást ugyanott lehet elvégezni, mégpedig az **Edit / Preferences / New Document** oldalon a mintának megfelelően:

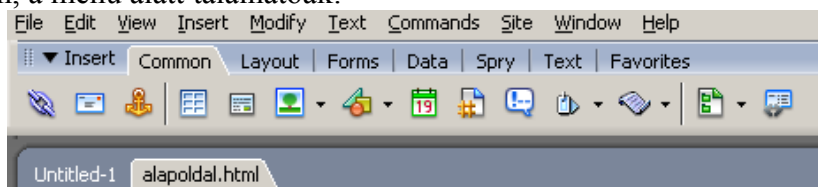


29. ábra Dreamweaver alapbeállítások

### 12.4.3. Használat

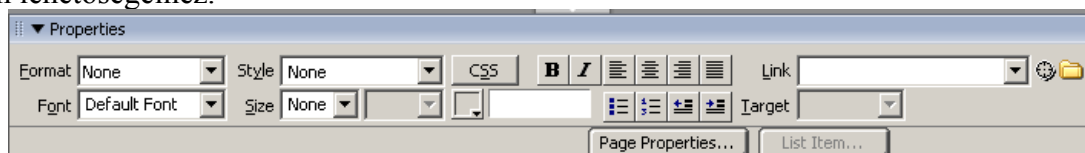
Ugyan nagyon sokat tud a program, de az alapfunkciókat gyorsan meg lehet találni, ha valaki tud szöveget szerkeszteni.

Az **eszköztárak** a CS3-as változatban külön fülecskéken keresztül érhetők el, amelyek az ablak tetején, a menü alatt találhatók:



30. ábra Dreamweaver eszköztárak

Az egyes elemek beállításai (**Properties**) pedig, a szerkesztő ablak alatti, külön részen található. Ezek a beállítások különböző elemek esetén különbözőek lehetnek, igazodva az elem lehetőségeihez.



31. ábra Dreamweaver properties

A Dreamweaver-nek is több **üzemmódja** van, amelyek között a szerkesztő ablak bal felső sarkában lehet váltani. A lehetséges módok a következők:

- Design - tervező nézet, az NVU Normál üzemmódja, ilyenkor nem látszik a kód, olyan, mintha egy szövegszerkesztővel dolgoznánk
- Code - csak a HTML kód látszik, azonban az NVU-tól eltérően itt is használhatók a különböző formázó elemek
- Split - egyszerre jelenik meg a tervező és a kódszerkesztő nézet, mindkettőn egyszerre lehet dolgozni



32. ábra Dreamweaver split nézet

A CS4-es változat alapértelmezett kinézete egy kicsit eltér a CS3-tól, de minden funkció ugyanúgy megtalálható benne. Ha nem tetszik az a kinézet, több lehetőség közül lehet választani az ablak tetején lévő Design legördülő menüből. Itt beállítható a CS-as felület is, a **Classic**-ot választva.

Ha valaki mélyebben szeretne elmélyedni a Dreamweaver lehetőségeiben, akkor rendelkezésére állnak magyar nyelvű könyvek, internetes ismertető, tanfolyamok és videók is. Például:

- Könyv: [Tanuljuk meg a Dreamweaver használatát 24 óra alatt](#)
- Ismertető oldal: <http://www.freeweb.hu/kesro/dream/>
- Tanfolyam: <http://www.weblapmester.com/weblap-keszites-tanfolyam.html>
- További hasznos linkek: <http://dreamweaver.lap.hu/>



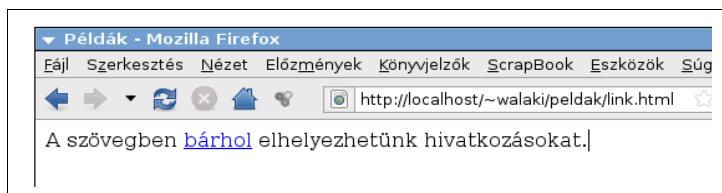
## 13. Hivatkozások

### 13.1. Hivatkozások másik oldalra

Hivatkozásokat az `<a>` TAG segítségével lehet létrehozni. A nyitó TAG-en belül lehet meghatározni a cél dokumentumot a **href** paraméterben. A nyitó és a záró TAG-ek között a hivatkozáshoz tartozó szöveget vagy képet lehet elhelyezni. A közrezárt részt a böngészőprogram a dokumentum többi részétől eltérően jeleníti meg (pl. aláhúzással, kerettel, ...), az egérkurzorral fölé érve, pedig a mutató alakja megváltozik. Rákattintva pedig a hivatkozott oldal nyílik meg.

A legtöbb esetben a hivatkozás egy másik fájlra, dokumentumra mutat. A hivatkozás kezdetét ekkor a `<a href="protokoll://elérési_út/fájlnév.kit">` utasítás jelzi, a hivatkozást a `</a>` utasításelem zárja le. Mind a protokoll, mind az elérési út elhagyható, amennyiben azonos URL-en van a kiindulási dokumentum és a hivatkozott.

```
A szövegben <a href="cimsorok.html">bárholt</a>
elhelyezhetünk hivatkozásokat.<br />
```



33. ábra Hivatkozás

Ehhez a példához arra van szükség, hogy a **link.html** fájl és a hivatkozott **cimsorok.html** fájl ugyanabban a könyvtárban legyenek elhelyezve!

### 13.2. Hivatkozás az aktuális oldalon belül

Ebben az esetben a hivatkozás az adott fájl egy távolabbi részére mozdtítja a böngészőablakot. A hivatkozás kezdetét a `<a href="#jelző">` utasításnak a dokumentumban való elhelyezése jelzi. A hivatkozást a `</a>` utasítás zárja le.

Az oldalon belül külön meg kell jelölni azt a részt, ahova majd a hivatkozás mutatni fog. Ezt a részt (praktikusan: könyvjelzőt), a `<a name="jelző">` és a `</a>` utasítások kell, hogy határolják. Az így definiált szövegrész a böngészőben nem lesz megkülönböztetve.

A "jelzo" tetszőleges név lehet, de mindkét helyen meg kell egyeznie!

```
A szövegben ilyenkor is <a href="#alcim">bárholt</a>
elhelyezhetünk hivatkozásokat.<br />
Itt sok-sok szöveget vagy képet el lehet helyezni,<br />
akár több sorban is!<br />
<h2><a name="alcim">Alcím</a></h2>
Ide fog ugrani, ha a "bárholt" szóra kattint!
```

### 13.3. Hivatkozás egy másik oldalon belülrre

Ez a megoldás az előző két példa kombinációja, vagyis másik oldalra hivatkozunk, de az oldalon belül tetszőleges pozícióra.



Ebben az esetben a hivatkozás egy másik fájl valamely pontosan meghatározott részére mutat. A hivatkozás kezdetét a

```
<a href="protokoll://elérési_út/fájlnév.kit#jelző">
```

utasítás jelzi, és a hivatkozást szintén a

```
</a>
```

elem zárja le.

A hivatkozott fájl kell, hogy tartalmazzon egy olyan részt (könyvjelzőt), ahová a hivatkozás mutat. Ezt a részt a `<a name="jelző">` és a `</a>` utasítások határolják.

### 13.4. Képes hivatkozás

Az eddigi példákban a hivatkozások nyitó és záró TAG-jai mindig szövegeket fogtak körbe. Azonban bármilyen másik látható elemet is hivatkozássá lehet alakítani, így a képeket is. Ekkor az `<a>` TAG-ek egy `<img>` TAG-et fognak körbe, pl.:

```
<h1 align="center">Egyik oldal</h1>
<p>Itt egy kép lesz maga a hivatkozás:</p>
<p align="center"> <a href="masik.html">  </a> </p>
<p align="center">Ha a képre kattint, a másik oldal
nyílik meg! </p>
```

### 13.5. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mik azok a hivatkozások?
2. Írd le a kódot, ami a **link** névvel a **http://index.hu** címre hivatkozik!
3. Adott egy **index.html** és a **filmek** könyvtárban a **filmek.html** fájl. Az **index.html** fájlban található a következő kódrészlet:  

```
/<p align="centter">A filmekről <a
src=filmek.html>itt</src> olvashatsz!<p>/
```
4. Keresd meg a hibát és javítsd ki!
5. Hogyan lehet ugyanazon oldalon belül adott pozícióba ugrani link segítségével?
6. Mit csinál a következő kódrészlet?  

```
/<a name="Bumm">Robbanás</a> /
```
7. Egy oldalon hány darab hivatkozás lehet?

A / jelek csak a HTML kódok elejét és végét jelzik!

## 14. Felsorolás, számozás

### 14.1. Számozott lista

A számozott lista elemeit az `<ol>` `</ol>` TAG-ek veszik körbe. Az egyes elemek, pedig `<li>` `</li>` TAG-ek határolják. A számozás automatikus és 1-től indul, de a kezdőelem értékét meg lehet adni a nyitó TAG **start** paraméterében. Például a következő példában az „Első elem” 10. sorszámot kap, a következő 11-et, és így tovább:

```
<ol start="10">
<li>Első elem</li>
<li>Második elem</li>
<li>Harmadik elem</li>
</ol>
```

A számozás itt is lehet szám (ez az alapértelmezés), de a **type** paraméterrel meghatározhatunk kisbetűs (*a*), nagybetűs (*A*), kisbetűs római (*i*), és nagy betűs római (*I*) számozási formát is. Például az `<ol type="a">` a kisbetűs formát állítja be.

## 14.2. Számozatlan lista

Ezek az elemek ugyanazt a hatást érik el, amit a szövegszerkesztőkben lévő felsorolások. Tetszőleges számú elem szerepelhet egy listában, amelynek minden eleme előtt egy speciális jel jelenik meg.

A számozatlan listát az `<ul>` `</ul>` TAG-ek határolják. Az egyes elemeket pedig a `<li>` `</li>` TAG-ek közé kell elhelyezni. Például:

```
Ez itt egy oldal első sora, ami után egy lista
következik:
<ul>
<li>Első elem</li>
<li>Második elem</li>
<li>Harmadik elem</li>
</ul>
```

### 14.2.1. Definíciós lista

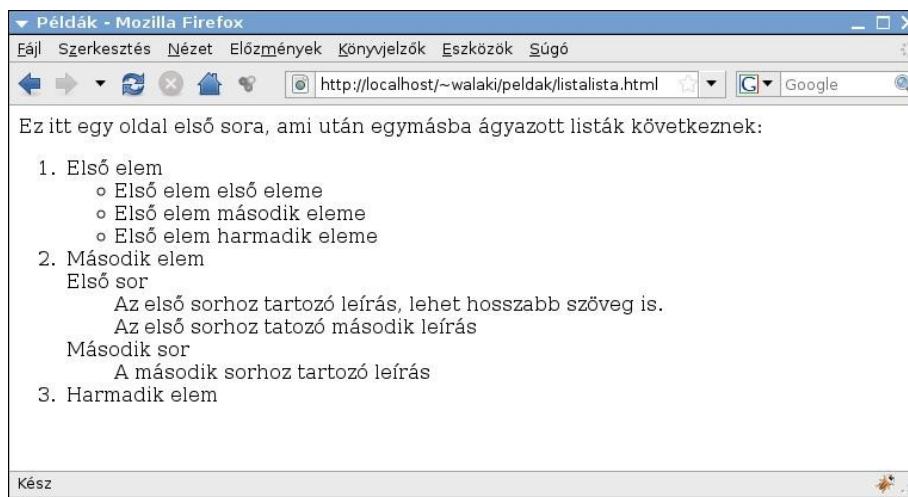
A leíró listát (angolul definitions lists) a `<dl>` `</dl>` TAG-ek határolják. A listák elemeit a `<dt>` jelzi, míg az elemhez tartozó leírásokat a `<dd>`. Egy lista elemhez több leírás is tartozhat. Például:

```
Ez itt egy oldal első sora, ami után egy lista
következik:
<dl>
  <dt>Első sor
    <dd>Az első sorhoz tartozó leírás, lehet hosszabb
szöveg is.
    A leírás tördelése automatikus. Szépen igazodnak a
betördelt sorok
    az első sor kezdőpontjához.</dd>
    <dd>Az első sorhoz tartozó második leírás</dd>
  </dt>
  <dt>Második sor
    <dd>A második sorhoz tartozó leírás</dd>
  </dt>
</dl>
```

### 14.3. Többszörös és vegyes listák

A különböző listák egymásba is ágyazhatók. Például számozott listában lehet számozatlan és viszont. Az alábbi példa bemutat néhány variációt:

```
Ez itt egy oldal első sora, ami után egymásba ágyazott listák következnek:
<ol>
  <li>Első elem</li>
  <ul>
    <li>Első elem első eleme</li>
    <li>Első elem második eleme</li>
    <li>Első elem harmadik eleme</li>
  </ul>
  <li>Második elem</li>
  <dl>
    <dt>Első sor
      <dd>Az első sorhoz tartozó leírás, lehet hosszabb
szöveg is.</dd>
      <dd>Az első sorhoz tartozó második leírás</dd>
    </dt>
    <dt>Második sor
      <dd>A második sorhoz tartozó leírás</dd>
    </dt>
  </dl>
  <li>Harmadik elem</li>
</ol>
```



34. ábra Egymásba ágyazott vegyes listák

### 14.4. Ellenőrző kérdések

A lecke tanulmányozása után próbáld meg önállóan válaszolni a következő kérdésekre!

1. A HTML kódban milyen TAG-ek segítségével lehet felsorolásokat készíteni?
2. Hogyan definiálható egy számozott lista? Írd le a kódot!

## 3. Javítsd ki a hibát!

```
/<b>Processzorok jellemzői:<b>
<ol>órajel</ol><ul>tokozás</ul><ul>cache mérete</ul></ol>/
```

- Írj egy példát 2 elemű számozatlan felsorolásra, ahol az elemek előtt üres körök jelennek meg!
- Mit jelent a leíró lista?
- Hogyan készíthetünk leíró listát?
- Hány elemű lehet egy számozott felsorolás?
- Szerepelhet-e egy definíciós listában több számozott lista?
- Mi a különbség a számozott és a számozatlan lista között?

A / jelek csak a HTML kódok elejét és végét jelzik!

## 15. Teszt

Komplex teszt végrehajtása a Moodle e-learning keretrendszerben, automatikus értékeléssel.

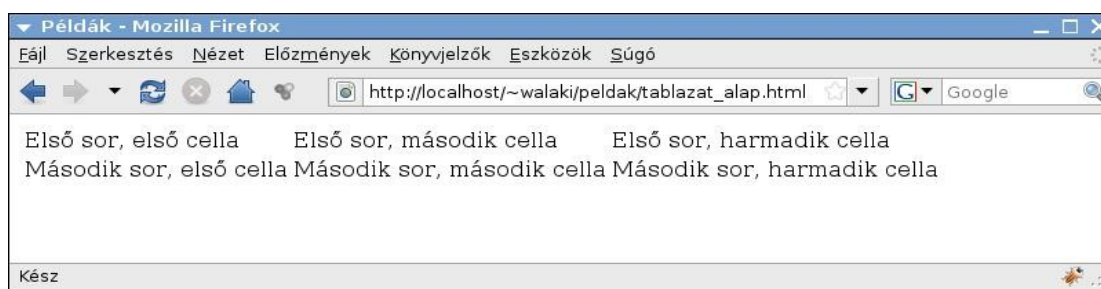
## 16. Táblázatok

### 16.1. Táblázatok alapjai

Szövegszerkesztőkben létrehozható táblázatok itt is készíthetünk. Egy táblázatot a `<table>` utasítással kezdünk és a `</table>`-vel zárjuk. A táblázat sorait a `<tr>` utasítással kezdjük, a soron belüli elemeket (cellákat) pedig a `<td>` TAG-el.

A táblázat celláiban tetszőleges szöveg vagy kép lehet. Egy 3\*2-es alaptáblázat például a következőképpen definiálható:

```
<table>
  <tr>
    <td>Első sor, első cella</td>
    <td>Első sor, második cella</td>
    <td>Első sor, harmadik cella</td>
  </tr>
  <tr>
    <td>Második sor, első cella</td>
    <td>Második sor, második cella</td>
    <td>Második sor, harmadik cella</td>
  </tr>
</table>
```



35. ábra Egyszerű táblázat

## 16.2. A táblázat címe

A táblázatnak a címét a `<caption>` és a `</caption>` utasítások között kell megadni. Az így megadott cím nem a táblázatban, hanem előtte fog megjelenni! A cím az **align** paraméter segítségével balra (*left*), jobbra (*right*), felülre (*top*), alulra (*bottom*) igazítható. Egy példa alulra igazított táblázat címmel:

```
<table>
<caption align="bottom">A táblázat címe</caption>
<tr>
. . .
</tr>
</table>
```

## 16.3. Cellák összevonása

A szövegszerkesztőkben és a táblázat kezelő programokban lehetőség van arra, hogy egyes cellákat összevonjunk. Erre a HTML is biztosít lehetőséget a `<td>` TAG **colspan** és **rowspan** paramétereivel. Az így formázott táblázatokat azonban nehéz áttekinteni, így kezelni is.

A HTML-ben a táblázatokat soronként (`<tr>` elem) és azon belül cellánként (`<td>` elem) írjuk le. Minden sorban azonos számú cellának, vagyis azonos számú `<td>` elemnek kell lennie, különben a táblázat szétesik. A **colspan** paraméternek egy számot adhatunk értékül, ami azt jelzi, hogy az adott `<td>` elem, hány cellát határoz meg. A `<td colspan="2">...</td>` kód kettő cellát kapcsol össze egy cellává. Az érték nagyobb is lehet, kisebb értéknek meg nincs értelme. Amire figyelni kell, hogy az összevont cellák elemei (a `colspan="2"`-vel meghatározott cella két cellát ér!) mindig külön kerüljenek számolásra a soron belül, vagyis a soronkénti azonos cellaszám megmaradjon.

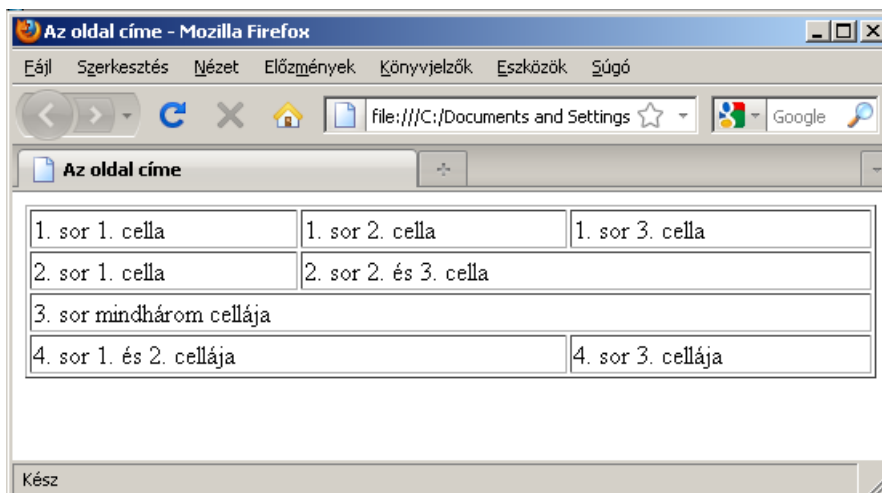
A következő példa egy 3x3-as táblázatban mutatja be a különböző celleösszevonási lehetőségeket. Az első sorban mindhárom cella megmaradt külön, hogy ehhez lehessen viszonyítani a többi sor celláinak elhelyezkedését. A második sorban az első cella maradt külön, de a másik kettő összevonásra került. A harmadik sorban mindhárom cella össze lett vonva. A harmadik sorban, pedig az első két cella került összevonásra, míg a harmadik maradt külön:

```
<table width="100%" border="1">
<tr>
<td>1. sor 1. cella</td>
<td>1. sor 2. cella</td>
<td>1. sor 3. cella</td>
</tr>
<tr>
<td>2. sor 1. cella</td>
<td colspan="2">2. sor 2. és 3. cella</td>
</tr>
<tr>
<td colspan="3">3. sor mindhárom cellája</td>
</tr>
<tr>
<td colspan="2">4. sor 1. és 2. cellája</td>
```

```

    <td>4. sor 3. cellája</td>
  </tr>
</table>

```



36. ábra Táblázat összevont cellákkal

A példában bekapcsolásra került a keret (**border** paraméter) is, a cellák határainak láthatóvá tétele miatt.

### 16.3.1. Cellák függőleges összevonása

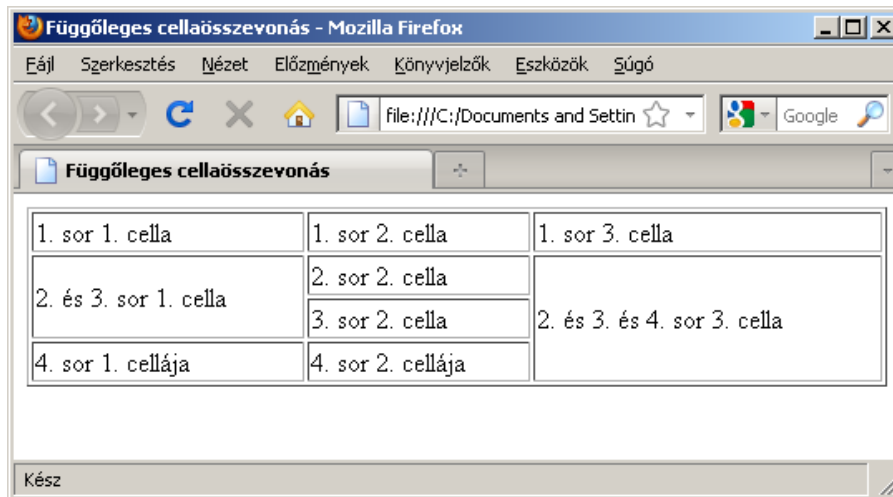
A függőleges összevonáshoz a **rowspan** paramétert kell használni. Azonban itt is figyelembe kell venni, hogy az összevont cellákat minden sorban figyelembe kell venni. Ha két cellát függőlegesen vonunk össze, akkor a cellával mindkét sorban számolni kell.

A függőleges cellaösszevonást mindig csak a legelső sorban kell jelezni, a többi sorban figyelmen kívül kell hagyni. A következő példában 4x3-as táblázat első sora 3 cellát tartalmaz viszonyítási alapnak. A 2. és 3. sor 1. cellái össze lettek vonva. Ugyanígy összevonásra kerültek a 2., 3., és 4. sor 3. cellái. A kód és az eredmény:

```

<table width="100%" border="1">
  <tr>
    <td>1. sor 1. cella</td>
    <td>1. sor 2. cella</td>
    <td>1. sor 3. cella</td>
  </tr>
  <tr>
    <td rowspan="2">2. és 3. sor 1. cella</td>
    <td>2. sor 2. cella</td>
    <td rowspan="3">2. és 3. és 4. sor 3. cella</td>
  </tr>
  <tr>
    <td>3. sor 2. cella</td>
  </tr>
  <tr>
    <td>4. sor 1. cellája</td>
    <td>4. sor 2. cellája</td>
  </tr>
</table>

```

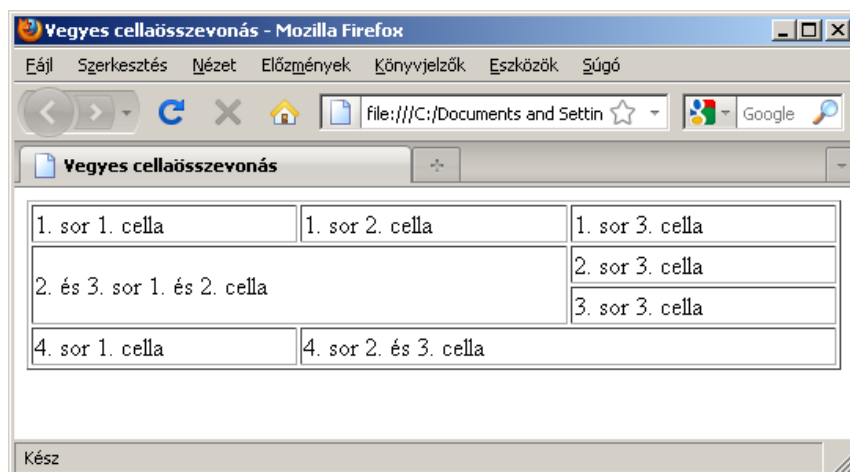


37. ábra Táblázat függőleges összevonással

### 16.3.2. Vegyes összevonás

A vízszintes és függőleges összevonások egyszerre is alkalmazhatók, így igen összetett táblázatokat lehet létrehozni. A következő példában a 2. és 3. sor 1. és 2. cellája egyszerre került összevonásra:

```
<table width="100%" border="1">
  <tr>
    <td>1. sor 1. cella</td>
    <td>1. sor 2. cella</td>
    <td>1. sor 3. cella</td>
  </tr>
  <tr>
    <td colspan="2" rowspan="2">2. és 3. sor 1. és 2.
cella</td>
    <td>2. sor 3. cella</td>
  </tr>
  <tr>
    <td>3. sor 3. cella</td>
  </tr>
  <tr>
    <td>4. sor 1. cella</td>
    <td colspan="2">4. sor 2. és 3. cella</td>
  </tr>
</table>
```



38. ábra Táblázat vegyes összevonással

## 16.4. Táblázatok alkalmazása

A táblázatok elsősorban különböző adatok rendezett megjelenítésére szolgálnak. A fő alkalmazási területe mindenképpen ez, vagyis rendezett adatmegjelenítés.

A táblázatokat azonban könnyen fel lehet használni a weboldalak szerkezetének kialakítására is, a táblázat keretei (border) nélkül ugyanis nem látszik, hogy van egyáltalán táblázat, viszont pontosan meg lehet határozni, hogy egy cella milyen széles, illetve maga a táblázat milyen szélességű legyen, és még igazítani is lehet.

## 16.5. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mit értünk táblázat alatt?
2. Milyen részei vannak egy táblázatnak?
3. Tanulmányaid kapcsán hol találkoztál eddig táblázattal?
4. Sorold fel a táblázatokkal kapcsolatos TAG-eket!
5. Mire szolgál a `<td>` TAG?
6. Helyes-e a következő kódrészlet? Indokold!  

```
/ <tr><td>alma<td>körte<td>dinnye</tr> /
```
7. Hogyan lehet olyan kétsoros táblázatot készíteni, ahol az első sor csak egy cellából áll, míg a második három cellából? Írd le a teljes kódot!
8. Lehet-e képet elhelyezni egy táblázatban?
9. Mire használatos a `<caption>` TAG?

A / jelek csak a HTML kódok elejét és végét jelzik!

## 17. Táblázatok formázása

### 17.1. Teljes táblázatra vonatkozó formázási lehetőségek

A táblázat nyitóutasítása tartalmazhat a teljes táblázatra vonatkozó beállításokat. Az utasítás teljes formája:

```
<table border="szám" align="hely"
```



```
width="táblázatszélessége" nowrap cellpadding="pszám"
cellspacing="kszám" bgcolor="színkód">
```

Ahol a **border** opció a rácsozat szélességét határozza meg. Nulla érték esetén nincs rácsozat. Az **align** a teljes tábla elhelyezkedését határozza meg ( *left*, *right*, *center* lehet). A **width** a teljes tábla szélességét határozza meg, értékként % is megadható. A **cellspacing** a cellák közötti üres terület szélességét határozza meg, a **cellpadding** pedig a cellákon belüli szélek nagyságát. A **nowrap** opció a cellák szövegének tördelését tiltja le. Végül a **bgcolor** a táblázat háttérszínét határozza meg.

Például a következő kódrészlet egy 2 pixeles kerettel rendelkező, az oldal szélességéhez viszonyítva 60% széles, középre igazított, sárga háttérszínű táblázatot határoz meg, amiben a cellák közötti távolság 5 pixel, míg a cellákon belüli margók 2 pixelesek:

```
<table border="2" width="60%" align="center"
cellspacing="5" cellpadding="2" bgcolor="yellow">
. . .
</table>
```

## 17.2. Táblázatok sorainak formázása

A táblázat sorai külön is formázhatók. A teljes tr TAG a következő:

```
<tr align="vízszintesigazítás" bgcolor="háttérszín"
valign="függőlegesigazítás">
```

Az **align** értéke lehet *left*, *right*, *center*, vagyis az adott sor igazítható balra, jobbra és középre. A **valign** paraméterrel pedig a sorban lévő szöveg függőleges elhelyezését lehet megadni *top*, *bottom*, *middle* értékekkel, amik fent, lent és középre igazítást állítanak be. A **bgcolor** soronként is használható, így minden sornak külön meg lehet adni a háttérszínét.

```
<tr align="right" bgcolor="red">
  <td>A sorban minden cella tartalma jobbra igazított és
piros háttérszínű lesz</td>
  <td>Ez a cella is </td>
</tr>
```

## 17.3. Táblázat celláinak formázása

A táblázat minden egyes cellája külön is formázható. A **<td>** TAG teljes alakja a következő:

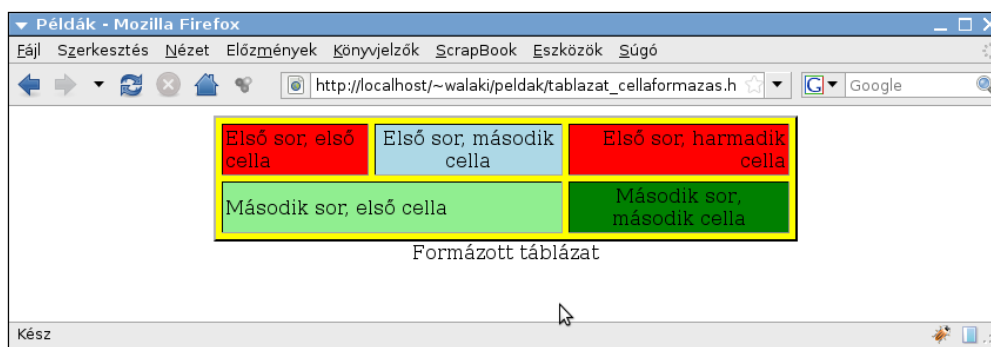
```
<td align="vízszinteshely" valign="függőlegeshely"
bgcolor="háttérszín" width="szélesség" height="magasság"
nowrap="true" rowspan="szám" colspan="szám">
```

Az **align** paraméterrel a vízszintes igazítás állítható be balra (*left*), jobbra (*right*), középre (*center*), sorkizártan (*justify*). A **valign** paraméterrel pedig a függőleges igazítás állítható fentre (*top*), alulra (*bottom*) és középre (*middle*). A **bgcolor** a cella háttérszínét határozza meg. A **nowrap** paraméterrel letiltható az automatikus sortörés a cellán belül, ha megadjuk.

A **width** paraméterrel a cella szélessége állítható be, akár %-os módon is. A **height** paraméterrel pedig a cella magasságát lehet meghatározni. Itt is megadható %-os érték is.

A **colspan** segítségével vízszintesen összevonhatunk cellákat. A paraméter értéke határozza meg, hogy hány darab cella kerül összevonásra. A **rowspan** ugyanezt teszi, csak a sorokkal, vagyis ezzel sorokat lehet összevonni.

```
<table border="2" width="60%" align="center"
cellspacing="5" cellpadding="2" bgcolor="yellow">
  <caption align="bottom">Formázott táblázat</caption>
  <tr align="right" bgcolor="red">
    <td align="left">Első sor, első cella</td>
    <td align="center" bgcolor="lightblue">Első sor,
második cella</td>
    <td>Első sor, harmadik cella</td>
  </tr>
  <tr align="center" bgcolor="green">
    <td align="left" bgcolor="lightgreen"
colspan="2">Második sor, első cella</td>
    <td>Második sor, második cella</td>
  </tr>
</table>
```



39. ábra Cellaformázott táblázat

## 17.4. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Sorold fel a táblázatokkal kapcsolatos TAG-eket!
2. Hogyan kell keretes táblázatot készíteni?
3. Hol és hogyan lehet állítani a háttérszínt?
4. Mit állít be a `cellspacing="12"` paraméter?
5. Miket lehet beállítani egy több cellából álló sor esetén?
6. Hogyan lehet a táblázat celláiban a szöveget igazítani?
7. Lehet-e egy táblázat cellájában egy újabb táblázat?

## 18. Speciális karakterek

### 18.1. Alapok

A HTML-ben a <, > és & jelek speciálisak. Ezek adják meg HTML tagek elejét és végét, így a dokumentumon belül nem a *kisebb*, *nagyobb* és az *és* jeleket jelentik.

Az egyik legegyszerűbb hiba, amit egy webfejlesztő véthet, hogy az és jelet felhasználja a dokumentumban, így valami váratlan történik. Ha például azt írja, hogy „az angolszász jelölésben a tömeg stones&pounds”, akkor egyes böngészőkben ez úgy végződik, hogy „...stones&”.  
 Ez azért van, mert a „&pounds;” szövegrész valójában egy HTML karakter referencia. A karakter referencia egy módszer az olyan karakterek beillesztésére a dokumentumokba, amelyeket egyébként csak nehezen vagy sehogyan sem írhatunk be a billentyűzetet használva, vagy a dokumentum kódkészletében nem szerepel.

Az és jel (&) vezet be egy ilyen referenciát, és a pontosvessző (;) zárja le. Ennek ellenére sok kliens eszköz elég megbocsátó a HTML hibák iránt, és nem veszi figyelembe, hogy nincs lezáró pontosvessző, így a „&pound” kifejezést is karakter referenciaként értelmezi.

A referenciák lehetnek számok (numerikus referenciák), vagy rövid szavak (egyedi referenciák) is. Vannak olyan karakterek, amelyekre mindkét módon lehet hivatkozni, és vannak olyanok is, amelyekre csak numerikusan.

Ha egy és jelet akarunk beírni a dokumentumba, akkor az „&amp;” karakter referenciát, vagy a „&#38;” numerikus referenciát kell használnunk. A karakter referenciák listája többek között az [evolt.org](http://evolt.org) oldalán is megtalálható.

**Kódokat használva, a fájl karakterkódolásától és kódlap beállításától függetlenül, mindig a kívánt karakterkép fog megjelenni!**

### 18.2. Ékezetes betűk kódjai

Amennyiben ékezetes karaktereket szeretnénk megjeleníteni az oldalon, akkor azt HTML kódokkal is megtehetjük, néhány kompromisszummal (az ő helyett hullámos ő, az ű helyett kalapos ű karakter jelenik meg. A karaktereket meghatározhatjuk számokkal (NumKód) és nevekkkel (Kód) is. Összefoglalva az ékezetes karakterek leírási módjait:

Betű	NumKód	Kód	Betű	NumKód	Kód
á	&#225;	&aacute;	Á	&#193;	&Aacute;
é	&#233;	&eacute;	É	&#201;	&Eacute;
í	&#237;	&iacute;	Í	&#205;	&Iacute;
ó	&#243;	&oacute;	Ó	&#211;	&Oacute;
ú	&#250;	&uacute;	Ú	&#218;	&Uacute;
ö	&#246;	&ouml;	Ö	&#214;	&Ouml;
ü	&#252;	&uuml;	Ü	&#220;	&Uuml;
ő -> ò	&#245;	&otilde;	Ő -> Ò	&#213;	&Otilde;
ô -> ô	&#244;	&ocirc;	Ô -> Ô	&#212;	&Ocirc;
ű -> û	&#251;	&ucirc;	Ű -> Û	&#219;	&Ucirc;

ű -> ũ	&#361;	-	Ű -> Ũ	&#360;	-
ő	&#337;	-	Ő	&#336;	-
ú	&#369;	-	Ú	&#368;	-

Ez a fajta leírásmód a kezdeti időkre volt jellemző, ma már nem ajánlott. Helyette a megfelelő kódlap használata, és a kódban történő (`<meta>` TAG) helyes megadása (ISO-8859-2 vagy UTF-8) használatos.

A teljesség kedvéért meg kell említeni, hogy a "kalapos" ő és ú betűk amiatt vannak, mert ezek a kódok az ISO-8859-1 kódtábla elemeit azonosítják. Ez a kódtábla egyáltalán nem tartalmaz "rendes" ő és ú betűképet. A fenti táblázatban a világos kék háttérű elemek már másik kódtábla részei. Nem sokan tudják, hogy kóddal is lehet "rendes" ő és ú betűt írni.

Numerikus kóddal nemcsak az ékezetes betűkre lehet hivatkozni, hanem a teljes ABC elmeire is. A teljes lista megtalálható a szabványban is, de más oldalakon is:

- <http://www.w3.org/TR/1999/REC-html401-19991224/sgml/entities.html>
- <http://franka-egom.ofm.hu/segedanyagok/netoktato/chars.htm>
- [http://webdesign.about.com/library/bl\\_htmlcodes.htm](http://webdesign.about.com/library/bl_htmlcodes.htm)

### 18.3. Fontosabb szimbólumok

A [HTML 4.01 szabvány](#) 3 részre bontja a speciális karaktereket:

- ISO-8859-1 karakterek
- szimbólumok, matematikai jelek és görög ABC betűi (ISO 8879 és az ISO 10646 alapján)
- fontosabb jelek és nemzeti karakterek (ISO 8879 és az ISO 10646 alapján, amik a CP-1252 128-159 azonosítók közé esnek)

A szabvány oldalán nem látszanak a megjelenő karakterképek, ott csak a kódok találhatók. Ezért itt kigyűjtésre kerültek a legfontosabb szimbólumok a karakterképekkel együtt:

Jel	NumKód	Kód	Jel	NumKód	Kód
&	&#38;	&amp;	Π	&#8719;	&prod;
<	&#60;	&lt;	Σ	&#8721;	&sum;
>	&#62;	&gt;	≈	&#8776;	&asymp;
"	&#34;	&quot;	½	&#189;	&frac12;
©	&#169;	&copy;	¼	&#188;	&frac14;
®	&#174;	&reg;	♥	&#9829;	&hearts;
™	&#153;	&trade;	∞	&#8734;	&infin;
§	&#167;	&sect;	⇐	&#8656;	&iArr;
€	&#8364;	&euro;	⇒	&#8658;	&rArr;
±	&#177;	&plusmn;	⇔	&#8660;	&hArr;
¶	&#182;	&para;	μ	&#181;	&micro;

## 18.4. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mik azok a speciális karakterek?
2. Hogyan írhatunk le egy HTML kódon belül speciális karaktereket?
3. Hogyan lehet HTML kódban megbízhatóan leírni azt, hogy / 100€ /?
4. Írd le HTML kóddal a következőt:  
$$/ 5 * \beta \geq \mu * \varepsilon, \beta \in \mathbb{R} /$$
5. Milyen karaktert jelöl a / &#937; / kód?
6. Milyen módokon és feltételekkel lehet HTML kódban "ő" (hosszú ő) betűt írni?
7. Mi jelenik meg a böngészőben a következő HTML kód esetén:  
`/ &amp;otil&#100;e&#59;:&#337; /`
8. A HTML szabványban melyik kódtáblák használatosak speciális karakterek esetén?

A kérdésekben a / jelek csak a kódok kezdetét és végét jelölik!

## 19. Multimédia

### 19.1. Multimédia alapok

**Média** a médium szó többes száma, vagyis több médiumot jelent. Ebből a szóból képzett multimédia sok-sok médiummal egyenlő. A média a mondanivaló kifejezésére használatos közvetítő közegek összessége. Ilyen közeg lehet pl.:

- szöveg (nyomtatott, internetes)
- kép (állókép, fénykép)
- hang (rádió)
- mozgókép (tv, videó)

A mai szóhasználatot tekintve a **multimédia** már erősen kötődik az informatikához. Ebben a környezetben olyan számítógépes környezetet jelent, ahol az információ közléséhez felhasználnak szöveget, álló- és mozgóképet, valamint hangot is, továbbá a megjelenítésükhöz szükséges berendezést is (multimédiás számítógép hardver és szoftver elemei).

#### 19.1.1. Multimédia a web oldalakon

Web oldalak is tartalmazhatnak ma már mindenféle multimédiás elemeket. Ilyen elemek lehetnek például:

- az oldalon elhelyezett szöveges tartalmak
- képek
- mozgóképek (animált gif)
- hangok (zenei állományok, pl.: wav, mp3)
- videók (avi, mpg, stb állományokat)
- dinamikus HTML elemek (képek, szövegek, látványelemek mozgatása JavaScript segítségével)
- Flash alapú tartalmak
- Java appletek

Az **animált gif**, olyan .gif kiterjesztésű kép állomány, amelyben a kép maga változik. Ezeket a kezdeti időkben használták előszeretettel az oldalak érdekesebbé tételéhez. Ma már nem jellemző a használata, mivel minden modern számítógép képes videók lejátszára is.

A DHTML (**dinamikus HTML**) egy lehetőség arra, hogy a JavaScript nyelv segítségével bármely, az oldalon lévő elemet vagy jellemzőjét módosíthassuk az oldal megjelenése után is, akár különböző eseményekhez (egér mozgás, billentyű leütés) kapcsolva.

Az **Adobe Flash** egy szoftver amely az Adobe Systems termékcsaládba tartozik. Egy olyan professzionális multimédiafejlesztő-alkalmazás, amelynek segítségével könnyedén fejleszthetünk webes alkalmazásokat, játékokat, muzikát, vagy akár mobil tartalmakat. A Flash hatékonyan ötvözi a vektoros rajzolóprogram és a professzionális animációszerző-program minden előnyét.

A **Java** nyelven megírható programokat kétfajta elnevezés szerint csoportosíthatjuk. Az egyik az alkalmazások, azaz az önálló Java programok csoportja. A másik csoportot az **appletek** (programcskák) képezik, amelyeknek fő tulajdonságuk, hogy honlapokba, azaz HTML oldalakba lehet őket beszerezni. Az önálló programok végrehajtását a Java értelmezők, az appletek végrehajtását pedig az appleteket futtatni képes böngészőprogramok intézik.

### 19.1.2. MIME types - multimédiás elemek típusdefiníciója

A **MIME** (Multipurpose Internet Mail Extensions) internetes szabvány, ami eredetileg az SMTP-vel továbbított e-mailek formátumának jelzésére szolgált, de később átvette a HTTP és a SIP is. Egy üzenet (a HTML oldal is) a sima szöveg mellett számos egyéb dolgot is tartalmazhat: nem ASCII karakterkészletű kódolt szöveget, HTML kódot, képet, hangot, videót stb. A **Content-type** fejléc jelzi, hogy a fogadónak a kapott bájt sorozatot ezek melyikeként kell értelmeznie.

A fejléc egy típus/altípus alakú jelölést és szöveges típusok esetén opcionálisan egy karakterkódolást tartalmazhat. A típus jelzi a tartalom jellegét, ez általában a következők egyike: text (szöveg – ember által is olvasható szöveges formátum), image (kép), audio (hang), video vagy application (alkalmazás – általában egy specifikus szoftver által használt formátum). Lehetőség van több különböző típusú részből álló üzenet küldésére is a multipart (többrészes) tartalomtípussal.

A MIME szabvány nyitott, bárki regisztrálhat új tartalomtípust. A csak egy bizonyos cég által használt formátumok típusai általában application/vnd.gyártó-program alakúak (a vnd a vendor (forgalmazó) szóból van). Ha az altípus neve elején x- van, az azt jelenti, hogy nem szabványos típus – az ilyen típust nem lehet regisztrálni az IANA-nál.

Néhány gyakran használt tartalomtípus:

Típus/Altípus	Leírás
text/plain	ASCII szöveg (ez az alapértelmezett tartalomtípus)
text/html	HTML fájl
text/CSS	CSS fájl
image/gif	GIF kép
image/jpeg	JPEG kép
audio/mp3	MPEG Layer 3 zene
audio/x-wav	WAVE hang
video/mpeg	MPEG film
application/octet-stream	közelebről meg nem határozott jelentésű bájt sorozat
application/javascript	Javascript fájl

application/vnd.ms-excel	Microsoft Excel dokumentum
application/x-shockwave-flash	Adobe Flash fájl
multipart/mixed	többrészes üzenet
multipart/alternative	egy üzenet több különböző formátumban

HTML kód esetén is lehetőség van arra, hogy az oldalba különböző típusú tartalmat illesszünk be. A MIME type ennek a tartalomnak a típusát határozza meg. A böngésző, vagyis a megjelenítő rendszer, a típus alapján el tudja dönteni, hogy az adott állományt milyen módon kell kezelni, hogyan kell megjeleníteni, stb.

A részletes lista több helyen is elérhető, pl.:

- [http://www.w3schools.com/media/media\\_mimeref.asp](http://www.w3schools.com/media/media_mimeref.asp)
- <http://www.iana.org/assignments/media-types/>
- <http://www.tanit.hu/mimetypes>
- <http://www.webmaster-toolkit.com/mime-types.shtml>

### 19.1.3. Kódekek

A kódek olyan szoftver, amely digitális médiatartalmak (például zeneszám és videó) tömörítésére és kibontására szolgál. A Windows Media Player és más programok digitális médiatartalmak lejátszására és létrehozására használják a kódekeket.

Manapság több száz audió és videó kódek van használatban. Néhányat a Microsoft vállalat hozott létre, azonban a kódekek nagy része más vállalatok, szervezetek vagy személyek által létrehozott program. Alapértelmezés szerint a Windows operációs rendszer és a Media Player alkalmazás a legnépszerűbb kódekeket (például a Windows Media Audio, a Windows Media Video és az MP3 kódeket) tartalmazza.

Előfordulhat azonban, hogy a lejátszani kívánt tartalmat olyan kódekkel tömörítették, amelyet a Windows vagy a Media Player alkalmazás alapértelmezés szerint nem tartalmaz (például egy DivX videó kódekkel vagy Ogg Vorbis audió kódekkel tömörített fájl). Mivel a Player alkalmazás bővíthető, sok esetben a megoldás a szükséges kódek ingyenes vagy fizetős letöltése az internetről. Néhány esetben a Media Player alkalmazás automatikusan használja a más digitális médialejátszó és létrehozó program által a számítógépre telepített kódekeket.

A kódek nem keverendő össze a videófájl-formátummal, ami a kódek által kódolt hang/kép információ tárolására szolgál. A leggyakoribb hang/kép fájlformátumok (például .ogg, .mpg, .avi, .mov) egy, de akár több különböző kódekkel kódolt információt is tárolhatnak. (Például az AVI kiterjesztésű film lehet DivX, de akár XviD is.)

A különböző kódekeket egybe is szokták csomagolni, így nem kell egyesével telepíteni őket. Ezeket a szoftvercsomagokat nevezik **codeck pack**-nak. Több ilyen csomag szabadon elérhető az interneten, pl.:

- [ACE Mega CoDecS Pack](#)
- [K-Lite Codec Pack Full 5.6.9](#)
- [XP Codec Pack](#)

**A kódekeket azon a gépen kell telepíteni, ahol le szeretnénk játszani valamilyen multimédiás tartalmat! Ezt a felhasználónak kell telepítenie, az operációs rendszer ugyanis nem tartalmaz minden kódeket!**

#### 19.1.4. Bővítmények (plugin-ek)

A képeken kívül minden más multimédiás elem böngészőn belüli megjelenítéséhez, kiegészítők szükségesek a felhasználó számítógépén. A kiegészítők a böngésző programok megjelenítési képességeit egészítik ki. Ezeket lehet külön programként is telepíteni, mint például a QuickTime, ami a lejátszóval együtt telepíti a böngészőkhöz a bővítményt is. Másik esetben a böngészők felismerik, hogy olyan elemet kellene megjeleníteni az oldalon, amihez újabb kiegészítőre van szükség, és azt jelzik is, akár rögtön a letöltést is biztosítják.

A bővítményeket külön ki-be lehet kapcsolni például a FireFox-ban:

### 19.2. Multimedia szabványosan, az `<object>...</object>` elem

Multimédiás tartalom megjelenítésére a HTML 4 szabvány bevezette az `<object>` TAG-et, ami elvileg minden multimédiás elem megjelenítésére alkalmas.

Az `<object>` TAG általános alakja igen összetett, mivel sok mindent meg lehet vele jeleníteni. Külön van nyitó és záró TAG-je is, közöttük elvileg tetszőleges elem lehet, de elsősorban arra szolgál, hogy ha az adott elem nem jeleníthető meg valamilyen okból, akkor más kódot is végre lehessen hajtani.

```
<object> ... </object>
```

- Az `<object>` fontosabb attribútumai:
- **data** - az objektum neve, pl.: alma.jpg
- **type** - az objektum MIME típusa, pl.: image/jpeg
- **classid** - az objektumot egy URI határozza meg
- **codebase** - az URI alapútvonalát definiálja
- **codetype** - classid-vel együtt használatos, annak típusát határozza meg
- **standby** - az objektum betöltődése alatt megjelenő üzenet
- **width** - az objektum szélességét állítja
- **height** - az objektum magasságát állítja
- **align** - az objektum elrendezését állítja: left, right, top, bottom, middle
- **border** - az objektum keretét állítja

A `<param>` TAG segítségével pedig az adott objektumnak lehet további paramétereket beállítani. A `<param>`-ot az `<object>` TAG-eken belül kell használni, és egyszerre több is lehet!

```
<object ...>
<param ...>
<param ...>
Az a szöveg, ami akkor jelenik meg, ha a kívánt
objektumot valamiért nem lehet megjeleníteni!
</object>
```

A teljes paraméterlistát a HTML 4.01 szabványból lehet megismerni:

<http://www.w3.org/TR/1999/REC-html401-19991224/struct/objects.html#edef-OBJECT>



### 19.2.1. Kép

A HTML 4.01 szabványban Képek megjelenítésére továbbra is használható az `<img>`, de már rendelkezésünkre áll az `<object>` elem is.

Az `<object>` TAG esetén mindig meg kell határozni a kép fájl MIME típusát is, nem elég az állomány nevében a kiterjesztés. Az `<img>` elemen belüli **alt** attribútum helyett az `<object>` TAG-ek közötti szöveg használatos. A **height**, **width**, **align** paraméterek ugyanúgy működnek, mind a két esetben.

```
<p>Szöveg közé <b>img</b> TAG-al  beillesztett
kép.</p>
<p>Szöveg közé <b>object</b> TAG-al
<object data="viragkicsi.jpg" type="image/jpeg">
Virágocska
</object>
beillesztett kép.</p>
```

Ugyanazt a kódot IE8 alatt megjelenítve, az `<object>`-el definiált kép nem jelenik meg, vagyis **az IE8 nem támogatja a jpg képek object TAG-el történő megjelenését!**

#### 19.2.1.1. Függőleges igazítás

A képek kezelése ugyanaz maradt a függőleges igazításnál, vagyis az **align** opció az `<object>` TAG esetén is ugyanúgy működik:

```
<p>Szöveg után elhelyezett felülre pozícionált kép:
<object align="top" data="viragkicsi.jpg"
type="image/jpeg">viragkicsi.jpg</object> Ez pedig a kép
utáni hosszabb szöveg.</p>
<p>Szöveg után elhelyezett alulra pozícionált kép:
<object align="bottom" data="viragkicsi.jpg"
type="image/jpeg">viragkicsi.jpg</object> Ez pedig a kép
utáni hosszabb szöveg.</p>
```

#### 19.2.1.2. Vízszintes igazítás

A vízszintes igazítás is ugyanúgy működik az `<object>` esetén, mint az `<img>`-vel:

```
<p>Szöveg után elhelyezett jobbra pozícionált kép:
<object align="right" data="viragkicsi.jpg"
type="image/jpeg">Virágocska</object> Ez pedig a kép utáni
hosszabb szöveg, esetleg olyan hosszan, hogy több sorba
tördelődjön.</p>
```

### 19.2.2. Hang

Amennyiben a böngészőben szeretnénk hangot lejátszani, minden esetben szükség van további szoftver összetevőkre azon a gépen, ahol az oldal megjelenik.

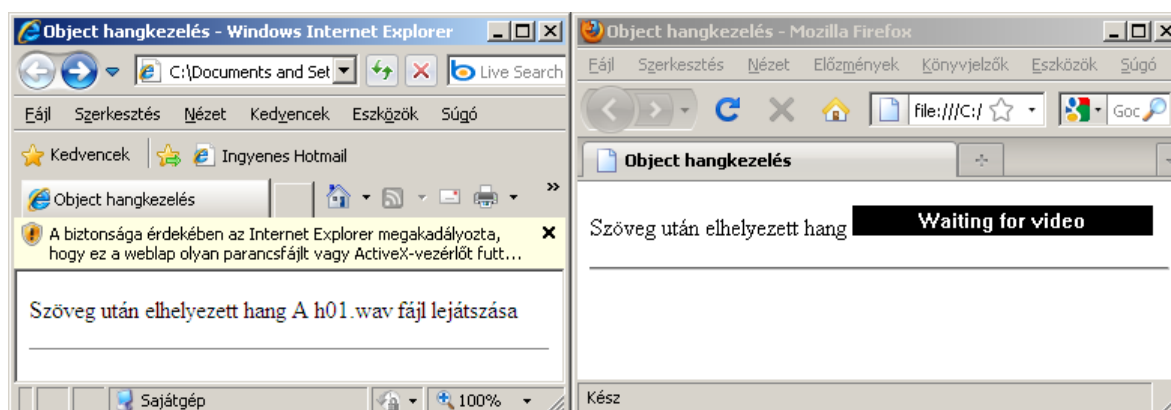
1. Az egyik ilyen összetevő lehet a QuickTime vagy a Windows Media Player bővítmény, ami egyaránt szükséges hangok és videók lejátszásához is. **Ezeket**

**a bővítményeket a felhasználónak kell letöltenie és telepítenie!** Bővebb lista a FireFox bővítményeihez: <https://addons.mozilla.org/hu/firefox/browse/type:7>

2. Ezen kívül elérhetőnek kell lenni a lejátszó számítógépen a lejátszandó fájl kodekjének is, magyarul a számítógépen le kell tudni játszani más programmal is a hang vagy videó fájlt. **A kodekeket is a felhasználóknak kell telepítenie a számítógépre!**

Az eredmény gépenként más és más lehet. Nem lehet garantálni, hogy az adott számítógépen meg fog-e szólalni a hang. Jobb esetben legalább üzenetet kapunk, hogy mi a probléma, illetve hogyan lehet orvosolni azt. A következő példa egy egyszerű .wav fájlt játszana le, de amint látszik is, az adott gépen se az IE8, se a FireFox nem tudta megoldani. Az IE8 számára hiányzik a Windows Media Player bővítmény, a FireFox sajnos itt még ennyit sem mond.

```
<p>Szöveg után elhelyezett hang
<object data="h01.wav" type="audio/x-wav" width="200"
height="20">
<param name="autoplay" value="false">
<param name="autoStart" value="0">
A h01.wav fájl lejátszása
</object>
<hr>
```



40. ábra Hang bővítmények nélkül

A bővítmény telepítése után megjelenik a média lejátszó kisméretű kezelő felülete, de sajnos ő se tudja lejátszani:

Léteznek olyan oldalak, ahol a multimédiás képességeket lehet tesztelni. Ilyen oldal például:

- <http://www.student.oulu.fi/~sairwas/object-test/>
- <http://joliclic.free.fr/html/object-tag/en/>

A <http://joliclic.free.fr/html/object-tag/en/object-audio.html#wav> oldal szerint minden böngésző támogatja .wav fájlok lejátszását. Ehhez azonban a rendszeren fel kell lennie telepítve az ehhez szükséges kiegészítőknek!

### 19.2.3. Videó

A videók lejátszásához is szükség van böngésző bővítményre és megfelelő kodekre. Ugyanazt lehet elmondani, mint a hang esetén, vagyis jól beállított számítógépen meg fog jelenni a videó a fontosabb böngészők alatt.

A legelterjedtebb videó formátumok, amiket támogatnia kell a rendszereknek:

- avi
- mpg
- mov
- wmv

Vigyázni kell azonban arra is, hogy **avi** és **mpg** fájl formátum sokféle van. Olyan formátumot kell választani, ami mindenhol rendelkezésre áll.

A következő példa a Windows rendszerben alapban megtalálható **clock.avi** fájlt játssza le a böngészőben:

```
<p>Szöveg után elhelyezett videó:  
<object data="clock.avi" type="video/x-msvideo"  
width="80" height="80">  
Óra  
</object>
```

A szélesség (**width**) és magasság (**height**) megadása nélkül a legtöbb helyen nem jelenik meg a videó!

### 19.2.4. Flash

A flash formátum .swf fájlokban tárolja az adatokat. Ilyen állományok is lejátszhatók az `<object>` TAG segítségével. Azonban a helyes megjelenítéshez ebben az esetben is szükség van a felhasználó gépén a flash böngésző bővítményre. **Ennek telepítését is a felhasználónak kell elvégeznie!**

```
<p>Szöveg után elhelyezett flash animáció:  
<object type="application/x-shockwave-flash"  
data="11planets.swf" width="150" height="100">  
<param name="movie" value="11planets.swf">  
<param name="loop" value="false">  
<a href="11planets.swf">11planets.swf</a>  
</object>
```

Ugyanez a kód Internet Explorer 8 alatt is működik, ha engedélyezzük az ActiveX vezérlőt:

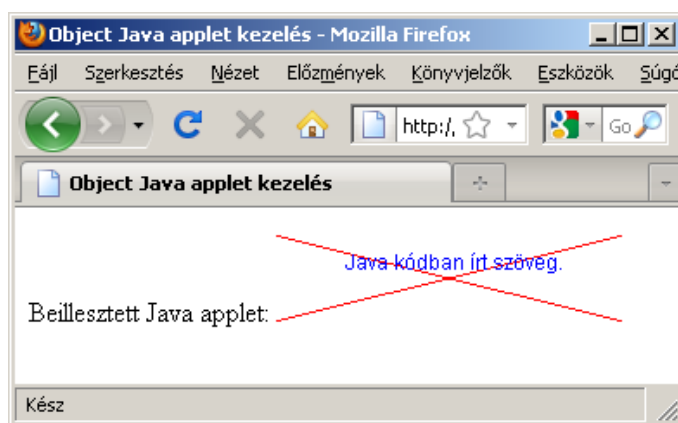
### 19.2.5. Java applet

A Java appletek Java nyelven írt programok, amelyeket be lehet illeszteni a weboldalakra. A Java programok forrását .java fájlokban szokás tárolni. Ezeket a forrás fájlokat a felhasználás előtt le kell fordítani valamilyen fordító program (pl.: a JDK - Java Development Kit vagy SDK javac) segítségével. A lefordított programocskát (applet) rendszerint .class kiterjesztésű fájlban kerül tárolásra.

Az `<object>` TAG segítségével könnyedén beilleszthetők Java appletek az oldalba, arra azonban figyelni kell, hogy a normális megjelenéshez szükség van egy futtató környezetre is (JRE - Java RunTime Environment). A futtató környezet része kell legyen egy böngésző bővítmény (Java plugin) is. **Ennek telepítését a felhasználónak kell elvégeznie!**

A következő példa egy 200x50-es területet határoz meg a *JavaApplet* nevű Java applet részére, ami ki is használja ezt:

```
<p>Beillesztett Java applet:
<object codetype="application/java"
classid="java:JavaApplet.class" width="200" height="50">
Egy Java applet minta lenne itt!
</object>
</p>
```



41. ábra Java applet FireFox-ban

A felhasznált Java applet kódja:

```
import java.applet.*;
import java.awt.*;

public class JavaApplet extends Applet {
    public void paint(Graphics g) {
        int xmax=200; int ymax=50;
        g.setColor(Color.red);
        g.drawLine(0,0,xmax-1,ymax-1);
        g.drawLine(xmax-1,0,ymax-1);
        g.setColor(Color.blue);
        g.drawString("Java kódban írt szöveg.",40,20);
    }
}
```

### 19.3. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mit jelent a multimédia szó?
2. A HTML szabvány szerint hogyan lehet videót beilleszteni az oldalba?
3. Mi az a **flash**?

4. Mi a különbség a Java program és a Java applet között?
5. Mire szolgálnak a MIME type-ok?
6. Hogyan kell egy mpg fájl esetén a MIME típust meghatározni?
7. Milyen esetekben van szükség kódkekre?
8. Milyen bővítményekre van szükség, ha a mai modern oldalakat hiba nélkül szeretnénk megjeleníteni?
9. Mire szolgál az `<object>` TAG **classid** paramétere?
10. Helye-e a következő kódrészlet?  
`/ <object> classid=image/jpeg code=alma.png </object> /`  
 Indokolja is meg!
11. Lehet-e egy oldalon háttérzenét (felhasználó beavatkozás nélkül automatikusan elindul, és nem is állítható le) lejátszani? Ha igen, akkor azt hogyan lehet és milyen korlátokkal?
12. Illessze be az oldalba az **effects.mpg** videót, 100x100-as méretben! Írja le a szabványos kódot!
13. Helyezzen el egy jpg képet az oldalon! Írja le a kódot, ami minden böngészőben működik!

A kérdésekben a / jelek csak a kódok kezdetét és végét jelölik!

## 20. Komplex feladat

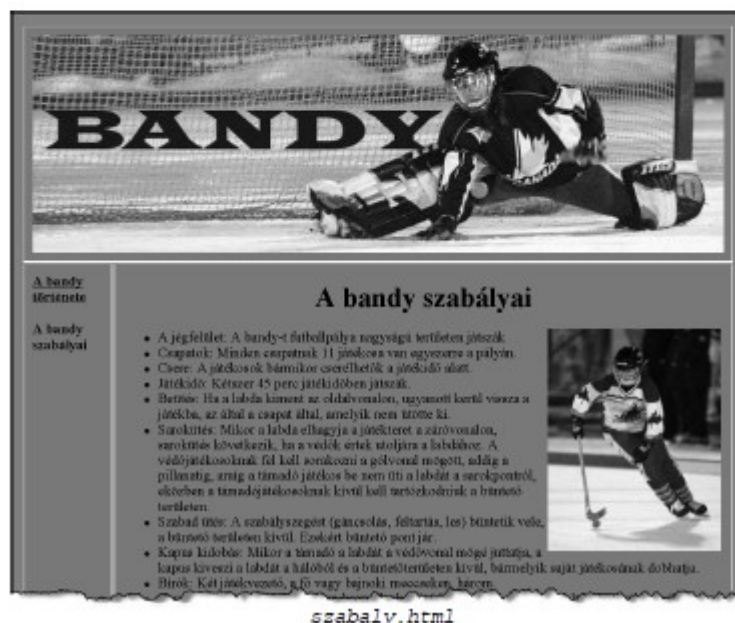
### 20.1. A feladat

A bandy téli sport, mely a jégchokihoz, a focihoz és a gyepabdához hasonlít. A sport történetének és szabályainak bemutatására készítsen két weblapot! A weblapok készítéséhez forrásként a *tortenet.txt*, *szabaly.txt*, *gorog.gif*, *jatekos.jpg* és a *csik.jpg* álmányokat, illetve a megadott mintát használja fel! A szövegfájlok UTF-8 kódolásúak.

1. Hozzon létre két weblapot *tortenet.html* és *szabaly.html* néven! Mindkét weblap következő tulajdonságai egyezzenek meg a két lapon:
  - a. A böngésző címsorában megjelenő cím „BANDY” legyen!
  - b. Az oldal háttérszíne, a szöveg színe és a linkek minden állapotának színe legyen fekete (#000000 kódú)!
  - c. Készítsen egy 2×2-es táblázatot! A táblázat legyen középre igazított, 800 pont széles! A háttérszíne legyen narancssárga (#FF5A18 kódú)! Állítson be 2 pontos szegélyt és 10 pontos cellamargót!
  - d. A táblázat első sorában vonja össze a cellákat, és ide illessze be a *csik.jpg* képet!
  - e. A második sor első cellájának szélességét állítsa 140 pontosra! A cella tartalmát igazítsa függőlegesen felülre a mintának megfelelően!
  - f. Ebbe a cellába írja be és formázza a mintának megfelelően „A bandy története” és „A bandy szabályai” szöveget! A megfelelő szöveget alakítsa a másik oldalra mutató linkké!
2. A *tortenet.html* lapon az elkészített egységes táblázat második sorának második cellájába írja be a „A bandy története” szöveget, majd formázza egyes szintű címsor stílussal, és helyezze vízszintesen középre!

3. A cím alá illessze be a `tortenet.txt` állományból a szöveget! A szöveg legyen sorki-zárt igazítású!
4. Helyezze el és igazítsa a szövegben a `gorog.gif` képet a mintán látható módon! A kép szövegtől való vízszintes távolságát állítsa 5 pontosra!
5. A `szabaly.html` oldalon az elkészített egységes táblázat második sorának második cellájában írja be a „A bandy szabályai” szöveget, és formázza a `tortenet.html` oldalon lévő címmel megegyező módon!
6. A cím alá illessze be a `szabaly.txt` állományból a szöveget! Állítson be felsorolást a szabályokat leíró bekezdésekre!
7. A mintának megfelelően illessze be és igazítsa a `jatekos.jpg` képet!

## 20.2. Minta



## 20.3. Hozzávalók

A feladat megoldásához szükséges fájlok letölthetők egyben is a következő címről:  
[http://tar.inter-studium.hu/HTML\\_Alapok/bandy\\_feladat.zip](http://tar.inter-studium.hu/HTML_Alapok/bandy_feladat.zip)

## 21. A fejléc elemei

### 21.1. A fejléc jelentősége

A fejlécet a `<head>` TAG-ek határolják. Az eddigi tanulmányok alapján itt határozható meg a böngésző címsorában megjelenő szöveg (`<title>` TAG-ek között), illetve itt állítható be a HTML oldal kódlapja (`<meta>` TAG).

A fejlécben elhelyezett utasítások hatása a `<title>`-n kívül nem látható, elsősorban a böngésző programnak szólnak, a böngésző működését befolyásolják. Bizonyos esetekben ezek közül többet is használni kell, így meg kell mindegyikkel ismerkedni.

### 21.2. Lehetőségek

A fejlécben a következő szabványos elemek lehetnek:

- `<title>` - böngészőablak címsorának szövegét határozza meg, csak egyszer szerepelhet
- `<base>` - az itt szereplő URL határozza meg a báziscímet, melyből a relatív címeket értelmezni kell. Az intelligens kiszolgálók korábban nem kötelező megadni. Ez is csak egyszer szerepelhet.
- `<meta>` - különböző egyedi adatok adhatók itt, mint pl. a dokumentum alkotója, a létrehozó program, rövid tartalom, illetve ennek segítségével adhatók meg egyedi HTTP protokoll fejléc információk, többször is előfordulhat
- `<link>` - a dokumentum kapcsolatait lehet itt meghatározni
- `<style>` - beágyazott stíluslap (CSS) információk elhelyezésére szolgál, többször is előfordulhat
- `<script>` - a dokumentum kliens-oldali scriptjeit határozza meg, többször is szerepelhet
- `<noscript>` - egy alternatív lehetőség arra az esetre, ha a `<script>` nem támogatott

A következő elemek már nem támogatottak, de transitional DTD-val használhatók:

- `<basefont>` - az oldal alapértelmezett betűméretét lehet itt meghatározni, csak egyszer, a HTML szabvány már nem támogatottként jelöli, és a böngészők is különbözőképpen kezelik, használata nem ajánlott
- `<isindex>` - egysoros szövegbeviteli mező jelenik meg, helyette **text** típusú `<input>` elem használata javasolt

#### 21.2.1. A title elem

A `<title>` TAG-en belüli szöveg fog megjeleni a böngésző címsorában, vagyis gyakorlatilag itt határozzuk meg az oldal címét:

#### 21.2.2. A meta elem

A `<meta>` TAG segítségével egyedi információkat (tulajdonságok) lehet elhelyezni egy-egy HTML dokumentumban. Egy részük elsősorban a kereső rendszereknek szólnak, és csak plusz információkkal látják el az oldalt (pl.: készítő neve), más részük HTTP fejléc információkat helyez el az oldalba.



Egy HTML dokumentumban bármennyi `<meta>` szerepelhet és nincs záró TAG-je. Négy paramétere lehet:

- **name** - egyedi tulajdonság nevét határozza meg, az értéket ilyenkor a content fogja tartalmazni
- **http-equiv** - a HTTP fejléc jellemzőjének meghatározására szolgál, a jellemző értékét pedig a content fogja tartalmazni
- **content** - a name és a http-equiv által meghatározott tulajdonság értékét tartalmazza
- **schema** - a megadott tulajdonság típusát lehet meghatározni (pl.: ISBN)

Eddig csak a dokumentum kódlapjának meghatározására használtuk, ahol a **http-equiv** jellemzőn keresztül definiáltuk a kódlapot:

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

Itt is egyszerre kellett megadni a **http-equiv** és a **content** paramétert is. Az elsőben a tulajdonság nevét adjuk meg, a másodikban az értékét. Ugyanígy kell alkalmazni a **name** paraméter esetén is.

A `<meta>` TAG lehetőségeiről a következő címen lehet részletesebben olvasni:

[http://www.metatags.org/all\\_metatags](http://www.metatags.org/all_metatags)

#### 21.2.2.1. A name paraméter használata

A `<meta>` TAG **name** paraméterét akkor alkalmazzák, ha a kereső programok számára egyedi információkat szeretnének megadni. Nincsen szabványos elfogadott listája a lehetséges tulajdonság neveknek, a HTML dokumentumok szerzői maguk dönthetik el, hogy milyen adatokat közölnek ennek segítségével.

A keresőprogramok azonban használják a következő tulajdonságokat:

- **keywords** - az oldallal kapcsolatos fontosabb kulcsszavakat lehet itt meghatározni vesszővel (,) elválasztva
- **description** - az oldal rövid (1000 karakterig) leírását lehet itt elhelyezni
- **robots** - a kereső robotok működését lehet itt befolyásolni bizonyos utasításokkal
- Gyakran előfordulnak a következő tulajdonságok is:
- **author** - készítő nevét lehet megadni
- **generator** - a kód előállítását végző program neve szokott itt szerepelni, egyes programok automatikusan kitöltik

Ezek egyike se jelenik meg az oldalon!

#### 21.2.2.2. A robots tulajdonság

A Keresőrobotok működését szabályozó `<meta>` tag, mely meghatározza, hogy hogyan indexeljék (vagy hogyan ne indexeljék) be az oldalt a robotok. A lehetséges értékei (több is szerepelhet egyszerre):

- all
- none
- index



- noindex
- follow
- nofollow

Az alapértelmezett az **all**, ekkor minden beindexelhető. Az **index** azt jelenti, hogy az adott oldal beindexelhető, a **follow** pedig azt, hogy az oldalon található linkek követhetők. A **none**, **noindex**, **nofollow** értelemszerűen ennek az ellentétét jelenti. (Ha csak a **noindex** van megadva, akkor az oldal nem beindexelhető, a rajta levő linkek azonban követhetők.)

Mindezek mellett a **robots.txt** fájl segítségével is elérhetjük a kívánt hatást. Bővebben a <http://www.robotstxt.org/wc/exclusion.html> oldalon lehet olvasni a lehetőségekről.

### 21.2.2.3. A http-equiv paraméter használata

A **http-equiv** jellemzőben használható fontosabb tulajdonságok:

- **Content-Type** - az oldal kódlapjának meghatározására szolgál

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

- **Content-Language** - az oldal tartalmának nyelvét határozhatjuk meg itt, hasonlóan a **html** TAG **lang** attribútumához

```
<meta http-equiv="Content-language" content="hu-HU,hu" />
```

- **Last-Modified** - az oldal utolsó módosításának dátumát lehet itt megadni, esetleg a kereső rendszerek is használhatják arra, hogy kell-e frissíteni a nyilvántartásukat

```
<meta http-equiv="Last-modified" content="Sun, 12 Jan 2010 10:32:52 GMT" />
```

- **Refresh** - időzítve másik oldalra lehet automatikusan átirányítani a felhasználót, a következő példában az oldal betöltése után 5 másodperccel automatikusan megnyílik a w2 oldala:

```
<meta http-equiv="Refresh" content="5; URL=http://www.w3.org" />
```

- **Expires** - a dokumentum lejáratí idejét lehet itt meghatározni, elvileg használhatják ezt az adatok a kereső rendszerek is, hogy megállapítsák, mikor érdemes újra indexelniük, valamint egyes böngészők is használják, hogy megállapítsák érdemes-e cache-elni (átmenetileg tárolni)

```
<meta http-equiv="Expires" content="25 mar 2010 15:50" />
```

- **Revisit-After** - itt lehet meghatározni, hogy a kereső rendszerek hány nap múlva keressék fel megint az oldalt, sajnos nem minden kereső támogatja

```
<meta name="Revisit-After" content="10 days" />
```

Egy teljesebb lista található a következő címen (angol):

<http://vancouver-webpages.com/META/metatags.shtml>

### 21.2.3. A style elem

A `<head>` elemek közötti `<style>` TAG segítségével stíluslap (CSS) meghatározásokat lehet elhelyezni az adott oldalra vonatkozóan. Van záró TAG-ja, többször is szerepelhet, és minden böngésző helyesen értelmezi.

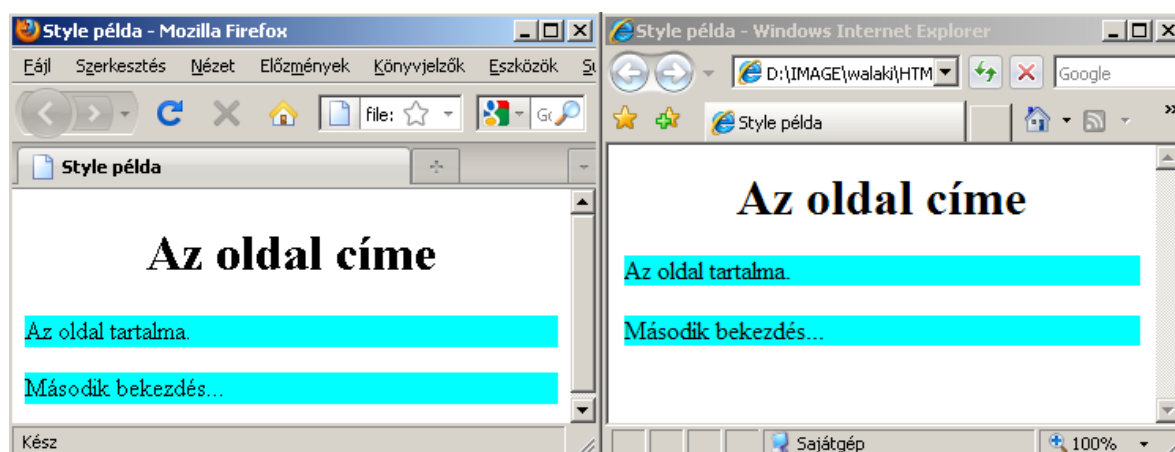
Három paramétere lehet:

- **type** - kötelező megadni, ez határozza meg a stílus leírónyelv típusát, tipikusan *"text/css"* az értéke
- **media** - meghatározza, hogy milyen kimeneti eszköz esetén kerüljön alkalmazásra, pl.: *screen* (képernyő) vagy *print* (nyomtató)
- **title** - nevet lehet adni a stílus információknak

A következő példa stíluslap segítségével beállítja a bekezdések (`<p>` TAG) háttérszínét világoskékre:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<html lang="hu">
<head>
  <title>Stílus példa</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-2" />
  <style type="text/css" title="monitor" media="screen">
    p {background-color: cyan;}
  </style>
</head>
<body>
  <h1 align="center">Az oldal címe</h1>
  <p>Az oldal tartalma.</p>
  <p>Második bekezdés...</p>
</body>
</html>
```

42. ábra A style elem használata (forrás)



43. ábra A style elem használata

### 21.2.4. A script elem

A `<script>` segítségével a dokumentumon belüli *kliens-oldali* scripteket lehet definiálni. A *kliens-oldali* kifejezés azt jelenti, hogy az itt meghatározott kód (program), a felhasználó gépén kerül végrehajtásra. Rendszerint javascript programok beillesztésére használatos, de további script nyelvek (*text/vbscript* vagy *text/tcl*) is támogatottak a szabvány szerint.

A `<script>` többször is szerepelhet, és nem csak a fejlécben (`<head>` elemek között), hanem a dokumentumtörzsben (`<body>` elemek között) is. Van záró TAG-je is, a script kódját a nyitó és záró TAG-ok közé kell elhelyezni.

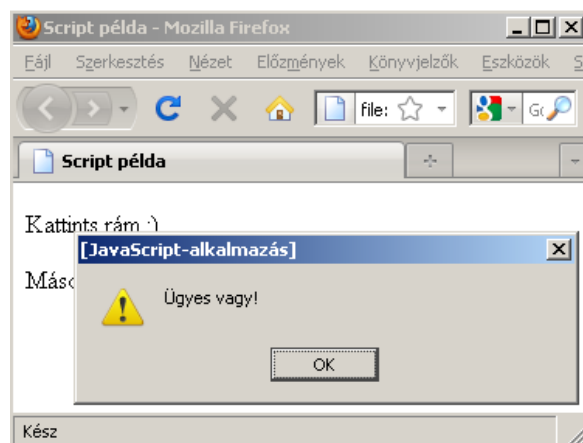
A `<script>`-nek 4 paramétere lehet:

- **type** - kötelező megadni, a script típusát kell itt elhelyezni (pl.: *text/javascript*)
- **src** - ha a scriptet külön fájlban tároljuk, itt kell megadni a fájlra való hivatkozást
- **charset** - a külső script fájl kódolását lehet itt megadni (az alapértelmezés a Latin1)
- **defer** - értéke true vagy false lehet, true esetén elhalasztásra kerülnek a scriptből való dokumentum tartalom módosítások

A következő példában, ha a megadott szövegre kattint a felhasználó, megjelenik egy üzenet. A forrásban a JavaScript kódját megjegyzések közé (`<!-- ... -->`) célszerű elhelyezni, hogy abban az esetben se "rontsa el" az oldal megjelenését, ha a böngésző nem támogatja a JavaScript-et.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<html lang="hu">
<head>
  <title>Script példa</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-2" />
  <script type="text/javascript">
    <!--
      function jsonclick() {
        alert('Ügyes vagy!');
      }
    -->
  </script>
</head>
<body>
  <p onclick="jsonclick()">Kattints rám :)</p>
  <p>Második bekezdés...</p>
</body>
</html>
```

44. ábra A script elem használata (forrás)



45. ábra A script elem használata

### 21.2.5. A noscript elem

Egyes böngészők nem képesek végrehajtani a JavaScript programokat. Ilyen esetekben használható a `<noscript>`. A `<noscript>`-en belüli elemek csak akkor kerülnek megjelenítésre, ha a `<noscript>` előtt definiált `<script>` végrehajtása sikertelen.

Az oldalra szöveget kiírni csak a dokumentumtörzsben (`<body>` TAG-ek között) lehet, így a fejlécben elhelyezett `<noscript>` nem hatásos, a törzsben elhelyezve azonban igen:

```
<p onclick="jsonclick()">Kattints rám :)
<noscript>A böngésző nem támogatja a JavaScriptet!
</noscript>
</p>
```

### 21.2.6. A link elem

A `<link>` elem definiálja a dokumentum kapcsolatait. A `<head>` szekcióban tetszőleges számban helyezhető el. A legtöbb böngésző támogatja ezt az elemet. A **rel** és **rev** attribútumok határozzák meg a kapcsolat természetét a dokumentum és a kapcsolt erőforrás között. Lehetséges attribútumai:

- **rel** - hivatkozáskapcsolatot határoz meg az aktuális dokumentumtól a kapcsolt erőforrás felé
- **rev** - ellenkező irányú kapcsolatot fejez ki
- **href** - a hivatkozás megadása
- **type** - a hivatkozás tartalmának típusa
- **media** - itt lehet megadni, hogy a hivatkozott elem milyen típusú médiára van tervezve
- **hreflang** - a hivatkozás nyelvét lehet megadni
- **charset** - a hivatkozott dokumentum karakterkódolását lehet meghatározni

Leggyakrabban külső stíluslapok (CSS) meghatározásakor használják. Ilyenkor a stílus információk egy vagy több különálló állományban kerülnek tárolásra. Ennek formája:

```
<link rel="stylesheet" href="style.css" type="text/css"
media="screen" />
```

Itt is alkalmazható a relatív hivatkozási forma, vagyis a példában a **style.css** fájlt ugyanabba a könyvtárba elhelyezve, mint a HTML dokumentumot, elég csak a fájl nevét megadni.

## 21.3. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. HTML dokumentum esetén mit nevezünk fejlécnek?
2. Hogyan lehet meghatározni, hogy az oldal megjelenítésekor a böngésző címsorában a "Kis Pista oldala" szöveg jelenjen meg? Írd le a teljes HTML kódot, vagyis készíts egy szabványos HTML fájlt!
3. Mit határoz meg a következő kódrészlet?

```
/<base href="http://endre.atw.hu/sajat/index.html" />/
```

4. Mit jelent a relatív hivatkozás egy link esetén?
5. Hol lehet használni a `<meta>` TAG-et?
6. Milyen paraméterei lehetnek a `<meta>` TAG-nek?
7. Az eddigi HTML elemekhez képest miért különleges a `<meta>` TAG?
8. Mire használatos a **content** paraméter?
9. Mik azok a keresőprogramok? Nézz utána az interneten, ha szükséges!
10. Egy HTML dokumentumban kötelező-e megadni az **author** tulajdonságot? Indokold!
11. Mik azok a stíluslapok?
12. Írja le a kódot, ami stíluslap információkat határoz meg a dokumentumon belül!  
Például legyen az egyes szintű címsor háttere piros színű!
13. Mi az a JavaScript?
14. Hogyan kell a HTML kódba JavaScript programot elhelyezni? Írd le a kódot!
15. Hogyan lehet a HTML kódban meghatározni azt, ha a stíluslap információk külön fájlban vannak elhelyezve (hogyan lehet ezekre hivatkozni)? Írd le a kódot!

A kérdésekben a / jelek csak a kódok kezdetét és végét jelölik!

## 22. Térképek

### 22.1. Mi az a térkép?

A térkép gyakorlatilag egy egyszerű kép, amin megfelelő parancsok segítségével különböző területeket lehet definiálni úgy, hogy az egyes területek különböző hivatkozásként működhetnek. Vagyis egy képen több hivatkozás is lehet.

Ez a technika ténylegesen is használható térképekhez, például úgy, hogy a magyarországi megyékhez tartozó oldalakat egy térképen lehet kiválasztani.

### 22.2. Térkép *img* elemmel

A régebbi eljárás szerint először definiálni kell az egyes területeket a `<map>` és az `<area>` TAG-ek segítségével, majd a kép beillesztésekor az `<img>` **usmap** attribútumával hozzá kell rendelni a képhez a megfelelő térképet.

A `<map>` TAG-nek van nyitó és záró TAG-je is. Egyetlen kötelező attribútuma van, a **name**, amivel nevesíthető a terület. A **name**-ben megadott elnevezés segítségével lehet hozzárendelni a területet a képhez.

```
<map name="terulet_neve"> ... </map>
```

A `<map>` tagokon belül használható az `<area>`, amivel a tényleges területeket lehet meghatározni. Egy **map** utasításon belül tetszőleges számú `<area>` utasítás lehet. Az `<area>` elemnek nincs záró tagja. Lehetséges attribútumai:

- **shape** - a terület típusát lehet itt meghatározni, értékei lehetnek:
  - **default** - a beillesztett teljes képre vonatkozik
  - **rect** - egy téglalapot lehet meghatározni a **coords** attribútummal
  - **circle** - egy kört lehet meghatározni a **coords** attribútummal
  - **poly** - egy tetszőleges sokszöget lehet meghatározni a **coords** attribútummal
- **coords** - a **shape**-ben meghatározott elem pixel koordinátáit kell itt megadni, vesszővel elválasztva

- **rect** esetén 4 értéket, *balx,baly,jobbx,jobby* alakban, ahol a *balx,baly* a bal felső sarok koordinátája, a *jobbx,jobby* pedig a jobb alsó saroké
- **circle** esetén 3 értéket kell megadni, *x,y,r* alakban, ahol az *x,y* a kör középpontjának a koordinátái, míg az *r* a sugara
- **poly** esetén *x,y* értékeket határozhatunk meg, gyakorlatilag bármennyi képpontban, azonban az első és az utolsó koordinátáknak egyezniük kell, ha zárt területet szeretnénk meghatározni
- **href** - az adott területhez tartozó hivatkozást határozhatjuk meg, ugyanúgy, mint az `<a>` TAG **href** attribútumánál
- **alt** - a képeknél megszokott módon egy alternatív szöveget lehet megadni, ami akkor jelenik meg, ha a böngésző valamilyen okból nem tudja megjeleníteni a képet
- **title** - az itt meghatározott szöveg akkor jelenik meg, ha az egeret az adott terület felé viszi a felhasználó, a mai böngészők mindegyike támogatja már
- **accesskey** - egy gyorsbillentyűt lehet megadni itt, amit, ha lenyom a felhasználó, az ugyanaz, mintha a terület felett kattintott volna, sajnos nem minden böngésző támogatja

A következő példa egy *mapneve* nevű térképet határoz meg, ahol 20 képpontú kör és egy 40 képpontos négyzet van meghatározva:

```
<map name="mapneve">
<area shape="circle" coords="100,100,20" alt="kör" />
<area shape="rect" coords="0,0,40,40" alt="negyzet" />
</map>
```

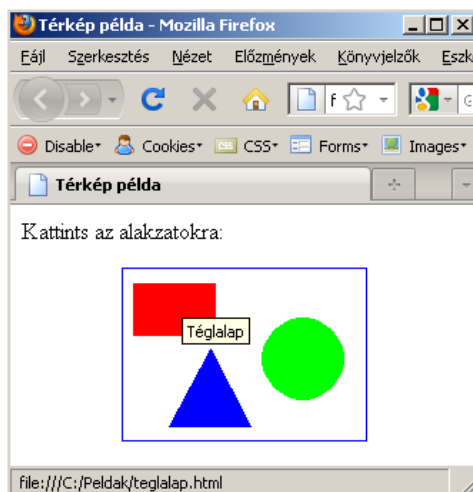
### 22.2.1.1. Példa az img és map parancsokra

A következő példa egy előre elkészített képen 3 objektumot határoz meg a képnek megfelelően, egy téglalapot, egy kört és egy háromszöget. Mindegyik elemet külön ki lehet választani, és mindegyikhez különböző hivatkozás tartozik:

```
<body>
Kattints az alakzatokra:<br />
<p align="center">

</p>
<map name="alakzatok">
<area href="teglalap.html"
shape="rect"
coords="7,10,64,46"
alt="teglalap"
title="Téglalap" />
<area href="haromszog.html"
shape="poly"
coords="32,110,89,110,61,56"
alt="haromszog"
title="Háromszög" />
<area href="kor.html"
shape="circle"
coords="125,63,30"
alt="kör"
title="Kör" />
</map>
</body>
```

46. ábra Térkép img és map elemmel (forrás)



47. ábra Térkép img és map elemmel

## 22.3. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mire szolgálnak a térképek HTML kódban?
2. Térképek készítésekor milyen TAG-eket kell használni?
3. Hol kell megadni a `<map>` TAG-et?
4. Hogyan kell definiálni egy 30 pixeles kört a kép bal sarkában? Írd le a kódot!
5. Hol és hogyan lehet beállítani a hivatkozásokat?
6. Mi határoz meg a **shape** attribútum?
7. Milyen értékei lehetnek a **shape**-nek?
8. Mit határoz meg a `<map>` TAG **name** attribútuma?
9. Hogyan kell felhasználni a `<map>` **name** atribútumának értékét?
10. Mit jelent, ha a **title** attribútumnak az értéke "térkép"?

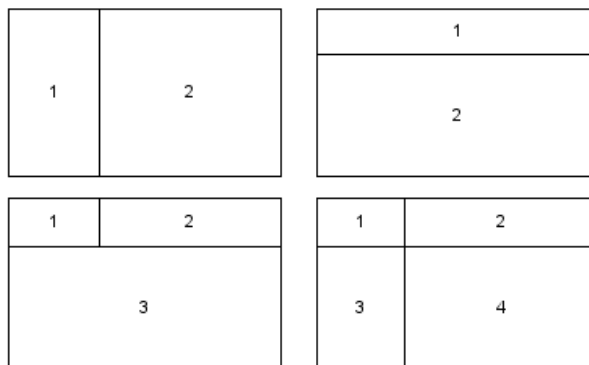
## 23. Keretek

### 23.1. Keretek fogalma

Egy weboldalt nem csak táblázatokkal lehet részekre osztani, hanem a `<frameset>`, `<frame>` utasítások segítségével is. Ráadásul ebben az esetben már komplett html fájlokat lehet betölteni az oldal különböző részeibe. Ezek a részek vonalakkal lesznek elválasztva egymástól, amikkel a felhasználó módosíthatja az egyes részek méretét. De arra is lehetőség van, hogy ne engedjük meg ezt a módosítást.

Minden egyes terület külön sorokra és oszlopokra osztható, amelyek aztán újabb sorokra és oszlopokra oszthatók, minden egyes elemhez külön html oldalakat lehet hozzárendelni.

Néhány példa a keretek alkalmazására, ahol minden egyes szám egy html fájlt jelöl:



48. ábra Keretek alkalmazása

## 23.2. Kétoszlopos minta

Kétoszlopos oldal megjelenítéséhez 3 állományra van szükség:

- a kiinduló keretet meghatározó (**keretek\_fo.html**)
- a bal oldali részt meghatározó (**keretek\_bal.html**)
- illetve a jobb oldali részt meghatározó (**keretek\_jobb.html**)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "
http://www.w3.org/TR/html4/frameset.dtd">
<html lang="hu">
<head>
<title>Keretek</title>
<meta http-equiv="Content-type" content="text/html; charset=iso-8859-2">
</head>
<frameset cols="22%,78%">
<frame src="keretek_bal.html" name="menu" marginwidth="10" marginheight
="10" noresize noresize="true">
<frame src="keretek_jobb.html" name="tartalom" marginwidth="10"
marginheight="10" noresize>
</frameset>
</html>
```

49. ábra Kétoszlopos minta keretek\_fo.html (forrás)

Amint látszik ennek a fájlnek a felépítése eltér a hagyományos HTML fájlok felépítésétől. Itt nincs `<body>` TAG, helyette a `<frameset>` TAG-et kell alkalmazni. A `<frameset>`-nek is van nyitó és záró TAG-ja is, valamint lehetnek attribútumai is. A példában egyetlen attribútum van, a **cols**, ami most a kétoszlopos megjelenítést állítja be 22%- illetve 78 százalékra. Ezen kívül a dokumentum típust is át kell állítani a példának megfelelően frameset-re:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

Az egyes keretek egyedi jellemzőinek beállításait külön `<frame>` TAG-ek segítségével lehet megtenni.

A keretekben elhelyezett fájlok (**keretek\_bal.html** és **keretek\_jobb.html**) már teljesen "normális" HTML fájlok, tetszőleges tartalommal.



### 23.3. Keretek meghatározása

A `<frameset rows="oszlophatárok">` kezdőutasítással osztható fel a képernyő függőlegesen, a `<frameset cols="sorhatárok">` utasítással pedig vízszintesen. Ahol az oszlop- és sorhatárok megadhatók képernyőpontban ill. százalékosan - vesszővel elválasztva -, a maradék képernyőterületre pedig a \* dzsókerkarakter használatával lehet hivatkozni. Mivel vagy csak vízszintesen, vagy csak függőlegesen osztható fel a képernyő, ezért a mindkét irányban osztott böngészőablak létrehozásához a `<frameset>` elemeket egymásba kell ágyazni! Tehát például egy függőleges felosztáson belül kell vízszintesen elválasztott részekre tagolni egy oszlopot.

A fenti módon definiált területekre a `<frame src="objektum">` utasítás tölti be a megadott objektumot, mely objektum lehet egy teljes HTML fájl, annak egy meghatározott része, ill. egy kép. Az így kitöltendő keretek viselkedését szabályozza az utasítás következő alakja:

```
<frame name="név" src="objektum" scrolling="érték"
marginwidth="szám" marginheight="szám">
```

Az adott keretnek ad nevet a **name** opció, a szkrollozást letilthatja `scrolling="no"` kiegészítés (ezenkívül a **yes** és az **auto** értékeket veheti fel a **scrolling** opció). A **marginwidth** és a **marginheight** pedig a kereten belüli margók szélességét szabályozza.

Az `<a href=...>` utasítás ismeri a `target="név"` opciót. (A `target="_top"` opcióval az egész böngészőablakot elfoglalja a hivatkozott dokumentum, tehát feloldja az ablak keretekre osztását!) Ha ezek egyike sem szerepel, akkor a hivatkozás a hivatkozó objektum keretében jelenik meg, felülírva azt!

### 23.4. Beillesztett keretek

A beillesztett keret azt jelenti, hogy a HTML dokumentumon belül bárhol el lehet helyezni egy másik HTML dokumentumot, külön keretek (`<frameset>`, `<frame>`) használata nélkül. Erre szolgál az `<iframe>` elem.

Az `<iframe>`-nek van nyitó és záró TAG-ja is, és hasonlóan működik, mint az `<object>`. Az `<iframe>` nyitó TAG-jában több attribútum is szerepelhet:

- **src** - a beillesztendő oldal elérése
- **width** - az ablak szélessége
- **height** - az ablak magassága
- **scrolling** - **yes** érték esetén a beillesztett ablaknak lesz görgető sávja, **no** esetén nem lesz
- **frameborder** - az ablak keretének méretét lehet meghatározni
- **marginwidth**, **marginheight** - az ablak körüli margók nagyságát lehet megadni
- **name** - az ablak azonosítására szolgáló név

Az `<iframe>` elemmel meghatározott keret alapból nem lehet átméretezni, így nincs is **noresize** attribútuma.

Az `<iframe>` TAG-ek között, pedig az `<object>`-hez hasonlóan olyan HTML kódot lehet elhelyezni, ami csak akkor jelenik meg, ha az `<iframe>` megjelenésével probléma van, vagyis, ha a böngésző nem támogatja az `<iframe>`-et, illetve, ha a dokumentum beillesztése nem lehetséges.

Az `<iframe>` által beillesztett fájl is egy teljesen „normális” HTML fájl. Az előző `<frameset>`-es megoldáshoz képest azonban annyi az előnye, hogy szabadon elhelyezhetőek tetszőleges számban az oldalon, és a kiinduló állomány is egy „normális” HTML fájl.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<html lang="hu">
<head>
  <title>Keretek iframe</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-2">
</head>
<body bgcolor="#C0C0FF">
  <h1 align="center"> Az oldal címe</h1>
  <p>A következő részben egy másik HTML fájl került beillesztésre:</p>
  <iframe src="keretek_beillesztett.html" width="200" height="120" frameborder="2">
    A böngésző nem támogatja a beillesztett kereteket!<br>
    A megjelenített oldal: <a href="keretek_beillesztett.html">beillesztett oldal</a>
  </iframe>
</body>
</html>
```

50. ábra Az iframe alkalmazása (forrás)

## 23.5. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mit értünk keretek alatt a HTML esetén?
2. Milyen esetekben kell keretes oldalt készítenünk?
3. Milyen TAG-eket használunk keretes oldalak készítésekor?
4. Hány részre lehet osztani egy oldalt?
5. Hogyan lehet olyan oldalt készíteni, ahol van egy különálló felső és alsó csík, közöttük pedig 3 egyenlő részből álló külön területek? Írd le a kódot! Próbáld is ki!
6. Az előző példához hány darab html fájlra van szükség?
7. Lehet-e az egyes területek méreteit módosítani a felhasználónak? Mikor és hogyan?
8. A keretek definícióját tartalmazó fájlban milyen DOCTYPE-ot kell használni?
9. Mit határoz meg a következő kód részlet?  
`/<frameset rows="10%,80%,10%">/`
10. Mi a különbség a `<frame>` és az `<iframe>` elemek használata között?
11. Mi szerepelhet az `<iframe>` nyitó és záró TAG-ja között?

**A / jelek csak a HTML kódok elejét és végét jelzik!**

## 24. Keretek alkalmazása

### 24.1. Lehetőségek

- **Oldal szerkezet kialakítására**

Kereteket elsősorban az oldal szerkezetének kialakítására használnak. Segítségével könnyedén megoldható, hogy egyetlen oldalon különböző szerkezetű részek legyenek. Pl.: az oldal tetején és bal oldalon középre rendezett szöveg, még a jobboldali rész sorkizárt.

- **Menüvezérelt oldal készítésére**

A menü lehetőséget biztosít a felhasználónak, hogy szabadon válasszon a lehetséges oldal tartalmak közül. A keretek segítségével könnyen megvalósítható az, hogy a tartalmi rész változzon, míg a menü, illetve egyéb (pl. fejléc) változatlan maradjon.

- **Egyedi igények**

Az, hogy alkalmazunk kereteket vagy sem, főként az oldal készítőjétől függ. Neki kell döntenie, hogy az adott igényekhez alkalmazható-e keret vagy sem. Néhány esetben sokkal egyszerűbb beillesztett keretek (iframe) alkalmazása, mintha a teljes oldalt keretbe ágyaznánk.

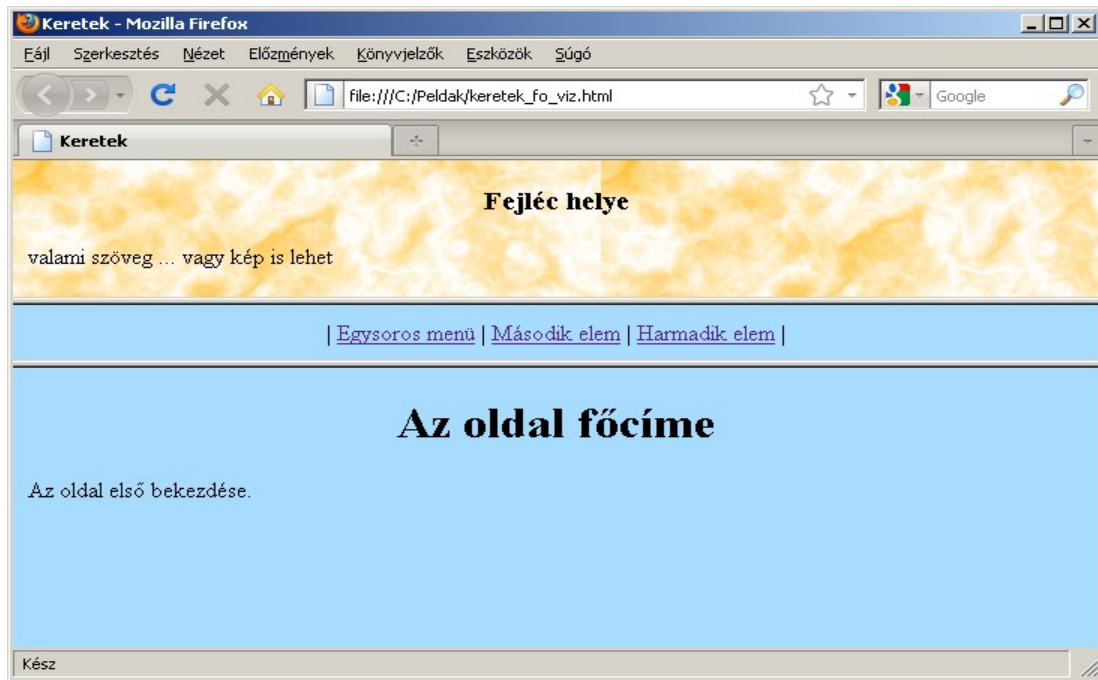
#### 24.1.1. Vízszintes felosztás

A következő példa három különálló, vízszintesen elválasztott részre osztja az oldalt. A felső részen elhelyezhető az oldal fejléce, a következő részen a menü, míg a harmadik részen a tartalmat lehet megjeleníteni.

A felosztást a `<frameset rows="100,40,*">` utasítás végzi el, ahol az első szám (100) a felső rész magasságát határozza meg, a második (40) a menü magasságát, míg a \* azt jelenti, hogy a tartalmi rész magassága mindig a tényleges tartalomnak megfelelő lesz. A `<frameset>`-en belüli `<frame>` utasítások az egyes részek tartalmát és jellemzőit határozza meg. Most mindegyik rész egyformán 10-es margókkal került meghatározásra (`marginwidth`, `marginheight`), és egyik sem méretezhető át (`noresize`).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="hu">
<head>
  <title>Keretek</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-2">
</head>
<frameset rows="100,40,*">
  <frame src="keretek_fejlec.html" name="fejlec" marginwidth="10" marginheight="10" noresize="true">
  <frame src="keretek_menu.html" name="menu" marginwidth="10" marginheight="10" noresize="true">
  <frame src="keretek_tartalom.html" name="tartalom" marginwidth="10" marginheight="10" noresize="true">
</frameset>
</html>
```

51. ábra Vízszintes felosztású keret, kiinduló állomány (forrás)



52. ábra Vízszintes felosztású keret

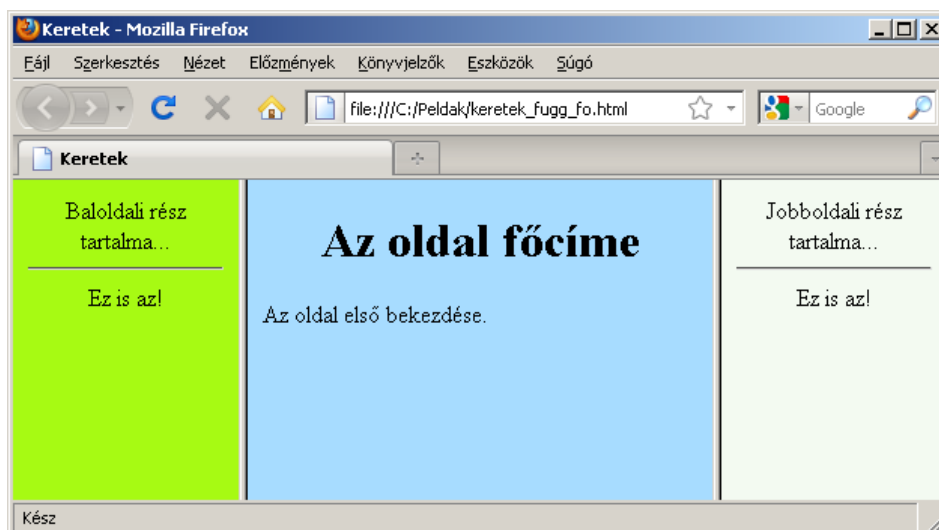
### 24.1.2. Függőleges felosztás

A függőleges felosztásra már volt egy minta a keretek ismertetésénél. Ott két részre lett felosztva az oldal, az oldal teljes szélességének 22 %-a volt a bal oldali rész, míg a jobboldali a maradék 78%:

A következő példában 3 részre kerül felosztásra az oldal úgy, hogy a baloldali és a jobboldali rész is 150 pixel széles, a középső pedig a maradék helyet foglalja el (`<frameset cols="150,*,150">`). Ilyenkor a középső rész szélessége mindig attól függ, hogy a megjelenítéskor a böngésző ablaka milyen méretű.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "
http://www.w3.org/TR/html4/frameset.dtd">
<html lang="hu">
<head>
  <title>Keretek</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-2">
</head>
<frameset cols="150,*,150">
  <frame src="keretek_fugg_bal.html" name="bal" marginwidth="10" marginheight="10"
noresize="true">
  <frame src="keretek_tartalom.html" name="tartalom" marginwidth="10" marginheight="10"
noresize="true">
  <frame src="keretek_fugg_jobb.html" name="jobb" marginwidth="10" marginheight="10"
noresize="true">
</frameset>
</html>
```

53. ábra Függőleges felosztású keret, kiinduló állomány (forrás)



54. ábra Függőleges felosztású keret

Kisebb méretű ablaknál csak a középső rész mérete változik:

### 24.1.3. Vegyes felosztás

Az egyes felosztási módokat szabadon egymásba lehet ágyazni, vagyis egy `<frameset>`-en belül akár több másik `<frameset>` is lehet. Ezzel a technikával tetszőleges oldal szerkezetet elő lehet állítani. A következő példa egy ilyen összetett oldalt jelenít meg, ahol van fejléc, fejléc alatti menürész, lábléc, a középső területnek pedig van bal- és jobboldala, valamint közöttük jelenik meg a tényleges tartalom.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "
http://www.w3.org/TR/html4/frameset.dtd">
<html lang="hu">
<head>
  <title>Keretek vegyes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-2">
</head>
<frameset rows="80,20,* ,50">
  <frame src="keretek_fejlec.html" name="fejlec" noresize="true">
  <frame src="keretek_menu.html" name="fejlecalatt" marginheight="0" noresize="true">
  <frameset cols="150,* ,100">
    <frame src="keretek_fugg_bal.html" name="kozep_bal" noresize="true">
    <frame src="keretek_tartalom.html" name="tartalom" noresize="true">
    <frame src="keretek_fuggjobb.html" name="kozep_jobb" noresize="true">
  </frameset>
  <frame src="keretek_lablec.html" name="lablec" marginheight="0" noresize="true">
</frameset>
</html>
```

55. ábra Vegyes felosztású keret, kiinduló állomány (forrás)

Érdemes megfigyelni, hogy elmaradtak a **marginwidth** és **marginheight** attribútumok, csak két helyen (fejléc és a lábléc) került megadásra, de ott 0-s értékkel. Ha nincs megadva ugyanis, az alapértelmezés 10. Ilyen összetett oldalak esetén csak arra kell figyelni, hogy ahány különálló részt szeretnénk meghatározni, annyi `<frame>` utasításnak kell szerepelnie benne, valamint célszerűen mindegyik ilyen résznek külön HTML fájl kell készíteni.



56. ábra Vegyes felosztású keret

## 24.2. A gyakorlat

Az oldalszerkezet kialakításának lehetőségei:

- keretek
- beillesztett keretek
- táblázatok
- CSS

### Keretek:

Néhány évvel ezelőtt robbanásszerűen elterjedtek a keretes oldalak. Akkoriban minden "modern" oldal kereteket alkalmazott az oldalak szerkezetének kialakításakor. Abban az időben még több olyan böngészőt is széles körben használtak, amik nem támogatták a kereteket, ezért minden oldalt fel kellett készíteni arra az esetre is, ha a böngésző nem támogatja a kereteket.

A keretek alkalmazásakor az egyes részek között mindenképpen van elválasztás (vagy keret vagy üres terület), ez sok esetben megnehezíti a megfelelő kinézet (design) megvalósítását. Ezekben az esetekben a keretek már nem jöhetnek szóba.

### Beillesztett keretek:

A beillesztett keretek (`<iframe>`) ugyanolyan problémát jelentettek, mint a hagyományos keretek (`<frameset>`), vagyis több olyan böngésző is használatban volt, ami nem megfelelően támogatta azokat.

### Táblázatok:

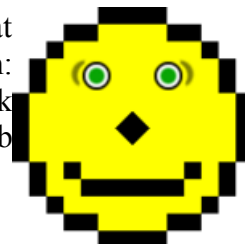
A keretek nem megfelelő támogatása miatt a fejlesztők az oldalak szerkezetének kialakítására elkezdtek alkalmazni a táblázatokat. A mai napig előszeretettel alkalmazzák a táblázatokat ilyen célokra, mivel a böngészők mindegyike megfelelően támogatja azokat.

Ennek ellenére nem ajánlják ilyen célokra, mivel egy összetettebb oldal esetén a kód nehezebben értelmezhető, kevésbé átlátható.

### CSS:

A CSS azzal a céllal került kialakításra, hogy megoldja a fenti problémákat. A szabványban minden megvan, ami a különböző oldalak kialakításához szükséges. Sajnos azonban a böngészők nem egyformán támogatják a szabványban leírtakat. Ugyanaz a stílus beállítás egyik böngészőnél így, másiknál úgy jelenik meg, megnehezítve a weboldalak készítését. Sajnos a ma használt böngészőknél is problémát jelent olyan CSS alapú oldal készítése, amelyik minden böngészőben ugyanúgy jelenik meg.

A böngészők CSS szabvány támogatásának mérésére tesztoldalakat is készítettek. Acid teszteknek is nevezik ezeket. Jelenleg 3 verziója van: [Acid1](#), [Acid2](#), [Acid3](#). Tanulságos lehet ezen tesztek lefuttatása az általunk használt böngészőkön. A kiinduló oldal: <http://www.acidtests.org/>. Bővebb információ az Acid tesztekéről: <http://hu.wikipedia.org/wiki/Acid-tesztek>.



### Összefoglalva:

Profi oldalak ma már nem alkalmaznak kereteket annak ellenére sem, hogy az elterjedt böngészők mindegyike már támogatja azokat. Ennek oka, hogy a keretek közötti elválasztó vonal látványosan "elrontja" az oldal kinézetét. Beillesztett keretekkel azonban több helyen is lehet találkozni.

**A mai oldalak többsége vagy táblázatokat alkalmaz az oldal szerkezetének kialakítására, vagy CSS alapon valósítja azt meg.**

## 24.3. Ellenőrző kérdések

A lecke tanulmányozása után próbálj meg önállóan válaszolni a következő kérdésekre!

1. Mire szolgálnak a keretek?
2. Hogyan lehet 3 egyforma magasságú részre osztani egy oldalt? Írd le a kódot!
3. Hogyan lehet 4 egyforma részre osztani az oldalt? Írd le a kódot! Keress több megoldást!
4. A következő kód milyen oldalszerkezetet határoz meg?  
`<frameset cols="200,50%,*">`
5. Lehet-e kereteken belül beágyazott kereteket használni?
6. Hány beágyazott keret lehet egyetlen oldalon?
7. Miért nem készülnek mostanában keretes oldalak?
8. Milyen hátrányai vannak a keretes oldalaknak?
9. Milyen előnyei vannak a keretes oldalaknak?
10. Milyen előnyei vannak a beillesztett keretek alkalmazásának?
11. Milyen hátrányai vannak a beillesztett keretek alkalmazásának?
12. Milyen alternatívák állnak rendelkezésre a keretek helyett?
13. Mi a probléma CSS alkalmazásakor?
14. Mi az az Acid teszt?

**A / jelek csak a HTML kódok elejét és végét jelzik!**



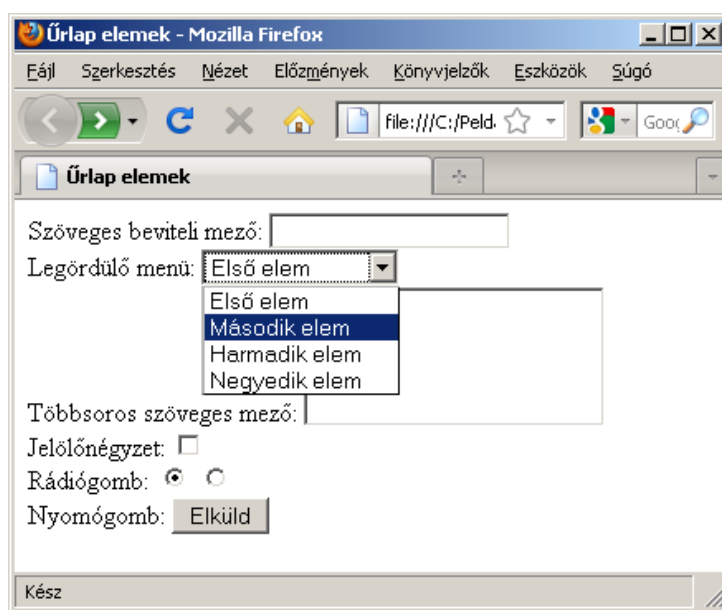
## 25. Űrlapok 1

### 25.1. Alapelemek

Az űrlapok segítségével adatokat lehet kérni a felhasználóktól, vagyis interaktívvá lehet tenni az oldalt. Az adatok megadására többféle lehetőséget is biztosít a HTML nyelv (zárójelben a HTML azonosító nevek):

- egyszerű szöveges beviteli mező (**text**)
- többsoros szöveges beviteli mező (**textarea**)
- jelölőnégyzet (**checkbox**)
- rádiógomb (**radio**)
- egyszerű nyomógomb (**button**, **submit**)
- rejtett mező, amit a felhasználó nem lát, de ezzel ugyanúgy adatokat lehet küldeni (**hidden**)

Ezek az elemek ugyanazok, amikkel a Windows alatti program beállításait lehet elvégezni, vagyis a használatuk nem okozhat senkinek sem problémát. Egy példa az elemek megjelenésére:



57. ábra Űrlap elemek

Az űrlapokon elhelyezett elemek segítségével különböző feladatokat oldhatunk meg :

- a felhasználótól bizonyos adatokat kérhetünk be, melynek strukturáját mi határozzuk meg
- lehetővé tesszük a szerveren tárolt adatok lekérdezését
- megengedjük a felhasználónak, hogy az adatbankot kezelje, adatait módosítsa
- interaktív módon adatokat kérhetünk le egy termékpalettáról, hogy valamit megrendeljünk belőle

### 25.2. Az űrlap (form) működése

Az űrlap célja, hogy adatokat kérjen be a felhasználótól, és valamilyen módon feldolgozza azokat. Az adatok bekérése és feldolgozása rendszerint külön oldalakon történik.



Egyiken megjelenik az űrlap, a beviteli elemekkel, a másik oldalon pedig a beírt értékeket dolgozzuk fel, megjelenítjük, vagy adatbázisba írjuk, vagy csak a megadott adatok alapján más-más elemeket jelenítünk meg.

A bekérési folyamat egyszerű HTML elemekkel megoldható. A feldolgozás azonban nem oldható meg tisztán HTML elemek segítségével. Ehhez mindenképpen valamilyen programra van szükség. A feldolgozó program tetszőleges programozási nyelven készülhet, amelyik képes web-es adatokat fogadni, és szöveges eredményt előállítani. A feldolgozó program készülhet JavaScript-ben, Java-ban, C nyelven, de manapság többnyire PHP nyelven készítik el a feldolgozó rutinokat.

Az űrlap feldolgozási folyamata:

- az első oldal megjeleníti az űrlapot
- a felhasználó adatokat ad meg, vagy választ ki
- a felhasználó elfogadja a megadott adatokat, tipikusan rákattint egy nyomógombra
- a böngésző elküldi az adatokat egy másik oldalnak
- a másik oldal feldolgozza az adatokat valamilyen program segítségével
- a másik oldal megjeleníti az eredményt

A HTML kódban elhelyezett űrlapokat a `<form>` utasítás fogja keretbe. Van nyitó és záró TAG-je is. A nyitó TAG-on belül pedig több paramétert is meg lehet adni. Az egyik ilyen attribútum a **method**, ami a továbbküldés módját határozza meg. Itt ugyanis két lehetőség is van, a **get** és **post**. Ha nem adjuk meg, akkor a **get** az alapértelmezett.

A **get** üzenetküldési mód az oldal URL-jében küldi el a paramétereket. Ez a fajta megoldás nem a legbiztonságosabb, mivel a felhasználó is látja, hogy milyen adatok kerülnek átadásra, és könnyedén lehet ezt módosítani. A **get** metódussal küldött adatok például a következőképpen jelennek meg böngészés közben:

```
http://www.pelda.hu/index.php?Nev=Kiss&Cim=Valahol
```

Ebben az URL-ben a **www.pelda.hu** a gép címe, az **index.php** a feldolgozó program, a **Nev** és a **Cim** a két továbbításra került adat neve, míg a **Kiss** és a **Valahol** a két adat értéke. A '?'-től kezdve a további részeket a böngésző illesztette hozzá a címhez, ha **get** metódussal kerültek továbbításra. A **get** metódussal küldött adatokat JavaScript-el is fel lehet dolgozni.

A **post** üzenetküldési mód ezzel szemben a felhasználó számára láthatatlan módon továbbítja az adatokat. Azonban ilyen esetekben mindenképpen programra van szükség, itt már a JavaScript sem működik.

### 25.3. A form TAG

A űrlap beviteli elemeit a `<form>` utasításnak kell körbefognia. Lényeges megkötés, hogy a `<form>` elemen belül nem lehet másik `<form>`. A `<form>` fontosabb attribútumai:

- **action** - értéke meghatározza az űrlap által használt program, parancsfájl URL-ét, elérési útját. Ez lehet egy egyszerű e-mail cím is (ekkor az űrlap adatait szöveggént küldi el oda), gyakrabban azonban egy CGI programra/script-re mutat.

- **method** - értéke *get* vagy *post* lehet, az **action** által kijelölt script működését befolyásolja, *get* esetén az URL-ben kerül továbbításra az adat, míg *post* esetén a felhasználótól rejtve
- **name** - az űrlap nevét lehet meghatározni, JavaScript esetén ezzel a névvel lehet hivatkozni a formra
- **id** - CSS és/vagy JavaScript esetén van jelentősége
- **enctype** - megadhatjuk az űrlap adatainak típusát is. Ez alapértelmezés szerint *'application/x-www-form-urlencoded'*, **file** típusú beviteli mező esetén *'multipart/form-data'* értéket kell megadni

Nézzük a következő példát:

```
<form name="teszt" method="get" action="urlapfeld.html">
...
</form>
```

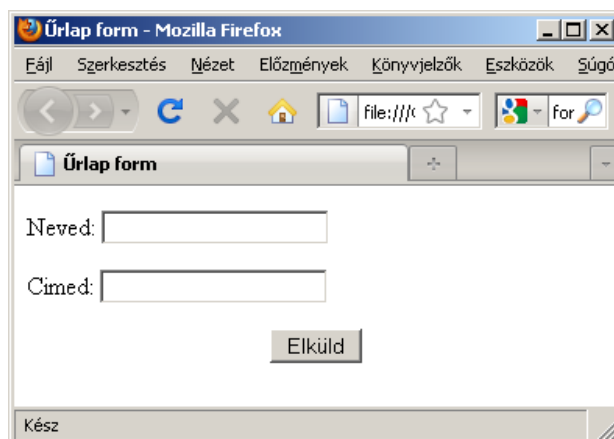
A fenti példában az űrlap neve *teszt*, a továbbküldési mód a *get*, és a feldolgozó oldal az **urlapfeld.html** lesz. A '...' helyén a különböző beviteli mezők leírásának kell szerepelni. De ezen kívül tetszőleges html elemek is elhelyezhetők a form-on, mint a szöveg, a bekezdés, a táblázat, a kép, stb.

Az alábbi példa már életszerű elemeket is tartalmaz a teljesség kedvéért:

```
<form name="teszt" method="get" action="urlapfeld.html">
<p>Neved: <input type="text" name="Nev" /></p>
<p>Cimed: <input type="text" name="Cim" /></p>
<p align="center"><input type="submit" value="Elküld"
/></p>
</form>
```

Az előző példához képest csak a középső rész változott. Amint látszik itt egyszerű szövegek is szerepelhetnek, mint a '*Neved:*' vagy a '*Cimed:*', valamint a szövegek és a beviteli mezők bekezdésekben (`<p>` elem) kerülnek elhelyezésre. A *text* típusú `<input>` elem egy egysoros szöveges beviteli mezőt helyez el az oldalon. Az `<input>` **name** attribútumával azt lehet meghatározni, hogy milyen nevű változóban kerüljön továbbításra a mezőbe írt szöveg. A *submit* típusú `<input>` elem egy nyomógombot határoz meg, amelynek a feliratát a **value** attribútum értékeként lehet megadni. Ha a felhasználó rákattint erre a nyomógombra, akkor `<form>` elem **action** attribútumában meghatározott oldal kerül megnyitásra.

A példát megjelenítve a következő eredményt kapjuk:



58. ábra Egyszerű űrlap

## 25.4. Űrlapkezelés JavaScript-el

A következő példa bemutatja az űrlapok működését és használatát. Az első fájl (**urlap.html**) három szöveges beviteli mezőt és egy nyomógombot jelenít meg. A nyomógomb megnyomásakor a szöveges mezőkbe írt adatokat továbbítja az **urlap2.html** fájl részére, **get** módszerrel, vagyis az oldal linkjében. A fogadáskor meg is jelennek ezek az adatok. A példában:

```
file:///C:/Peldak/urlap2.html?Nev=Nevem+Nincs&Cim=Cimem+Sincs&Tel=3453445
```

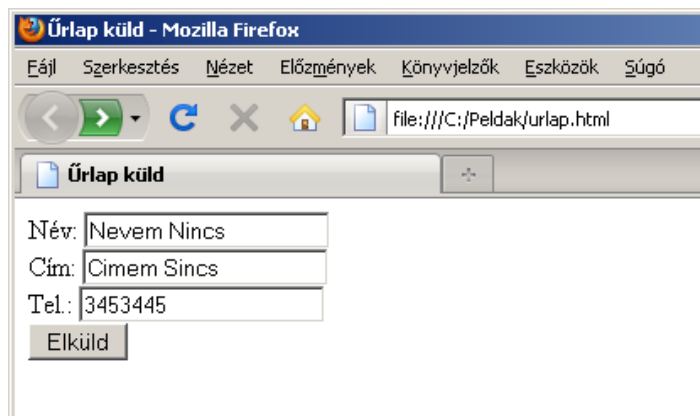
A második fájl a kapott paramétereket JavaScript segítségével kiolvassa és meg is jeleníti. Ezzel a megoldással az a probléma, hogy a speciális karaktereket kódolva küldi át, vagyis például az ékezetes betűk helyett egy % jel és egy hexadecimális szám szerepel. Ezt is vissza lehet alakítani a JavaScript segítségével. A jelenlegi példa azonban nem foglalkozik vele. A cél csak az, hogy bemutassa, hogy egy másik oldalon belül is fel lehet dolgozni az elküldött adatokat.

```
<form name="Adatok" method="get" action="urlap2.html">
  Név: <input type="text" name="Nev" Size="20" /><br />
  Cím: <input type="text" name="Cim" Size="20" /><br />
  Tel.: <input type="text" name="Tel" Size="20" /><br />
  <input type="submit" value="Elküld" />
</form>
```

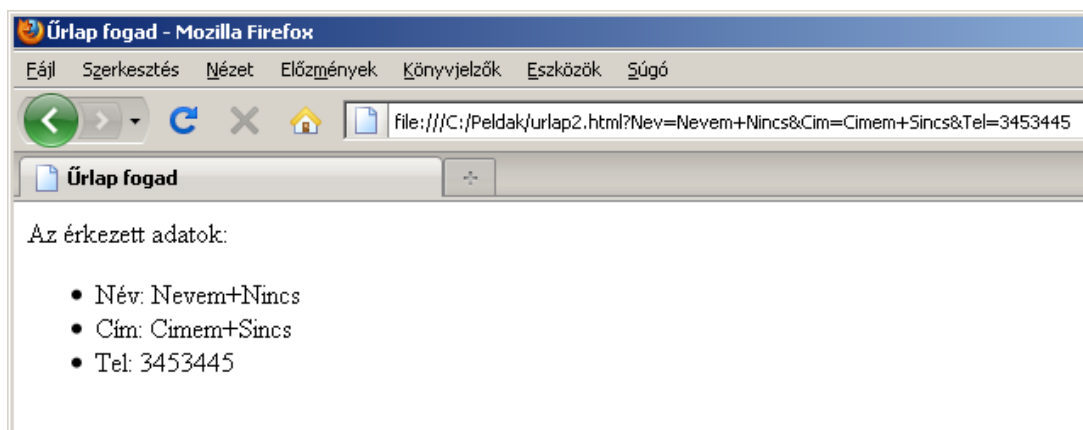
59. ábra Űrlap küldő oldal (forrás)

```
<body>
Az érkezett adatok:
<ul>
  <script type="text/javascript">
    var url = document.URL;
    var par = url.slice(url.indexOf("?")+1);
    var nev = par.slice(par.indexOf("Nev")+4, par.indexOf("&", par.indexOf("Nev")+4));
    var cim = par.slice(par.indexOf("Cim")+4, par.indexOf("&", par.indexOf("Cim")+4));
    var tel = par.slice(par.indexOf("Tel")+4);
    document.writeln("<li>Név: "+nev+"</li><li>Cím: "+cim+"</li><li>Tel: "+tel+"</li>");
  </script>
</ul>
</body>
```

60. ábra Űrlap fogadó oldal (forrás)



61. ábra Úrlap küldő oldal



62. ábra Úrlap fogadó oldal

A szóköz karakter helyett '+' karaktert küld. Ezzel is foglalkozni kellene, de a kód egyszerűsége miatt most ez sincs megoldva.

## 25.5. Az *input* elem

Az űrlap elemei közül ez az elem a legösszetettebb. Amiatt, hogy sokféle beviteli elem meghatározására szolgál, nagyon sok attribútuma is van. Az attribútumok használata attól függ, hogy milyen típusú elemet szeretnénk megjeleníteni. Az `<input>`-nak nincs záró TAG-ja. A legfontosabb közös attribútumok listája:

- **type** - a beviteli elem típusát lehet meghatározni
- **name** - a beviteli elem azonosító nevét lehet megadni
- **value** - a beviteli elem tartalmát (pl. nyomógomb felirata) lehet megadni
- **id** - a a beviteli elem azonosítóját lehet megadni, főként JavaScript használja
- **disabled** - ha értéke **true**, akkor az elem le lesz tiltva, nem lehet használni
- **readonly** - ha értéke **true**, akkor a beviteli elem tartalma nem módosítható
- **tabindex** - az elemek között TAB billentyűvel is lehet váltogatni, a **tabindex**-el a sorrendet lehet befolyásolni
- **accesskey** - az elemekhez rendelhetünk ezzel gyorsbillentyűket

A specifikus attribútumok:

- **checked** - **checkbox** esetén ezzel a paraméterrel lehet meghatározni, hogy kiválasztott-e vagy sem

- **size** - az egysoros beviteli elem méretét lehet meghatározni
- **maxlength** - az egysoros beviteli elembe írható karakterek maximális számát lehet meghatározni
- **src** - kép típusú elem eseté, a kép helyét lehet megadni
- **alt** - kép típusú elem esetén az alternatív szöveget lehet megadni
- **usemap** - klien oldali kép térképet is lehet használni
- **ismap** - szerver oldali kép térképet is lehet használni

A beviteli elem típusát a **type** paraméterrel lehet meghatározni. A **type** lehetséges értékei:

- **text** - egysoros szöveges beviteli mező
- **password** - ez is egy egysoros szöveges mezőt jelenít meg, azonban a beírt karakterek helyett '\*' karakter jelenik meg, tipikusan jelszavak megadásakor használják
- **checkbox** - egy jelölőnégyzetet jelenít meg, amely két állapottal rendelkezik: jelölt (checked) és jelöletlen, a jelölést a négyzetben megjelenő 'pipa' mutatja
- **radio** - egy rádiógombot jelenít, rendszerint több érték közül egynek a kiválasztására használatos
- **image** - kép típusú mező, amelyre kattintva ugyanazt lehet elérni, mint a submit típusú elemmel, vagyis el lehet küldeni az űrlap adatait
- **submit** - egy elküldő nyomógombot határoz meg
- **reset** - ez is egy nyomógombot jelenít meg, azonban ha a felhasználó erre kattint, akkor törli az űrlapon kitöltött mezőket, visszaállítja az alapértelmezést
- **button** - egyszerű nyomógombot határoz meg, ha funkciót szueretnének hozzárendelni, már valamilyen script-re (JavaScript, stb) van szükség
- **hidden** - a felhasználó számára nem látható adattípus, az űrlap kitöltött mezőivel együtt ezeknek az elemeknek az értéke is továbbításra kerül
- **file** - állományok HTTP protokollon keresztüli feltöltésére szolgál, egyegyszerű szöveges mezőt és egy nyomógombot jelenít meg, *Tallózás* felirattal, amelyre kattintva egy fájl kiválasztó ablak segítségével ki lehet választani egyetlen fájlt

### 25.5.1. Szöveges beviteli mezők

Egysoros szöveges beviteli mezőt a következő utasítással is megjeleníthetünk, ahol a **type** attribútummal a beviteli mező típusát adjuk meg, míg a **name** attribútum tartalma azt az azonosító nevet határozza meg, amilyen nevű változóban az elembe írt adat átküldésre kerül:

```
<input type="text" name="mezo_neve" />
```

A szöveges beviteli mezők lehetséges paraméterei:

- **size** - a megjelenített elem szélességét lehet meghatározni
- **maxlength** - a beírható karakterek maximális számát határozza meg
- **value** - a kezdőértéket adhatjuk meg

Kétféle szöveges beviteli mezőt használhatunk:

- egyszerű: `<input type="text" name="Nev" />`
- jelszavas: `<input type="password" name="Nev" />`

A kettő között a **type** attribútum értéke jelenti a különbséget, egyszerű esetén *text*, míg jelszavas esetben *password*.

Használatuk ugyanaz, mindkettőbe tetszőleges szöveget lehet írni. A különbség mindössze annyi, hogy a jelszavas mezőben nem látjuk viszont a beírt karaktereket, helyettük '\*' karakterek jelennek meg.

### 25.5.1.1. A size és maxlength attribútumok

A szöveges mezők megjelenítéskor a böngésző dönti el, hogy milyen széles lesz a beviteli mező, vagyis mennyi helyet foglal el. A beírható karakterek száma alapértelmezésben nincs korlátozva, vagyis tetszőleges hosszúságú karakterláncot be lehet írni. Mindkét jellemzőt szabályozni is lehet, vagyis meg lehet határozni, hogy mennyi helyet foglaljon el a beviteli elem, és azt is meg lehet adni, hogy maximum mennyi karaktert írhatson bele a felhasználó.

#### A size attribútum:

A **size** attribútum tartalma egy szám lehet, ami a beviteli mező méretét határozza meg:

```
<input type="text" name="adat" size="10" />
```

Ennek hatására egy 10 egység széles szöveges beviteli mező jelenik meg.

#### A maxlength attribútum:

A **maxlength** attribútum értéke szintén egy szám lehet, amivel a mezőbe beírható maximális karakter számot lehet meghatározni. A következő példa 4 karakterre korlátozza a beírható karakterek számát:

```
<input type="text" name="adat" maxlength="4" />
```

Ilyen esetekben a felhasználó, aki kitölti az űrlapot, nem tud 4 karakternél többet írni a mezőbe, ugyanis a böngésző nem engedi.

A **size** és **maxlength** attribútumokat együtt is lehet alkalmazni:

```
<input type="text" name="adat" size="5" maxlength="5" />
```

### 25.5.1.2. A value paraméter

Az egysoros beviteli mező esetén is megadható a **value** paraméter. Az itt megadott érték azonnal megjelenik a mezőben úgy, mintha a felhasználó írta volna be, és ha nem módosít rajta, akkor ez az érték kerül továbbításra is. Az űrlap alaphelyzetbe állításakor szintén a **value** paraméterben megadott érték állítódik be.

```
<input type="text" name="Nev" value="Add meg a neved!" />
```

Nem kötelező megadni, de sok helyen ajánlják akkor is, ha nem adunk meg értéket. Ilyenkor a zárójelek között nincs semmi.

```
<input type="text" name="Nev" value="" />
```

### 25.5.2. Nyomógombok

Nyomógombokat is az `<input ... />` utasítással helyezhetünk el az űrlapon. Háromféleképpen lehet nyomógombot megjeleníteni, ahol az eredmény azonos, de a viselkedés különböző. A lehetőségek:

- **submit** - `<input type="submit" value="nyomógomb_szövege" />`
- **reset** - `<input type="reset" value="nyomógomb_szövege" />`
- **button** - `<input type="button" value="nyomógomb_szövege" />`

Az eltérés a **type** attribútum értékében van, minden más azonos. A **value** paraméter értéke határozza meg, hogy milyen szöveg jelenjen meg a nyomógombon. A **button** típusú nyomógomb esetén szükség van más paraméterre (pl.: **onClick**) és **JavaScript**-re is, ha ténylegesen használni szeretnénk.

A **submit** típusú nyomógomb szerepe az űrlap véglegesítése és elküldése. Ha a felhasználó egy ilyen típusú nyomógombra kattint, akkor a teljes űrlap adatai továbbküldésre kerülnek a `<form>` **method** paraméterében meghatározott oldalnak. Ezért egy űrlapon legalább egy ilyen nyomógomb szokott lenni.

A **reset** típus, az űrlap alaphelyzetbe állítását végzi, vagyis, ha a felhasználó egy ilyen típusú nyomógombra kattint, akkor a már kitöltött értékek törlésre kerülnek, és visszaáll az alapértelmezett tartalom. Egy űrlapon legalább egy ilyen elemet is illik elhelyezni.

A **button** típusú nyomógombnak nincs funkciója, vagyis alaphelyzetben semmi se történik, ha a felhasználó rákattint. Nekünk kell funkciót hozzárendelni **JavaScript** kód segítségével. Ehhez rendszerint az `<input>` elem **onClick** attribútumát használják. Az **onClick** értékeként egy **JavaScript** eljárás nevét adhatjuk meg, ami bármilyen tevékenységet elvégezhet. Többnyire adatellenőrzésre használják, hogy minden elem ki van-e töltve és a kitöltött értékek megfelelnek-e az elvárásoknak.

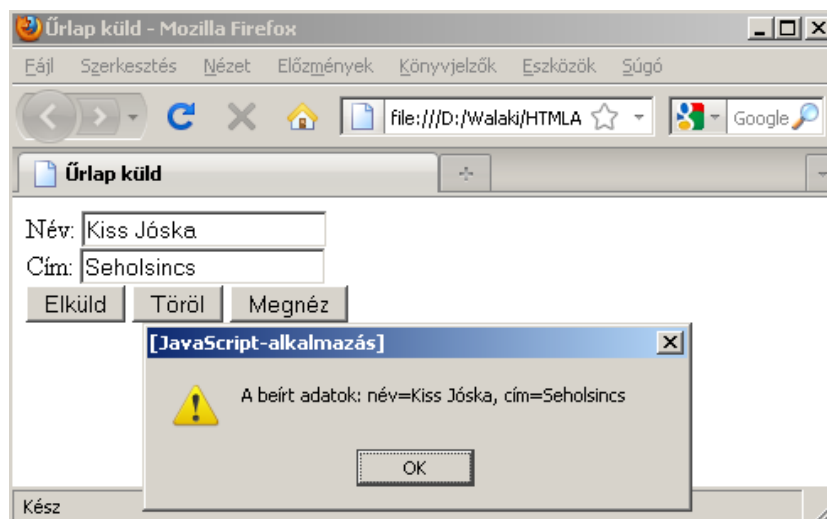
A következő példa mindhárom nyomógombtípus használatát és működését bemutatja:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<html lang="hu">
<head>
  <title>Űrlap küld</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-2">
  <script type="text/JavaScript">
    function megnez() {
      alert('A beírt adatok: név=' + document.Adatok.Nev.value +
        ', cím=' + document.Adatok.Cim.value);
    }
  </script>
</head>
<body>
<form name="Adatok" method="get" action="urlap2.html">
  Név: <input type="text" name="Nev" id="Nev" /><br>
  Cím: <input type="text" name="Cim" id="Cim" /><br>
  <input type="submit" value="Elküld" />
  <input type="reset" value="Töröl" />
  <input type="button" value="Megnéz" onClick="megnez()" />
</form>
</body>
</html>

```

63. ábra Nyomógombok (forrás)



64. ábra Nyomógombok

A példában, ha a felhasználó kitölti a két szöveges mezőt, és rákattint a *Megnéz* gombra, megjelenik egy külön ablakban, hogy milyen értékeket írt be. Ezt az ablakot a **JavaScript** kód `alert` utasítása végzi el, amelyikben az *Adatok* nevű űrlap *Nev* és *Cim* nevű mezőinek értékét írja ki.

Ha a felhasználó a *Töröl* gombra kattint, akkor a beírt értékek törlődnek.

Az *Elküld* gomb hatására pedig a beírt értékek `get` metódus alkalmazásával továbbküldésre kerülnek az **urlap2.html** oldalnak.

### 25.5.3. Rádió gombok

A rádiógombok több elem közül egynek a kiválasztására szolgálnak. A felhasználó az egymás mellett, vagy egymás alatt lévő elemek közül bármelyiket ki tudja választani.



Rádiógombok megjelenítése szintén az `<input>` utasítással történik. A **type** paraméternek ebben az esetben **'radio'** értéket kell adni. Azonban a rádiógombok melletti szöveget külön kell kiírni. Egy `<input>` utasítás egyetlen rádiógombot helyez el az űrlapon.

Ha azt szeretnénk, hogy a felhasználó több elem közül választhasson, akkor több `<input>` utasítás kell, azonban a **name** attribútumoknak mindegyik esetben azonosnak kell lennie.

Használható egyéb paraméterek:

- **value** - a rádiógomb értéke, ez kerül elküldésre az űrlap feldolgozásához
- **checked** - ha valamelyik rádiógombot alapértelmezetten már ki szeretnénk választani, akkor a **checked** értékének **'true'**-t kell adni

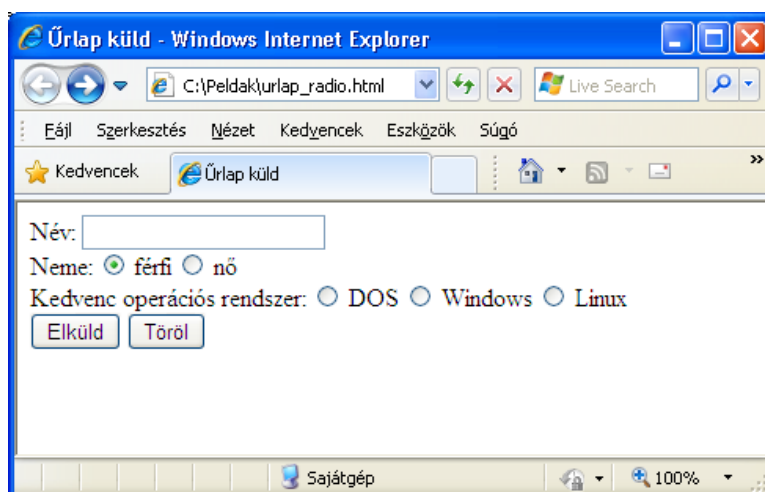
Mivel az összetartozó rádió gombok **name** paramétereinek értéke azonos, ezért a **value** paramétereknek kell különböző értéket adni.

A következő példa bemutatja a rádió gombok helyes használatát. Két csoportban jelenít meg rádió gombokat: *Neme* és *Opr.* Az elsőnél két elem közül lehet választani úgy, hogy az egyik alapértelmezésben ki van választva. A második csoportban már egyetlen elem sincs alapban kiválasztva, a felhasználónak kell választania.

A példa űrlap forrása (`urlap_radio.html`):

```
<form name="Adatok" method="get" action="urlap_radio_fogad.html">
  Név: <input type="text" name="Nev" id="Nev" /><br>
  Neme:
    <input type="radio" name="Neme" value="ferfi" checked="true" /> férfi
    <input type="radio" name="Neme" value="no" /> nő <br />
  Kedvenc operációs rendszer:
    <input type="radio" name="Opr" value="DOS" /> DOS
    <input type="radio" name="Opr" value="Win" /> Windows
    <input type="radio" name="Opr" value="Linux" /> Linux<br />
    <input type="submit" value="Elküld" />
    <input type="reset" value="Töröl" />
</form>
```

65. ábra Rádió gombok, küldő oldal (forrás)



66. ábra Rádió gombok, küldő oldal

Az elküldéskor a **name** paraméterben adott nevű változóban a kiválasztott elem **value** paraméterben megadott értéke kerül továbbításra. Ha a felhasználó egyetlen elemet se választ ki, akkor az adott nevű változó sem kerül továbbításra.

#### 25.5.4. Jelölőnégyzetek (CheckBox-ok)

A jelölőnégyzetek a rádiógombokhoz hasonlóan működnek, azonban az egyes elemek teljesen elkülönülnek egymástól. Egyszerre több elemet is ki lehet jelölni, és a kijelölést is meg lehet szüntetni. A kijelölt elem négyzetében kis pipa jelenik meg.

A jelölőnégyzetek megjelenítése is az `<input>` utasítással történik. A **type** paraméternek ebben az esetben '**checkbox**' értéket kell adni. Az elemek melletti szöveget itt is külön kell kiírni. Egy `<input>` utasítás egyetlen jelölőnégyzetet helyez el az űrlapon.

Ha azt szeretnénk, hogy a felhasználó több elem közül választhasson, akkor több `<input>` utasítás kell.

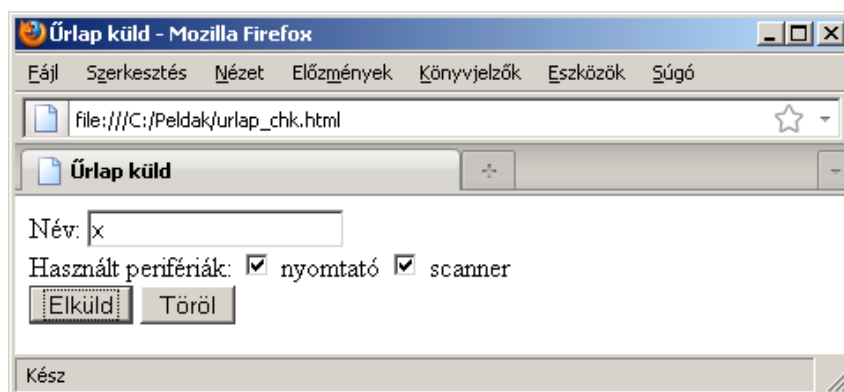
Használható egyéb paraméterek:

- **name** - a változó neve, amiben majd az '*on*' vagy a **value** által meghatározott érték kerül, ha ki van jelölve az elem
- **value** - a jelölőnégyzet értéke, ez kerül elküldésre az űrlap feldolgozásához, ha ki volt jelölve az elem
- **checked** - ha valamelyik elemet alapértelmezetten már ki szeretnénk választani, akkor a **checked** értékének '*true*'-t kell adni

A következő példában két jelölőnégyzet jelenik meg, amelyeket egymástól függetlenül ki lehet jelölni. A **value**-t nem kötelező használni. Ha nincs megadva, akkor az '*on*' érték kerül átadásra kijelölt esetben. Ha nincs kijelölve az elem, akkor a változó sem kerül továbbításra.

```
<form name="Adatok" method="get" action="urlap_chk_fogad.html">
  Név: <input type="text" name="Nev" id="Nev" /><br>
  Használt perifériák:
    <input type="checkbox" name="Nyomtato" value="yes" /> nyomtató
    <input type="checkbox" name="Scanner" /> scanner <br />
    <input type="submit" value="Elküld" />
    <input type="reset" value="Töröl" />
</form>
```

67. ábra Jelölőnégyzetek, küldő oldal (forrás)



68. ábra Jelölőnégyzetek, küldő oldal

## 25.6. Ellenőrző kérdések

A lecke tanulmányozása után próbálja meg önállóan válaszolni a következő kérdésekre!

1. Mit nevezünk űrlapnak?
2. Milyen célokra alkalmazhatunk űrlapokat?
3. Milyen küldési mechanizmusokat használhatunk az űrlapok esetén?
4. Milyen típusú beviteli mezőket lehet használni egy formon?
5. Melyik beviteli elemmel adhat meg a felhasználó többsoros szöveget?
6. Mi az a checkbox?
7. Mit jelent, ha a böngésző címsorában a következő URL jelenik meg?  
`http://www.weblap.hu/index.php?Nev=Kiss+Jozsef&Par=2`
8. Mit jelent űrlap esetén a post?
9. Hogyan lehet meghatározni egy űrlapot? Írd le a HTML kód legfontosabb elemeit!
10. Mit jelent a "küldő" és "fogadó" oldal kifejezés?
11. Milyen TAG-el lehet szöveges beviteli mezőt elhelyezni az űrlapon?
12. Mire szolgál a **value** paraméter szöveges beviteli mezőknél?
13. Hogyan lehet befolyásolni, hogy a szöveges beviteli mezőbe csak 40 karaktert lehessen beírni? Írja le a teljes kódot, ami egy ilyen beviteli mezőt helyez el az űrlapon!
14. Mit csinál a következő HTML kód? Pontosan magyarázza és rajzolja le!  
`<input type="text" size="10" maxlength="8" id="Nyul" value="Fel" name="Fu" />`
15. Mi az azonos és mi a különbség a rádió gombok és a jelölőnégyzet között?
16. Mi a szerepe egy űrlapon a submit típusú nyomógombnak?
17. Keresse meg és javítsa ki a hibát a következő kódrészletben!  
`<from type="submit" name="adat" value="Adat" akcio="feld.html">`  
Név: `<input type="teszt" value="name">`  
`<input type="summit">Elküld</input>`  
`</from>`
18. Mire kell figyelni, ha rádió gombokat helyezünk el az űrlapon?
19. Hogyan kell azt megoldani, hogy az űrlapon 5 rádió gomb legyen, de az első 3 és a másik 2 egymástól függetlenül, egymást kiváltó gombokként működjenek? Írja le a HTML kódot!
20. Minek hatására kerül továbbküldésre a felhasználó által beírt szöveg?
21. Milyen módon lehet feldolgozni a kapott adatokat?
22. Mire van szükség ahhoz, hogy a post metódussal küldött adatokat fel lehessen dolgozni?

## 26. Űrlapok 2

### 26.1. Az űrlap elemei

Egy űrlapot a `<form>...</form>` utasítások között lehet meghatározni. A közbenső részben sokféle HTML elem lehet, de van néhány olyan, amelyeknek csak az űrlapokon belül van jelentősége. Ezek az elemek a következők:

- `<input ... />`  
Segítségével egysoros szöveges beviteli mezőt, rádió gombokat, checkbox-okat, nyomógombokat lehet definiálni.

- `<button>...</button>`  
Nyomógombok meghatározására használható.
- `<select>...</select>`  
Legördülő-, illetve fix méretű választó menü létrehozására használható.
- `<textarea>...</textarea>`  
Többsoros szöveges beviteli mezőt lehet létrehozni vele.
- `<label>...</label>`  
A különböző beviteli elemekhez lehet címkét hozzárendelni segítségével.
- `<fieldset>...</fieldset>`  
Az űrlapon elhelyezett beviteli elemeket lehet csoportosítani segítségével úgy, hogy a `<fieldset>`-en belüli elemeket egy vonallal keríti körbe.
- `<legend>...</legend>`  
A `<fieldset>`-el meghatározott területnek lehet nevet adni.
- `<isindex ... />`  
Egysoros beviteli mezőt jelenít meg. Ma már nem támogatott!

## 26.2. A `button` elem

A `<button>` elemmel ugyanúgy nyomógombokat lehet elhelyezni az űrlapon, mint az `<input>` elem *submit*, *reset* illetve *button* típusával.

Felmerülhet a kérdés, hogy miért van külön elem a nyomógombokhoz, hiszen mindent meg lehet adni az `<input>` elemmel is. A `<button>` újabb elem, csak a HTML 4.0-ban jelent meg, és mivel van külön záró TAG-je is, ezért rugalmasabb, és akár képet is könnyedén el lehet helyezni a nyomógombon. A nyomógombon megjelenő szöveget illetve képet a `<button>` TAG-ek közé kell elhelyezni. Itt további HTML elemek is szerepelhetnek.

## 26.3. A `select` elem

A `<select>` utasítással listadobozt jeleníthetünk meg, amelyből beállítástól függően, egy vagy több elemet jelölhetünk ki. A lehetséges paraméterei:

- **name** - nevet rendel a listadobozhoz, ilyen nevű változóban kerül továbbküldésre a kiválasztott elem azonosítója
- **size** - az egyszerre megjelenő elemek számát szabályozhatjuk, ha nem adjuk meg, csak 1 elem látszik, a többi ilyenkor csak a legördülő menüből érthető el
- **multiple** - ha megadjuk, akkor a felhasználó egyszerre több elemet is kiválaszthat, mivel logikai típusú paraméter, így a *true* értéket adhatjuk, ha szükséges

A `<select>` TAG-ek között további HTML elemeket kell, illetve lehet használni. Ezek az elemek a következők:

- `<option>...</option>` - a lista elemeit kell az `<option>` TAG-ek között meghatározni, tehát ahány elemet meg szeretnénk jeleníteni, annyi `<option>`-re van szükség
- `<optgroup> ...</optgroup>` - az `<option>`-el meghatározott elemeket lehet vele csoportokba szervezni

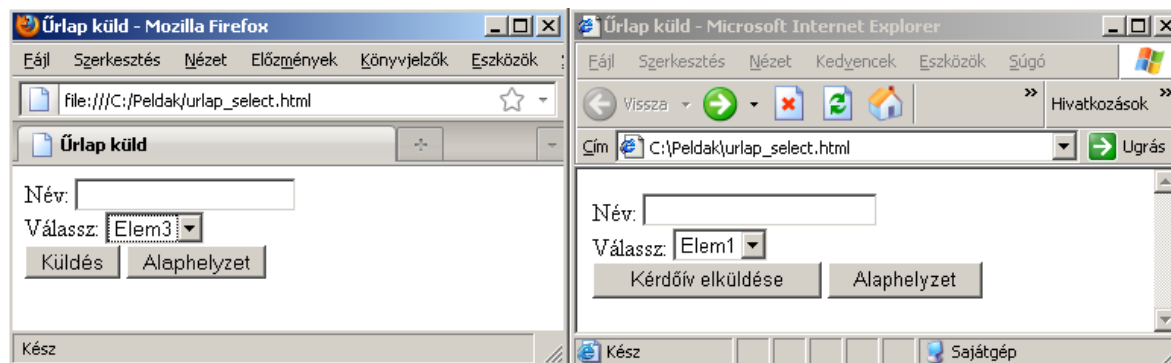
Egyszerű példa az alkalmazásra, ahol egy legördülő listadoboz jelenik meg, három kiválasztható elemmel:

```

<form name="Adatok" method="get" action="urlap_fogad.html">
  Név: <input type="text" name="Nev" id="Nev" /><br>
  Válassz:
  <select name="Lista">
    <option>Elem1</option>
    <option>Elem2</option>
    <option>Elem3</option>
  </select><br />
  <input type="submit" /> <input type="reset" />
</form>

```

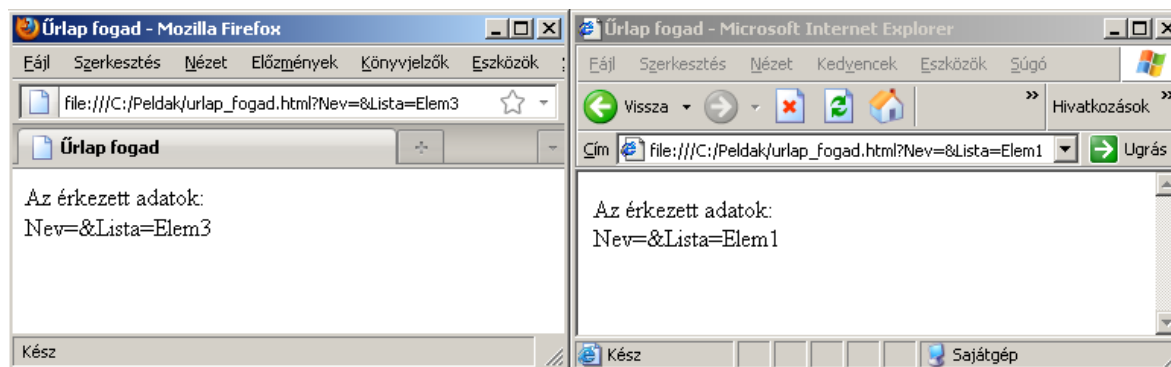
69. ábra A select elem használata (forrás)



70. ábra A select elem használata

Ebben a példában elmaradt a nyomógombok esetén a **value** megadása. Ilyenkor a böngészők helyettesítik be a nyomógombok szövegét. Amint látszik is, a **Firefox** és az **IE6** is más szöveget rendel a **submit** funkcióhoz.

A fogadó oldalon pedig az látszik, hogy a **<select>** **name** paraméterében meghatározott változóban kerül továbbításra a kiválasztott elem neve:



71. ábra A select elem használata, fogadó oldal

### 26.3.1. Az option lehetőségei

A lista elemeit az **<option>** TAG-ek között kell megadni, azonban további paraméterei is lehetnek az **<option>**-nek. A legfontosabbak:

- **selected** - ha értéke **true**, akkor az adott elem alapesetben ki van választva, ha nincs megadva, akkor a legelső elem lesz aktív
- **value** - ha megadásra kerül, akkor ez lesz az adott elem alapértelmezett értéke, felülírva az **<option>** TAG-ek közötti értéket

- **label** - akkor lehet érdekes, ha az adott elem neve hosszú, ilyenkor itt egy rövidebb azonosítót is meg lehet adni, és a továbbküldéskor az itt megadott név lesz a változó tartalma

A HTML 4.01 szabvány szerint legalábbis így kellene működniük. **FireFox és IE6 alatt azonban nem így működik!** Mindkét esetben a **value** határozza meg, hogy milyen érték kerüljön továbbításra, ha az elem kiválasztásra kerül. A **label** paramétert egyik böngésző sem veszi figyelembe. Ezzel szemben az IE8 és a Chrome is az előírásoknak megfelelően kezeli ezeket a paramétereket.

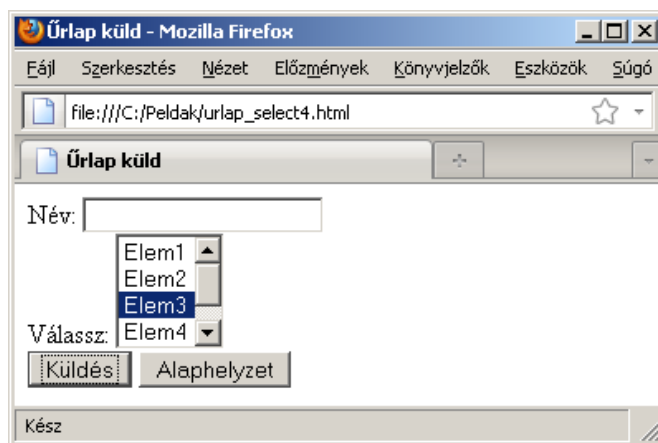
### 26.3.2. Fix méretű lista

A fixméretű lista egyszerűen a `<select>` **size** paraméterének megadásával állítható elő. Ha itt 1-nél nagyobb szám szerepel, akkor a lista ennyi eleme fog egyszerre látszódni.

A következő példában az 5 elem közül csak 4 látszik, de a görgetősávval elérhető a többi is:

```
<form name="Adatok" method="get" action="urlap_fogad.html">
  Név: <input type="text" name="Nev" id="Nev" /><br>
  Válassz:
  <select name="Lista" size="4">
    <option>Elem1</option>
    <option>Elem2</option>
    <option>Elem3</option>
    <option>Elem4</option>
    <option>Elem5</option>
  </select><br />
  <input type="submit" /> <input type="reset" />
</form>
```

72. ábra Fix méretű lista a select elemmel (forrás)



73. ábra Fix méretű lista a select elemmel

## 26.4. A textarea elem

A `<textarea>` segítségével többsoros szöveges beviteli mezőt lehet elhelyezni az űrlapon. A szövegdobozban automatikusan megjelenő szöveget a nyitó és a záró TAG-ek között kell elhelyezni.

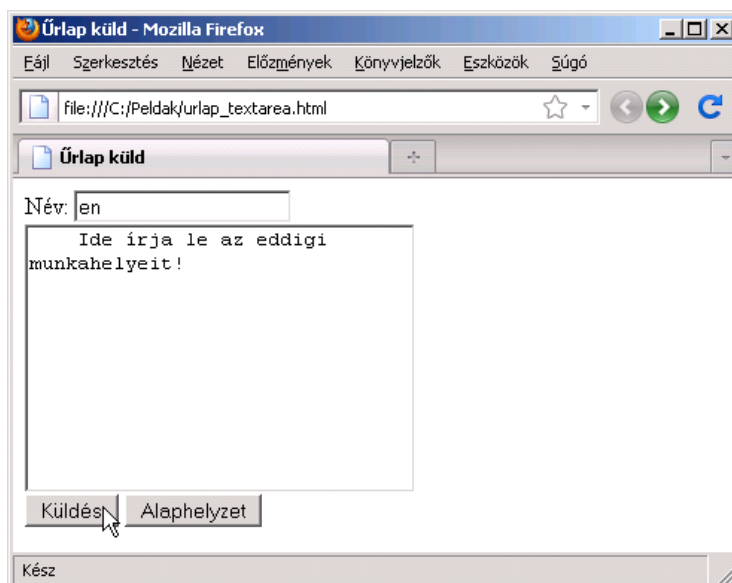
A legfontosabb paraméterei:

- **name** - a változó nevét adhatjuk meg, ilyen nevű változóban kerül továbbításra a szövegdobozba beírt szöveg az űrlap elküldésekor
- **rows** - a sorok számát határozzuk meg, ha nem adjuk meg, akkor a böngészők önállóan választanak egy méretet, tipikusan 5-öt
- **cols** - az elem szélességét határozhatjuk meg átlagos karakter szélességben, ezt sem kötelező megadni, és ilyenkor is a böngészők határozzák meg a méretet

A következő példa egy 30 karakter széles és 10 soros beviteli mezőt helyez el az űrlapon:

```
<form name="Adatok" method="get" action="urlap_fogad.html">
  Név: <input type="text" name="Nev" id="Nev" /><br>
  <textarea name="Munkahelyek" rows="10" cols="30">
    Ide írja le az eddigi munkahelyeit!
  </textarea><br />
  <input type="submit" /> <input type="reset" />
</form>
```

74. ábra A textarea elem használata (forrás)



75. ábra A textarea elem használata

A példából az látszik, hogy a `<textarea>` elemek közötti szöveg a szóközökkel együtt ugyanúgy jelenik meg, vagyis lehetőséget biztosít az előzetes formázásra is, úgy, mintha a `<pre>` elemet használnánk.

## 26.5. Űrlap elemek elhelyezése

Az űrlap elemek ugyanolyan HTML elemek, mint minden eddig tanult más elem is. Külön formázás nélkül, pontosan oda kerül elhelyezésre az oldalon, ahol a kódban szerepel. Ráadásul ezeknek az elemeknek ritka kivételtől eltekintve nincs olyan paraméterük, amivel az elhelyezkedést befolyásolni lehetne. Ezért más utat kell keresni az űrlap elemeinek elhelyezésére. Ilyen lehetőségek:

- `<div>` elem **align** paraméterének segítségével, ami már szabvány szerint nem támogatott, de a Transitional DTD-vel még használható

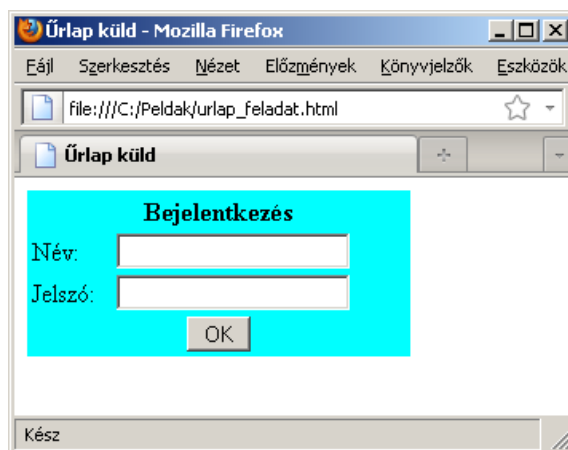
- `<p>` és `&nbsp;` (nem törhető szóköz) segítségével
- táblázatokkal
- CSS segítségével

Összetettebb űrlapok esetén a legegyszerűbb, ha az űrlap elemeit egy táblázat soraiban és celláiban helyezzük el. Így ugyanis minden elem elhelyezkedését külön lehet szabályozni, és az egyéb formázások (pl. háttérszín) is sokkal egyszerűbben megvalósítható.

Erre mutat egy példát a következő kódrészlet, ami egy 250 pixel széles, kék színű területen helyez el elemeket, rendezett formában:

```
<form name="Adatok" method="get" action="urlap_fogad.html">
  <table width="240px" bgcolor="cyan" border="0" >
    <tr>
      <td align="center" colspan="2"><b>Bejelentkezés</b></td>
    </tr>
    <tr>
      <td>Név:</td>
      <td><input type="text" name="Nev" id="Nev" /></td>
    </tr>
    <tr>
      <td>Jelszó:</td>
      <td><input type="password" name="Jelszo" id="Jelszo" /></td>
    </tr>
    <tr>
      <td align="center" colspan="2"><input type="submit" value="OK" /></td>
    </tr>
  </table>
</form>
```

76. ábra Űrlap elemek elhelyezése táblázattal (forrás)



77. ábra Űrlap elemek elhelyezése táblázattal

Ezen az űrlapon nem látszik, hogy táblázat alapú, mivel a keret ki van kapcsolva a `<table>` elemben (`border="0"`). Ha azonban ide 1-et írunk, már rögtön látható, hogy hogyan is épül fel az oldal.

A táblázatok alkalmazásánál azonban figyeljünk arra, hogy ne keveredjenek a táblázathoz tartozó és az űrlaphoz tartozó nyitó és záró elemek. **Mindig tartsuk be az XHTML előírását, vagyis az egymásba ágyazott elemek esetén mindig megfelelő sorrendben zárjuk az elemeket!**



Például helyesek a következő megoldások:

- `<form ...> ... <input ...> ... <table ...> ... </table> ... </form>`
- `<form ...> ... <table ...> ... </table> ... <table ...> ... </table> </form>`
- `<table ...> ... <tr><td><form ...> ... </form></td></tr> ... </table>`

## 26.6. Ellenőrző kérdések

A lecke tanulmányozása után próbálja meg önállóan válaszolni a következő kérdésekre!

1. Milyen elemek lehetnek egy űrlapon?
2. A HTML kódból honnan derül ki, hogy hol kezdődik és hol végződik egy űrlap?
3. Hány darab űrlap lehet egy HTML oldalon?
4. Mely űrlap elemekhez tartozik felirat?
5. Hogyan lehet azokhoz az elemekhez felíratot rendelni, amelyekhez alapesetben nem tartozik?
6. Mi a különbség a `<button>` és az `<input>` elemek által meghatározott nyomógombok között?
7. Adott a következő űrlap:

Készítse el az űrlapot megjelenítő kódot! Figyeljen a formázásokra is! Lehetőleg NE alkalmazzon táblázatot!

8. Hogyan lehet többszintű menüszerkezetet létrehozni egy űrlapon?
9. Értelmezze a következő kódrészletet (írja, hogy hogyan jelenik meg, és mi történik a továbbküldéskor):

```
<label for="">Kedvenc sport:</label>
<select name="sport" id="ksport">
<option>Futball</option>
<option>Asztalitenisz</option>
<option>Sakk</option>
<option>Kézilabda</option>
<option>Starcraft</option>
<option selected="true">Válasszon!</option>
</select>
```

10. Többsoros bevitelei mező esetén hogyan kell a kódban elhelyezni az elemben megjelenő alapértelmezett szöveget?

11. Milyen fontosabb paraméterei vannak a `<textarea>` elemnek?
12. Nézzen utána a HTML 4.01 szabványban, hogy milyen más paraméterei lehetnek az `<option>` elemnek?

## 27. Teszt

Komplex teszt végrehajtása a Moodle e-learning keretrendszerben, automatikus értékeléssel.

## 28. Project feladat

A feladat célja, olyan komplett, több oldalból álló, képeket és egyéb multimédiás elemeket is tartalmazó weblap megtervezése és elkészítése, amelyben alkalmazásra kerülnek a tananyagban tanult elméleti és gyakorlati ismeretek.

A feladatnak öt fő része van:

1. Témaválasztás – ki kell találni, hogy a weblap milyen témakörrel kapcsolatos
2. Információgyűjtés – az oldal tartalmi elemeinek összeállítása
3. Tervezés – az oldal kinézetének megtervezése
4. Elkészítés – az oldalak forrásának előállítása, kódolás
5. Beüzemelés – az oldalak elhelyezése a tárhelyen

### 28.1. Témaválasztás

Ajánlott olyan témát választani, ami valahogy kapcsolódik érdeklődési körünkhöz, esetleg valamelyik hobbinkhoz.

A kiválasztás során szem előtt kell tartani azt is, hogy a végeredménynek több oldalból kell állnia, amelyek megfelelő mennyiségű képet, és néhány egyéb multimédiás elemet is tartalmaznak.

Arra is figyelni kell, hogy az oldalaknak legyen tartalma is, vagyis szöveges információkkal is ki kell azokat tölteni.

Olyan témát célszerű tehát választani, amelyhez több oldalnyi szöveges információt is lehet kapcsolni, valamint több kép illetve néhány egyéb multimédiás elem is kapcsolódik hozzá.

### 28.2. Információgyűjtés

Miután sikerült megtalálni a megfelelő témát, jöhet a tartalom összeállítása. Ajánlott papíron is rögzíteni, hogy milyen adatok lesznek majd a weblapokon. Már a kezdeti fázisban érdemes azon is elgondolkodni, hogy az egyes oldalak milyen kapcsolatban vannak egymással, van-e olyan oldal, amelyik egy másik oldal aloldalának tekinthető. Ezt is rögzítsük papíron!

Készítsünk akár egyszerű szöveges állományokat is, amelyekben elhelyezzük a szöveges tartalmakat.

Érdemes kihasználni a könyvtárak adta lehetőséget, vagyis az állományokat külön könyvtárakba csoportosítsuk!

Keressünk képeket! Ha nincs megfelelő, elképzelhető, hogy magunknak kell néhányat elkészíteni.

A folyamat során eljuthatunk odáig is, hogy rájövünk, nem jó témát választottunk. Ha úgy érezzük, hogy nem fogjuk tudni az oldalakat megfelelő tartalommal is kitölteni, inkább válasszunk másik témát.

### **28.3. Tervezés**

A tervezés során az oldal megjelenésén van a hangsúly. Ez a feladat egy kicsit többet igényel. Kreativitás, és sajnos gyakorlat is szükséges hozzá. Azonban mindent el kell kezdeni valahol. Nem kell megijedni a feladattól!

Először érdemes szétnézni az internetes oldalak között, és keresni olyan oldalakat, amelyek kinézete tetszik. Ezeknek a címeit érdemes külön gyűjteni is. Nagyon sokat lehet tanulni csak abból, ha csak az oldalak kinézetét vizsgáljuk szakmai szemmel.

Kezdő szinten már a tervezés folyamán érdemes eldönteni, hogy milyen módon fog felépülni az oldal szerkezete. Kell-e külön fejléc, lábléc, menü illetve bal vagy jobboldali terület. Ezekről volt szó a keretes oldalakkal kapcsolatban, illetve a táblázatoknál is.

Kezdő szinten ott van a probléma, hogy lehet nekünk professzionális tervünk, ha nem fogjuk tudni megvalósítani. Ezért sajnos előfordulhat, hogy a végleges oldal akár több dologban is el fog térni a tervtől. Ez gyakran előfordul a profikkal is!

Ez az a téma, ami túlmutat a jelenlegi anyagunkon. A megfelelő terv elkészítésekor ugyanis tisztában kellene lenni a színekkel és alkalmazásukkal, tipográfiával, és még sok más dologgal is. Ez már egy külön szakma: web designer (web tervező). Most ne várjunk túl sokat magunktól.

Mindenesetre az oldal színeit ki kell választani, meg kell találni a témához, és ízlésünkhöz legjobban igazodó színválasztékot. Ajánlott több változatot készíteni, és mások véleményét is kikérni. Profik is így dolgoznak. Ilyenkor jön jól, ha otthon vagyunk a képszerkesztésben, vagy jól rajzolunk. Igazából a tervezés grafikus feladat.

Az oldal kinézetének kialakításánál oda kell figyelni a háttérszín és a szöveg színének a viszonyára, vagyis, hogy jól olvasható-e a tartalom. A szöveg igazítása is fontos. A képek és a szöveg arányaira is vigyázni kell, a „ne túl sok, de ne is túl kevés” elvet kövessük! Minden külön oldal esetén vizsgáljuk meg, hogy a terv megfelelő-e!

Végül gondoljunk bele, hogy akkor most hogyan is lehet ezt elkészíteni, hiszen nekünk kell mégiscsak elkészíteni a kódot. Ilyenkor derülhet ki az, hogy a tervet nem fogjuk tudni jelenlegi tudásunk alapján megvalósítani. Az is elképzelhető, hogy emiatt át kell dolgozni a tervet.

### **28.4. Elkészítés**

Adottak a tartalmi elemek és adott a kinézet, vagyis a megjelenés terve. Most következik az ezt megvalósító HTML kód előállítás.

Első lépésként az oldal szerkezetét kell kialakítani. El kell dönteni, hogy alkalmazni kell-e kereteket, vagy táblázatot. Ha igen, akkor először készítsük el a kiinduló oldal felépítését lényeges tartalmi elemek nélkül. Addig alakítsuk, amíg nem felel meg az elvárásoknak. Ha nem sikerül a tervnek megfelelően alakítani a kódot, lehet, hogy módosítani kell a terven.

Miután adott az oldalak szerkezeti megvalósítása, jöhet a tartalmak elhelyezése az oldalakon. Lépésről lépésre haladjunk, és ne felejtsük el, hogy minden módosítás után ellenőrizzük a megjelenést legalább IE (Internet Explorer) és FF (FireFox) böngészőkben.

Az oldalakhoz tartozó állományokat csoportosítsuk, például a képeket külön könyvtárba, videókat is. Ezzel áttekinthetőbbé tehetők az oldalakhoz tartozó állományok szerkezete. A fájlokra minden esetben relatív útvonallal hivatkozunk!

A kódolás során ugyan bármilyen programot használhatunk, de ha lehet, maradjunk a NotePad++, illetve a PsPad editor programoknál.

## **28.5. Beüzemelés**

Ha úgy érezzük, hogy az elkészült oldalak az elvárásainknak megfelelően, hiba nélkül jelennek meg, következhet a beüzemelés.

Ehhez mindenképpen egy külön tárhelyre lesz szükség, így aki még nem regisztrált magának, tegye meg most.

Az oldalakhoz tartozó állományokat, az eredeti szerkezetükben kell a tárhelyen is elhelyezni. A választott FTP program segítségével töltsük fel a fájlokat a tárhelyünkre, majd ellenőrizzük le több böngészővel is, hogy hibátlanul megjelenik minden oldal. Ha szükséges, javítsuk ki a hibát!

Ezzel kész! Lehet dicsekedni!