1. A WEBBÖNGÉSZŐ

Rendeltetése, funkciói

Alkalmas

- dokumentumok megtekintésére: (X)HTML, XML, CSS, JS, TXT, RTF, PDF
- amit nem képes megjeleníteni, azt a Letöltés mappába másolja
- értemezésére: (X)HTML, CSS
- szkriptnyelvek (JavaScript, JScript), alkalmazások futtatására

Funkciói

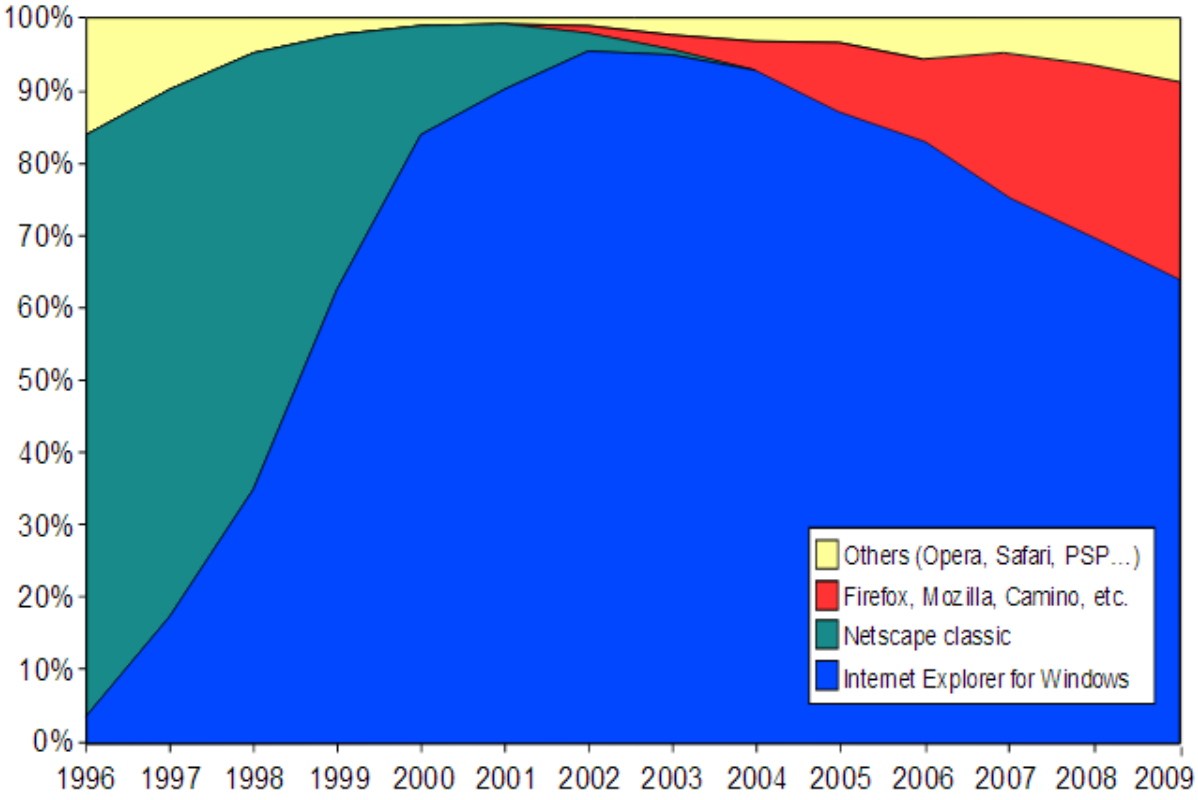
- tabbed browsing
- előzménykezelő
- könyvjelzőkezelő
- letöltéskezelő
- képek megjelenítése: BMP, JPG, GIF, PNG, SVG
- médiumok lejátszása (esetenként beépülőkkel): AVI, MOV, MP4, SWF, FLV, WMV, MP3
- sütik és űrlap-inputok mentése
- forráskód megtekintése
- bővítményekkel funkcionalitása jelentősen kibővíthető (néhány példa):
    - FTP-kapcsolat
    - tesztelés kül. felbontások mellett
    - weblap teljeskörű elemzése
    - szerkesztés és debugging
    - IP- és domain információk
    - SEO-információk
    - screenshots
    - oldalbetöltődési sebesség mérése

Történet

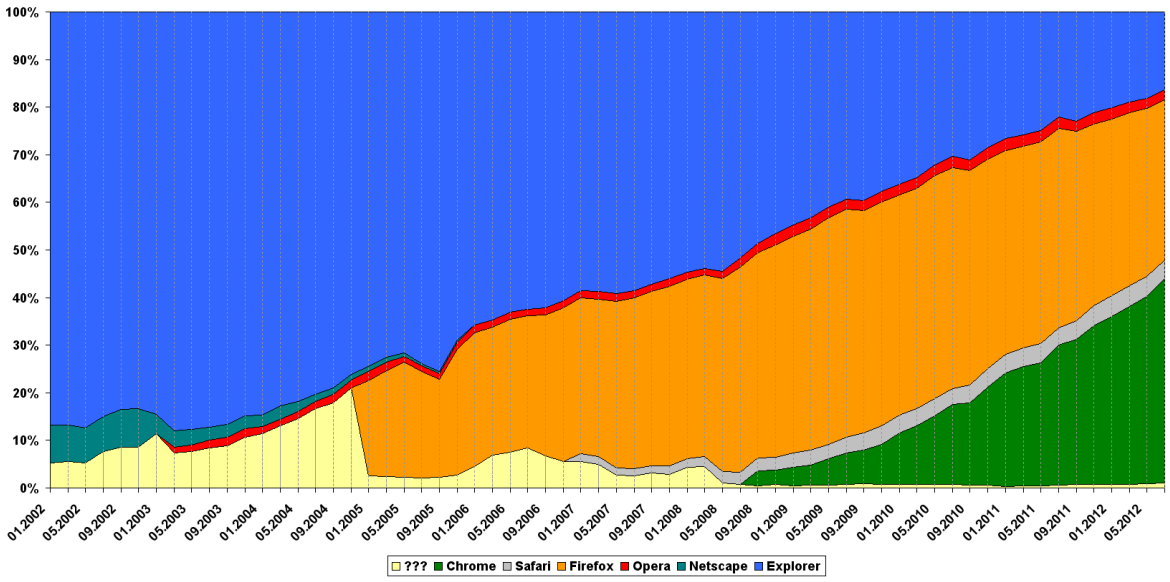
- 1989: WorldWideWeb (későbbi Nexus)
- 1993: NCSA Mosaic
- 1994: Netscape Navigator
- 1995: Microsoft Internet Explorer
- 1995: Opera
- 2002: Mozilla Firefox (korábban Phoenix, Firebird)
- 2003: Apple Safari
- 2008: Google Chrome

Böngészőháborúk

- Microsoft vs. Netscape 1995 – 2000



- Microsoft vs. Mozilla 2004 – 2009



A böngészők billentyűparancsai

Általános

- Backspace: funkciója megegyezik a Vissza gombéval.
- SHIFT-Backspace: funkciója megegyezik az Előre gombéval.
- F5: frissíti (újratölti) az oldalt.
- F3 vagy CTRL-F: megjeleníti a lapon belüli keresőmezőt.
- CTRL-U: a lap forráskódjának megtekintése.
- F11: teljes képernyős (eszköztárak nélküli) üzemmód.
- Tabulátor: a fókusz áthelyezése pl. linkek vagy űrlapelemek között.
- ALT-F4: bezárja a programot.

Tabbed browsing

- CTRL-T: új, üres tabet (kartotékot) nyit meg.
- CTRL-W: bezárja az aktuális tabet.
- Ha a CTRL nyomvatartásával kattintunk meg egy linket, vagy az egérgörgővel (scroll-kerék) kattintunk rá, a cél új tabben nyílik meg.
- A tabeket (a kis piros X-en kívül) úgy is becsukhatjuk, ha a fület akárhol megkattintjuk az egérgörgővel.
- CTRL-Tabulátor: a megnyitott kartotékok közötti váltás. A Firefoxban működik a CTRL és a számok (1-9) kombinációja is. A SHIFT-CTRL-Tabulátor visszafelé lapoz a megnyitott dokumentumok között.

Betűméret

- CTRL + ill. –: betűméretet növelése ill. csökkentése. Általában a böngészők menüsorában is megtalálható. A betűkkel együtt arányosan megnövel mindent: div-konténereket, képeket, háttérképeket, videókat.
- CTRL-0: visszaállítja a 100%-os nézetet. (Explorer: CTRL+alfanumerikus 0)

Kedvencek

- A CTRL-D-vel az aktuális oldalt könyvjelzővel (Explorer: "kedvenc") láthatjuk el.
- A CTRL-B (Bookmarks) megjeleníti a könyvjelzőket (kedvenceket).
- A CTRL-H (History) megjeleníti az előzményeket.

A böngészők helyi menüje

Az ún. helyi menü (kontextusmenü) az egér jobb gombjával nyitható meg, és más-más parancsok szerepelhetnek benne attól függően, hogy hova kattintunk. Ha egy képre kattintunk a jobb gombbal, olyan parancsokat fogunk látni, mint pl. "Képadatok megjelenítése", "Kép mentése", míg ha egy Flash-animációra kattintunk jobb gombbal: "Play", "Rewind", stb. A helyi menü a billentyűzettel is elérhető. Az egyes böngészők helyi menüje igen különböző, és szerepelhet benne bővítmények parancsa is.

További jegyzetek letöltéséhez látogasson ide: timetodesign.hu/tananyag.html

2. HTML

A HTML az internet uralkodó nyelvjárása, és talán a világ legfontosabb nyelve. A HTML ismerete nélkül semmilyen magasabb webtechnológiát nem értenénk meg. Tim Berners-Lee brit informatikus alkotta meg 1989-ben, hogy elősegítse a CERN magfizikai kutatóintézet részlegei közötti információcserét. Neki köszönhető a világ első böngésző-szoftvere (amelyet WorldWideWebnek keresztelt el) és az első webszerver is. Ő alapította meg 1994-ben Massachusettsben a W3C konzorciumot (World Wide Web Consortium), amely a világháló technológiai szabványait hivatott előírni, és azóta is fejleszti a HTML-t, a CSS-t és más technológiákat.

A HTML (Hypertext Markup Language, magyarul: "hiperszöveg-kijelölőnyelv") egy SGML-alapú kijelölőnyelv, amely szövegek, képek és hiperlinkek struktúrálását szolgálja. A HTML leíró nyelv, és nem programnyelv. Míg egy programnyelvben össze lehet adni 2 db számot, addig egy leíró nyelvben ezt nem tehetjük meg. Éppen ezért a HTML-megjelenítők nagyon toleránsak. Míg egy program egy pontosvessző hiánya miatt is működésképtelenné válhat, addig a böngészők csak megjelenítenek valamit, ha nem is egészen helyesen. A HTML teljesen platformfüggetlen, ami azt jelenti, hogy használata nincs programhoz kötve, előállítására bármelyik szövegszerkesztő program alkalmas. A HTML-dokumentum tartalmazhat egyszerű szöveget, beágyazott képeket és egyéb médiumokat (video, animáció, hang, stb). A böngészők által megjelenített tartalmakon kívül olyan metainformációkat is hordozhat, mint az oldal nyelve, rövid ismertetője, a kulcsszavak, a cég / szervezet földrajzi helye. A 4.01-es verzió évtizedes uralkodása után lassan a HTML 5-ös veszi át a jólstrukturált weblapok formátumának szerepét. Ezekkel párhuzamosan létezik még az Extensible Hypertext Markup Language (XHTML), amelyben az XML nyelvhez hasonlóan magunk is deklarálhatunk új tageket.

Történet

- 1992: HTML verziószám nélküli ősverzió. Kizárólag szöveget jelenített meg.
- 1993: Szövegkiemelés, képintegráció.
- 1995: HTML 2.0 - Űrlap bevezetésével.
- 1997: HTML 3.2 - Táblázatok, appletek beillesztése, kép körüli szöveg.
- 1997: HTML 4.0 - Stíluslapok, szkriptek és keretek (frame-ek). 3 szubverzió: Strict, Transitional, Frameset.
- 1999: HTML 4.01 - Kisebb korrekciók.
- 2000: XHTML 1.0 - A HTML újradefiniálása XML-ben (saját elemekkel lehet bővíteni).
- 2001: XHTML 1.1 - Szigorú változat (kevésbé terjedt el).
- 2006: XHTML 2.0 - Új elemeket vezetett be. A W3C a HTML5 javára leállította a fejlesztését.
- 2009: HTML5 - A HTML 4.01 és az XHTML 1.0 alapján új szókinccsel jött létre. A web 2.0 elvárásainak próbál eleget tenni. Sok funkciója korábban csak magasabb technológiákkal, pl. JavaScripttel volt lehetséges. JavaScript API.

Mivel a HTML platformfüggetlen, olyan egyszerű szöveges szerkesztőben is lehet szerkeszteni, mint a Windows Jegyzettömb (Notepad). A kód szerkesztése közben nem kell figyelnünk a szóközökre, tabulátorokra és ENTER-ekre (gyűjtőnéven whitespaces), valamint a kis- és nagy betűs írásmódra, mert a HTML szabvány ezeket figyelmen kívül hagyja.

Dokumentumtípusok

A HTML 4. verziójának megalkotásakor figyelembe kellett venni, hogy nagyon sok meglévő HTML-dokumentum tartalmaz olyan elemeket és attribútumokat, amelyek a prezentációt befolyásolják (ezeket nem akarták használhatatlanná tenni). Az eredmény a HTML három variánsa lett:

1. Strict: Ez a dokumentumtípus az elemek és attribútumok kemény magját foglalja össze. Hiányoznak belőle a prezentációs elemek, mint a <FONT>, A <CENTER>, és az <U> tagek, valamint a BGCOLOR="", az ALIGN="" és a TARGET="" attribútumok. A Strict dokumentumokban ezek szerepét a külső stíluslapok veszik át. A body, form, blockquote és a noscript elemeken belüli szövegeknek és a blokkot nem képző elemeknek olyan konténer-elemekben van a helyük mint a <P> bekezdés.

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

2. Transitional: Ez a dokumentumtípus megengedi a régebbi elemeket és attribútumokat, amelyek fizikai szövegkijelölést tesznek lehetővé. Ezáltal olyan szerzők is tudnak szabványos HTML-t írni, akik még nem választják szét a tartalmat és a designt.

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

3. Frameset: Ez a dokumentumtípus a Transitional összes elemén kívül tartalmazza a frameset-ek, azaz a keretek készítéséhez szükséges elemeket is.

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

Felépítés

A HTML-dokumentum két fundamentális része:

1. HTML-fejléc (HEAD), amely műszaki vagy dokumentációs információkat tartalmaz; ezek a title (lapcím) kivételével a browser területén nem jelennek meg.
2. HTML-test (BODY), amely a hasznos tartalmakat hordozza.

Lássuk, hogy néz ki a kód:

```
<HTML>
  <HEAD>
    <TITLE>Példaoldal</TITLE>
    <META NAME="DESCRIPTION" CONTENT="Példa a HTML-oldal alapfelszerelésére." />
    <!-- esetleg további információk -->
  </HEAD>
  <BODY>
    <P>A weboldal tartalma.</P>
  </BODY>
</HTML>
```

1. A HTML-fejléc

A kötelező title (oldalcím a böngésző a legfelső sávjában) elemen kívül 6 elem szerepelhet a fejlécben:

1. meta: a meta-tag attribútumaival technikai és dokumentációs adatokat lehet megadni:
    a. leírás (description)
    b. kulcsszavak (keywords)
    c. szerző (author)
    d. robots (utasítások keresőmotoroknak)
    e. tartalomtípus és karakterkódolás (Content-Type)
    f. továbbirányítás (refresh)
    g. geo-tag (földrajzi koordináták)
    h. sok egyéb metataget is bevezettek
2. base: bázis-URL-t vagy egy bázisframe-et ad meg
3. link: logikai kapcsolatok más erőforrásokkal (leggyakrabban külső stíluslapokkal)
4. style: stílusinformációkat tartalmaz (leggyakrabban CSS-stílusdeklarációkat)
5. script: vmilyen scriptnyelven íródott kódot ágyaz be (leggyakrabban JavaScript kódot)
6. object: külső adatot hív meg

Példa az egyik legfontosabb meta-tagre, a descriptionre (rövid oldalleírás):

```
<META NAME="DESCRIPTION" CONTENT="Bigyók és gizmók verhetetlen áron.">
```

2. A HTML-test

Szintaxis

A HTML-dokumentumokat kijelölések (ún. markupok) struktúrálják. A legtöbb ilyen kijelölést ún. tagek (ejtsd: tegek) párjai alkotják, tehát egy kezdő és egy záró tag. A kezdő tag elején áll a kisebb-mint karakter, ezt követi az elem osztálya, opcionálisan az elem attribútumai, végül a nagyobb-mint karakter, pl.:

```
<H1 CLASS="WARNING">
```

A browser a kezdő és a záró tag között foglaltakat jeleníti meg. A záró tag egy kisebb-mint karakterből, egy perből (slash), a tagből és a nagyobb-mint karakterből áll. A teljes kijelölés tehát így nézne ki:

```
<H1 CLASS="WARNING">Figyelem!</H1>
```

Elsőrendű címsor következik, mely a warning nevű osztályhoz tartozik, a szövege "Figyelem!". Az oldal megjelenítésekor ez a következőképpen jelenne meg (36px-es fekete Times New Roman):

Figyelem!

Az elemeket egymásba is lehet ágyazni, pl.:

```
<p>Bekezdés, amely egy <strong>kiemelt</strong> szót tartalmaz.</p>
```

A kezdő- és záró tagre szoruló elemeken kívül léteznek tartalom nélküli elemek, pl. a sortörés (break), ld.:

```
<p>Bekezdés, amely egy <strong>kiemelt</strong> szót tartalmaz.</p><br>

<p>Bekezdés, amely egy sortörés után következik.</p>
```

A HTML elsősorban leíró (descriptive), és nem megjenítésorientált (presentational) kijelölőnyelv. A HTML-elemek nem prezentációs utasítások, amelyek közlik a böngészővel, hogy hogyan jelenítse meg a szöveget, hanem sokkal inkább strukturáló kijelölés, amely által a szövegterületek valamilyen jelentést nyernek. Pl. a <H1></H1> első rendű fejezetcímet, a <P></P> bekezdést, az <EM></EM> pedig hangsúlyos szöveget jelöl. Hogy végül is hogyan jeleníti meg a böngésző az elemet, az a böngészőre ill. a megjelenítési környezetre van bízva. Mert bár a HTML-oldalakat általában képernyőkön jelenítik meg, olyan médiumokon is ki lehet őket adni, mint a papír vagy a szövegfelolvasó. A HTML-dokumentumok különböző médiumokon való prezentációjára a CSS stílusleíró nyelvet használják. Eredetileg a HTML-ben is voltak prezentációt szolgáló elemek, mint a <FONT></FONT>, a <B></B> tag-párok vagy a WIDTH="" attribútum, de ezek ma már elavultnak minősülnek és használatuk kerülendő.

Tehát a HTML <H1> tagje azt jelenti, hogy fejezetcím következik, de nem szól arról, hogy azt milyen betűmérettel vagy betűtípussal kell megjeleníteni. Ha ez egy osztályhoz is hozzá van rendelve – mint a példában a "warning" osztály – akkor ha a vonatkozó CSS-dokumentum stílusdeklarációkat tartalmaz erre az osztályra nézve, a "Figyelem!" a megfelelő formázással (pl. piros szín, nagyobb méret) jelenik meg.

A HTML-testben megkülönböztetünk block- és inline-elemeket. A block-elemek a megjelenítéskor saját, oldalszélességű blokkot képeznek, azaz a felettük és alattuk elhelyezett elemektől vizuálisan is elhatárolódnak.

ADDRESS, BLOCKQUOTE, CENTER, DEL, DIR, DIV, DL, FIELDSET, FORM, H1, H2, H3, H4, H5, H6, HR, INS, ISINDEX, MENU, NOFRAMES, NOSCRIPT, OL, P, PRE, TABLE, UL

Az inline-elemek ezzel szemben nem szakítják meg a szövegfolyamot:

A, ABBR, ACRONYM, APPLET, B, BASEFONT, BDO, BIG, BUTTON, CITE, CODE, DEL, DFN, EM, FONT, I, IMG, INS, INPUT, IFRAME, KBD, LABEL, MAP, OBJECT, Q, S, SAMP, SCRIPT, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TEXTAREA, TT, U, VAR

Itt megjegyzendő, hogy némelyik elemet CSS-szabályokkal át lehet konvertálni a másik formátumba. A block elemekhez tartoznak pl. a fejezetcímek:

```
<H1>Első Fejezet</H1>
```

Egy tipikus inline elem pedig a hiperlinkek (hivatkozások) beillesztésére szolgáló <a> tag:

```
A jelentkezési lap <A HREF="http://iskola.hu/jellap.pdf">ezen a
linken</A> tölthető le.
```

A hiperlinkek hivatkozások más erőforrásokra, legtöbbször más HTML-dokumentumokra, amelyek kattintásra elérhetők a böngészőben. Tipikus inline-elemek továbbá a <STRONG> vagy az <EM>, amelyekkel szövegrészeket emelhetünk ki.

3. XML

Az XML (Extensible Markup Language: Bővíthető Leíró Nyelv) egy strukturált, szöveges adatok tárolására használatos leíró nyelv, melyet elsősorban az adatok interneten történő megosztására alkottak meg. A nyelv a W3C (World Wide Web Consortium) szabványa, és az SGML nyelvből származik (Structured Generalized Markup Language: Általános Jelölőnyelv) – csakúgy, mint a weboldalakat leíró HTML, amelynek szintaktikája sokban hasonlít az XML-re. A különbség az, hogy míg a HTML elemeinek készlete előre meghatározott, addig az XML-ben a felhasználó maga hozhat létre új elemeket, tulajdonságokat, és értékeket.

A nyelv számos előnnyel rendelkezik:

- Ember által is érthető módon tárol adatokat.
- Egyszerű, könnyen elkészíthető és hozzáférhető.
- Többféle adatstruktúrát képes ábrázolni.
- Támogatja a Unicode karakterkódolást.
- Széles körben elterjedt, platform-független, bármilyen szövegszerkesztővel létrehozható.
- Tömör, formális, de szigorú szabvány.
- Kis file-méret.

Ez a leíró nyelv tehát lehetővé teszi, hogy bárki egyszerűen, szöveges módon adatokat rögzítsen úgy, hogy azok bármilyen program (a mi esetünkben a Flash/ActionScript) által is értelmezhetők legyenek.

XML-alapú osztálynapló

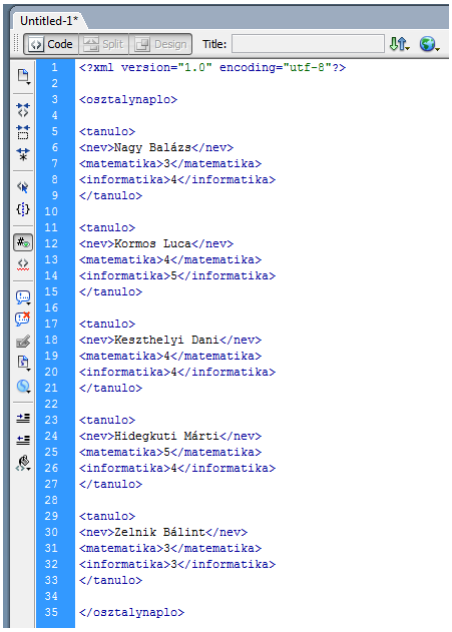
Bármely szövegszerkesztővel indítsunk egy új dokumentumot, és nevezzük el „osztalynaplo.xml"-nek! Az első sor az XML fejléce:

```
<?xml version="1.0" encoding="utf-8"?>
```

Ezzel közöljük a leendő feldolgozó programmal, hogy XML dokumentumról van szó, a szabvány 1. verzióját használjuk, és hogy Unicode karakterkódolással kezelje.

Ezek után szükségünk van egy ún, dokumentumelemre, amely magába foglalja a teljes adatrészt:

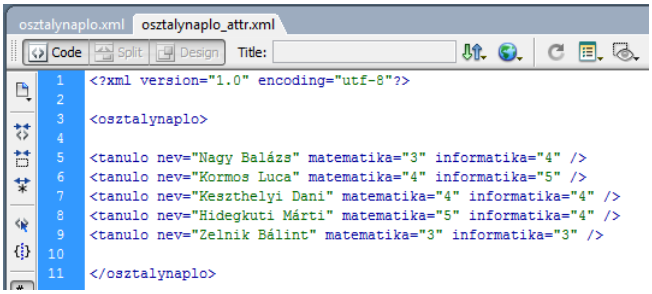
```
<osztalynaplo></osztalynaplo>
```



Csakúgy, mint a HTML-ben, az elemek itt is tagekből épülnek fel. Az adatok nyitó és záró taggel rendelkeznek (ill. előfordulhat, hogy a két tag között nincs érték, ez esetben a nyitó részben egyből le is lehet zárni az elemet).

Ezek után jön az adatrész, amit hasonlóan adunk meg: egy tanulo elem megnyitásával közöljük az új tanuló felvételét, majd minden egyes adatát külön megcímezve új elemekben tároljuk. Amennyiben nincs több adatunk az adott tanulóról, bezárjuk a hozzá tartozó taget.

A HTML nyelvhez hasonlóan itt is lehetnek az elemeknek tulajdonságai (attribútumai). A fenti adatbázist felépíthetnénk így is:



Mentsük el ezt az állományt osztalynaplo_attr.xml néven. Ezzel a módszerrel attribútumokat adunk meg, és azokat látjuk el értékkel. Éppen ezért nincs is szükségünk a nyitó és záró tag közötti helyre, így akár el is lehet hagyni a bezáró részt. A szabvány megköveteli, hogy minden nyitott elemet be kell zárni, de erre van lehetőség a nyitó tagben is: a bezáró reláció előtti / jel jelképezi az elem zárását.

A két írásmódot tetszés szerint lehet alkalmazni, akár kombinálni is. Az adatok mindenképp helyesen tárolódnak el, és mindkét esetben egyértelmű, jól felépített, és értelmezhető struktúrát lehet létrehozni. Különbség csak az adatok feldolgozásánál lesz, elvégre a másképpen eltárolt adatokat másképp is kell kiolvasni. A második módszer annyival jobb, hogy rövidebb (kisebb foglalási méretű) és átláthatóbb.

4. XHTML

A HTML (HyperText Markup Language = hiperszöveg leíró nyelv) szabványt sok kritika érte amiatt, hogy a tartalmi jelölésre szolgáló elemek mellett nagyon sok formai, megjelenítést meghatározó elem is beszűrődött, és így összemossa a tartalmat és a formát.

A HTML 4.0 szabványban ezért sok, a korábbi verziókban használt elemet érvénytelenítettek, és a designra a stíluslapok (CSS) használatát javasolták. A HTML egy sztatikus, előre definiált elemeket és paramétereket tartalmazó jelölő nyelv volt, ezen próbált a W3C az XHTML (eXtensible HTML = kiterjesztett hiperszöveg leíró nyelv) szabvánnyal segíteni.

Az XHTML családba tartozó dokumentumtípusok XML-alapúak, azonban a hagyományos (HTML 4.0-et ismerő) böngészővel is olvashatók maradnak. Úgy is fogalmazhatunk, hogy az XHTML a HTML megfogalmazása XML-ben (a HTML még SGML-ben lett definiálva). Jelentős eltérés nincs a két nyelv között, csak a formai követelmények lettek szigorúbbak.

- Könnyen áttekinthető és szerkeszthető dokumentumok, sztenderd XML-eszközökkel érvényesíthetők (validálhatók).
- Az XHTML dokumentumok a HTML 4.0 szabványt támogató szerkesztőprogramokkal is szerkeszthetők.
- Az XHTML dokumentumban használhatunk olyan alkalmazásokat (appletek, szkriptek), amelyek futtatásához szükséges a HTML vagy az XML dokumentum-objektum-modell (DOM).
- Az XHTML nyelvcsalád fejlődésével az XHTML 1.0 kritériumainak megfelelő dokumentumok egyre inkább együtt tudnak működni egymással a különböző XHTML környezetekben.

A validálás lényegében annyit jelent, hogy ellenőrizzük, hogy a dokumentum tartalma megfelel-e a nyelv szabályainak. Ha igen, akkor a dokumentumra azt mondjuk, hogy érvényes. Maga a W3C a http://validator.w3.org/ címen tartja online validátorát: A szoftver ellenőrzi a világhálón található dokumentum helyességét a benne deklarált dokumentumtípusnak (DOCTYPE) megfelelően. Noha a böngészők toleranciája miatt a hibás dokumentumok is elműködnek (sőt, pl. a Google főoldalán 37 hibát jelez a W3C-validator), a szabványos kód valamennyire a minőségi webdesign ismérve az ágazatban.

HTML / XHTML különbségek

1. Az XHTML-ben az elemek egymásba ágyazásánál ügyelnünk kell a sorrendre, vagyis az elemek nyitó tagjeinek fordított sorrendjében kell elhelyezni a záró tagjeiket. Míg a HTML-ben a

   ```
   <b><u>Vastag és aláhúzott</b></u>
   ```

   kód értelmezése sem okoz különösebb problémát a megjelenítő programoknak, az XHTML dokumentumban ugyanez nem szerepelhet így; helyesen így néz ki:

   ```
   <b><u>Vastag és aláhúzott</u></b>
   ```

2. Minden elemet a <html> elemen belül kell elhelyezni. Minden elemnek lehetnek további beágyazott elemei. Ezek az elemek páronként kerülnek megadásra. Ezáltal kisebb foglalási méretű, gyorsabb böngészőprogramokkal lehet dolgozni.

3. A tageket kis betűkkel kell írnunk. Mivel az XML-szabvány megkülönbözteti a kis- és nagybetűket, a
 és
 tag két különböző dolgot jelöl.

4. Azon elemek végére, amelyeknek a HTML-szabványban nincs záró párjuk (mint az area, base, bgsound, br, col, frame, hr, img, input, isindex, keygen, link, meta, option, param), / karaktert kell tennünk. Pl. `<br>` helyett `<br />`, vagy

   `<img src="kep.jpg">` helyett az `<img src="kep.jpg" />`

   kódot kell írnunk. Ne felejtsünk el szóközt tenni a / jel elé.

5. Az attribútumokat is kis betűvel kell írnunk.

   Rossz: `<table WIDTH="250">`

   Helyes: `<table width="250">`

6. Az attribútumok értékeit "" jelek közé kell zárnunk.

   Rossz: `<table width=250>`

   Helyes: `<table width="250">`

7. Nem megengedett az attribútum-érték párok minimalizálása.

   Rossz: `<input type="radio" checked>`

   Helyes: `<input type="radio" checked="checked" />`

8. A dokumentumon belüli navigáció és elemazonosítás céljából id attribútumot kell használni.

   Rossz: `<img src="kep.jpg" name="kep1" />`

   Helyes: `<img src="kep.jpg" id="kep1" />`

   Ez nem azt jelenti, hogy a name attribútum nem használható! A kettőt kombinálni is lehet:

   `<img src="kep.jpg" name="kep1" id="kep1" />`

9. A HTML-ben a lang attribútum szolgált az adott elem nyelvének megadására. Az XHTML dokumentumokban az xml:lang="érték" elemet is szerepeltetni kell.

   Rossz: `<div lang="hu">Nyelv meghatározása</div>`

   Helyes: `<div lang="hu" xml:lang="hu">Nyelv meghatározása</div>`

10. Kötelező XHTML-elemek

    Minden XHTML dokumentumnak rendelkezni kell DOCTYPE deklarációval. A html, head és body elemeknek szerepelniük kell, a title elemnek pedig a headen belül kell elhelyezkednie. Egy leegyszerűsített XHTML dokumentumsablon tehát a következőképpen néz ki:

    `<!DOCTYPE ide kerül a dokumentumtípus>`

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head><title>A weblap címe</title></head>

<body>Ide kerül a tartalom.</body>

</html>
```

Dokumentumtípusok

A DOCTYPE deklaráció nem része magának az XHTML-dokumentumnak, ezért záró eleme sincs, és kötelező az első sorban szerepeltetni. A DOCTYPE definiálja a dokumentum típusát. Az 1.0-s szabványban három dokumentumtípus létezik:

1. XHTML 1.0 Strict

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

A szigorú variáns a W3C által javasolt és preferált változat. A legtöbb, szövegek formázására szolgáló elem és attribútum hiányzik belőle. Ezek helyett a webszerkesztőknek a stíluslapokat kell használnia. A bodyban csak block-elemek állhatnak. Ez a szigorú változat az újonnan készülő weboldalaknál javasolt, amelyek formázása következetesen CSS-sel történik.

2. XHTML 1.0 Transitional

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

Ezt akkor érdemes használni, amikor hasznosítani szeretnénk a HTML prezentációs lehetőségeit. A Transitional (átmeneti) variáns egy kompromisszum, amit a W3C – széleskörű elterjedtségére való tekintettel – megenged, és online validatorével még szentesít is. Ez a változat megengedi azon elemek és attribútumok használatát, amelyeket elutasítottként (deprecated) bélyegeztek meg, és a Strict változatban már nincsenek meg; ilyenek a font, az align, és a bgcolor. A Transitional változatban szabad a bodyba inline-elemeket vagy tag nélkül álló szöveget tenni.

3. XHTML 1.0 Frameset

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

Ezt akkor használjuk, ha az ablakot több keretre (frame-re) osztjuk fel. Így egy HTML-be (a framesetbe) több kisebb ablak nyílik, amelyekbe más HTML-ek tölthetők be. Egyes esetekben célszerű lehet, de ez már egy kifutó technológia.

5. CSS: EGYMÁSBA ÁGYAZOTT STÍLUSLAPOK

Történet és definíció

1996: CSS1, 1998: CSS2 (csak 2011-ben vált végleges W3C-ajánlássá), 2006 óta: CSS3 fejlesztése

A CSS (Cascading Style Sheets) egy leíró nyelv, mely önmagában nem jelenít meg tartalmat, hanem megformázza egy másik leíró nyelvben (pl. HTML-ben) foglalt tartalmakat. A CSS-t a W3C azért hozta létre, hogy formázási és megjelenítési szempontból kiegészítse a HTML hiányosságait. A CSS elvileg minden böngésző számára egyértelműen meghatározza az elemek küllemét. A CSS egy platform- és program-független nyelv, tehát bármilyen szövegszerkesztővel létre lehet hozni CSS-dokumentumokat.

Ahhoz, hogy megértsük, mit köszönhet a világháló a CSS-nek, végezzünk el egy kísérletet. Nyissuk meg kedvenc böngészőnkben egy korszerű oldalt. Keressük meg a böngészőben a hozzárendelt oldalstílusok lekapcsolásának parancsát. A Firefoxban ez a View menü / Style / No style parancsa. Hölgyeim és uraim, ilyesfélén nézett ki a WWW 1996-ig.

A CSS változatos lehetőségeit hivatott szemléltetni a CSS Zen Garden nevű fórum. A 2003-ban lefektetett HTML-kódot a felhasználók által beküldött stíluslapokkal jeleníti meg. A legjobbak a jobb oldali toplistán aktiválhatók. Ugyanaz a tartalom – teljesen eltérő look & feel, a CSS-designer stílusától függően.

http://www.csszengarden.com/tr/magyar/

Hol?

CSS-utasítások négy helyen szerepelhetnek:

	Hely	*Példa*
1.	A böngésző alapbeállításaiban. Ettől kék a link, és itt van meghatározva, hogy pl. a főcím 36px-es fekete megvastagított Times New Romannel jelenjen meg.	

2.	Külső CSS-dokumentumban.	`p {`
		`color: red;`
		`}`

3.	A HTML head szekciójában egy <STYLE> tagben.	`<head>`
		`...`
		`<style type="text/css">`
		`p {`
		`color: red;`
		`}`
		`</style>`
		`</head>`

4.	A HTML bodyján belül a tagek style attribútumában.	`<p style="color: red">`

Hogyan?

A head szekcióban vagy a külső .css file-ban a következő szelektorok kaphatnak CSS-utasítást:

	Szelektor	*Példa*
1.	Minden elem, csillaggal jelölve.	`* { margin: 0; padding: 0; }`
2.	Azonosító (id), kettőskereszttel jelölve. Bármilyen tag tartozhat hozzá, de laponként egy adott azonosítójú elem csak egyszer szerepelhet.	`#bevezeto { font-weight: bold; }`
3.	Osztály (class), ponttal jelölve. Többféle tag tartozhat ugyanahhoz az osztályhoz.	`.idezet { font-style: italic; }`
4.	HTML-tag.	`p { color:red; }`

Ha külön CSS file-t hozunk létre, és abban tárolunk minden a weboldal által használt formázást, majd a HTML-lapoknak megmutatjuk, hogy ebből a CSS file-ból olvassák ki a rájuk vonatkozó utasításokat, akkor képesek leszünk kizárólag a CSS file-t módosítva a weboldalunk összes (akár több tucat) HTML-dokumentumát megváltoztatni. A tartalom és a forma szigorú szétválasztásának előnye az áttekinthetőség és a rugalmasság. Ezért mi most azt az esetet vizsgáljuk, mikor külön file-ban tároljuk weboldalunk stílusleírásait.

Szintaxis

Az osztály olyan formázásokat gyűjtő csoport, amelyet többféle HTML-taghez is lehet rendelni. Célszerű olyan nevet választani, amely utal arra a tartalomra, amelyre a stílust alkalmazni fogjuk. Pl. ha egy olyan bekezdést akarunk megformázni, amibe egy idézet fog kerülni, akkor nem árt a bekezdéseket formázó utasításokat tartalmazó osztályt "idezet"-nek nevezni. Később újabb bekezdéseket rendelhetünk az idezet osztályhoz (de akár más tageket is). Az osztálynevekben nem használhatók ékezetes és speciális karakterek. Hozzunk létre egy CSS-dokumentumot a New... menüvel és adjunk neki néhány tulajdonság-érték párt (pl. igazítás és betűköz). Végül zárjuk a deklarációs szakaszt egy kapcsos zárójellel:

```
.idezet {
      color: #3300CC;
      text-align: justify;
      letter-spacing: 1px;
}
```

Ezen a példán már látszik a CSS logikája:

```
szelektor {
      tulajdonság: érték;
      tulajdonság: érték;
      tulajdonság: érték mértékegység;
}
```

Minden szabályhoz tartozik egy szelektor és egy deklarációs szakasz. Ez utóbbi kapcsos zárójelek között pontosvesszővel elválasztott deklarációkat tartalmaz. A deklarációk formája a következő: a tulajdonság neve, kettőspont, értéke, esetleg egy mértékegység, végül pontosvessző. Mentsük el a file-t style.css néven.

Hozzunk létre egy HTML-dokumentumot styled.html néven. A stíluslappal való összekötést a HEAD szekcióban lévő <LINK> tag végzi:

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

Ebből a böngészők tudni fogják, hogy létezik egy style.css file, amit figyelembe kell venniük a lap formázásakor. Rendeljük az osztályhoz az egyik bekezdést a class="idezet" attribútummal, és frissítsük a böngészőt F5-tel. Ha mindent jól csináltunk, a bekezdés szövege lila színben fog pompázni.

Tehát összefoglalva a class attribútummal megadtuk a böngészőnek, hogy az idezet nevű osztályban felsorolt tulajdonságokkal kell felruháznia a bekezdést. Mivel a <HEAD> részben már elárultuk, hogy kell itt lennie egy .css-file-nak is, a böngésző végigolvassa azt, megtalálja az idezet osztályt, és annak utasításainak megfelelően formázza meg a bekezdést.

Persze általános HTML-tagekhez is rendelhetünk stílust. Formázzuk meg pl. a címsort:

```
h1 {
      font-family: Cambria, Georgia, Times, serif;
      font-size: 40px;
      font-weight: bold;
}
```

Ezt egyébként le is lehet rövidíteni:

```
h1 {
      font: bold 40px Cambria, Georgia, Times, serif
}
```

… mivel egyértelmű, hogy melyik érték melyik tulajdonságé. Ha csak egy deklarációnk van a szelektorban, az elválasztó művelet (a pontosvessző) is felesleges.

Formázzuk meg a linkeket is, pl.:

```
a {
	color: #FF6600;
	text-decoration: none;
}
```

A linkek rollover állapotát a hover látszólagos kiválasztóval befolyásolhatjuk:

```
a:hover {
	color: #FFCC00;
	text-decoration: underline;
}
```

Adjunk a felsorolásoknak egyéni strigulát:

```
ul {
	list-style-image: url(../img/li_makk.png);
}
```

Tegyük fel, hogy most azt szeretnénk, ha a tartalmunk a mindenkori böngészőablak közepén jelenne meg. Ehhez a body teljes tartalmát egy DIV tagbe kell zárnunk. A DIV olyan konténer, amelyeknek helyzetét és kiterjedését, de számos egyéb paraméterét, és a benne lévő elemeket is lehet befolyásolni CSS-sel. Az imént létrehozott divünket fix szélességűre állítjuk, relatív módon fogjuk pozícionálni, és megmondjuk neki, hogy a mindenkori ablak bal és jobb szélétől mindig azonos távolságot tartson. Mivel ebből laponként csak egy lesz, id-vel fogjuk ellátni, és hogy tudjuk is, mi volt a szándékunk vele, az azonosítója legyen: centirozni.

```
<body>
	<div id="centirozni">
		<h1>Lorem ipsum dolor sit amet</h1>
		<p class="idezet">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
	</div>
</body>
```

A CSS-be pedig valami ilyesmi kell, hogy kerüljön:

```
#centirozni {
	width: 500px;
	position: relative;
	margin-right: auto;
	margin-left: auto;
}
```