

DEBRECENI EGYETEM
INFORMATIKAI KAR
KÖNYVTÁR-INFORMATIKAI TANSZÉK

Weblap-szerkesztés lépésről lépésre

Témavezető:

Dr. Boda István
tanszékvezető egyetemi docens

Készítette:

Naár János
informatikus könyvtáros szak

Debrecen
2008

Tartalomjegyzék

BEVEZETÉS	1
1. A WORD WIDE WEB	3
2. A WEBLAP-SZERKESZTÉS TÖRTÉNETE, JELENTŐSÉGE.....	7
3. A LEÍRÓ JELÖLŐ NYELVEK KIALAKULÁSA	8
4. A WEBLAP-SZERKESZTÉS LEÍRÓ JELÖLŐ NYELVEI ÉS ALKALMAZÁSAI.....	10
4.1. SGML.....	10
4.1.1 Az SGML szabványról	11
4.1.2 DTD – dokumentum típusdefiníció	12
4.1.3 Az SGML felhasználási területei, előnyei és hátrányai.....	15
4.2. HTML	17
4.2.1 A HTML története – a különböző szintek jellemzése.....	17
4.2.2 A HTML elemek szintaxisa	19
4.2.3 A HTML adattípusai.....	20
4.2.4 A HTML dokumentum alapszerkezete, alapvető elemei.....	22
4.2.5 A HTML szövegformázó elemei.....	23
4.2.6 Képek beágyazása a HTML dokumentumba.....	26
4.2.7 Táblázatok a HTML dokumentumban.....	27
4.2.8 HTML keretek – a dokumentumablak felosztása	29
4.2.9 Hiperhivatkozások a HTML dokumentumban	31
4.2.10 Űrlapok a HTML dokumentumban	33
4.2.11 A HTML dokumentumok egyéb elemei.....	36
4.3. XML.....	37
4.3.1. Az XML jellemzői.....	38
4.3.2. Az XML felépítése	39
4.3.3. DTD.....	41
4.3.4. XML Schema definition Language (XSD)	42
4.3.5. DOM – Document Object Modell.....	44
4.3.6. XML és HTML kapcsolata.....	45
4.3.7. Az XML felhasználási területei és lehetőségei.....	46
4.4. XHTML – A HTML MEGÚJULÁSA XML ALAPOKON	48
4.4.1. Az XHTML dokumentum alapszerkezete.....	48
4.4.2. Az XHTML elemei.....	50
4.4.3. Az XHTML és a HTML összehasonlítása	50
4.4.4. Az XHTML oldal hibátlanlágának alapvető kritériumai	52
4.5. DHTML.....	52
4.5.1 JavaScript.....	53
4.5.2 Stíluslapok - CSS	58
5. WEBOLDALTERVEZÉS.....	64
5.1. TARTALOMTERVEZÉS.....	64
5.2. INTERAKTIVITÁS	66
5.3. OLDALTERVEZÉS	67
5.3.1. Oldalszerkezet kialakítása és design.....	68
5.3.2. Webes tipográfia.....	69
5.3.3. A honlap akadálymentesítése.....	70
5.4. A SZERKESZTÉS ÁLTALÁNOS SZEMPONTJAI.....	71
5.5. KÓDOLÁSI SZEMPONTOK	72
6. WEBLAP-SZERKESZTŐ PROGRAMOK.....	74
6.1. NOTEPAD / JEGYZETTÖMB.....	75
6.2. NVU	75

6.3.	MICROSOFT FRONTPAGE	76
6.4.	MACROMEDIA DREAMWEAVER.....	77
7.	WEBOLDALAK SZABVÁNYOSSÁGA, VALIDÁLÁS	80
8.	BROWSEREK	82
	ÖSSZEGZÉS	84
	FOGALOMTÁR	87
	RÖVIDÍTÉSJEGYZÉK.....	89
	FELHASZNÁLT IRODALOM.....	92
	MELLÉKLETEK.....	94

*„A múltban izomerőnket megsokszorozó eszközöket használtunk,
ma pedig olyanokat, amelyekkel az elménket sokszorozzuk meg.”*
(Bill Gates)

BEVEZETÉS

Már óvodás koromban – mikor még írni és olvasni sem tudtam – kapcsolatba kerültem a számítógéppel, és már képes voltam megkeresni és elindítani a kedvenc ügyességi játékprogramokat. Ez az érdeklődés általános iskolás koromban sem lanyhult, és ekkor a számítástechnika szakkör adta azt a többletet, amit igényeltem. Szakközépiskolában ennek köszönhetően természetesen informatikai szakmacsoportos osztályban tanultam, és érettségi utáni továbbtanulási döntésemben is szerepet játszott az informatika iránti érdeklődés. Jelentős mértékben motivált az is, hogy főiskolai hallgatóként 2006-ban a VIII. Pető József Országos Számítástechnikai Emlékversenyen első helyezést értem el. A szakdolgozati témaválasztásom szakterületét illetően egyértelműen befolyásolt az érdeklődés a számítástechnika, az internet és a Web iránt, valamint inspirált a tanulmányaim során megszerzett ismeretanyag.

Dolgozatomban a Word Wide Web-hez kapcsolódó fogalmak tisztázása és a weblapszerkesztés történetének, valamint a leíró jelölő nyelvek kialakulásának rövid ismertetése után elsőként szeretném bemutatni azokat a szabványokat és ajánlásokat, amelyek a digitalizált tartalmak tárolásának, továbbításának és feldolgozásának egységesítésére kerültek kidolgozásra. Be kívánom mutatni az egyes jelölő nyelvek alapvető jellemzőit, elemeit, a jelölés alapelveit, a szintaktikai és szemantikai szabályait, felhasználási területeit, valamint előnyeit és hátrányait más nyelvekkel való összehasonlításban. Azt is szeretném feltárni, hogy a technikai változások és a felhasználói igények milyen hatással voltak a szabványok és a webes tartalmak kezelését és megjelenítését szolgáló szoftverek fejlesztésére. Részletesen kívánom meghatározni a weboldaltervezés elveit, területeit, módszereit és szempontjait. Szeretném kiemelni a weboldalak szabványosságának fontosságát és bemutatni vizsgálati módszereit. Fontosnak tartom, azt is hogy átfogó képet alkossak a böngésző programokról.

A dolgozatomat az előbbieken megfogalmazott céloknak megfelelően nyolc fő részre tagoltam. Az első két részében a webről, annak kialakulásáról és weblapszerkesztés történetéről adok egy áttekintést. A dolgozat legterjedelmesebb negyedik részében a leíró jelölő nyelvek kialakulásának története került bemutatásra. A negyedik fejezetben részletesen

feldolgozom a weblapszerkesztés leíró jelölő nyelveit, és ismertetetem az ezekre vonatkozó szabványokat és ajánlásokat. Ebben a részében összehasonlító elemzés keretében megvizsgálom az egyes nyelvek különbözőségeit, az eltérésekből eredő előnyeit és hátrányait. A nyelvek elemeinek bemutatása során kisebb-nagyobb gyakorlati példákkal illusztráltam az alkalmazás lehetőségeit. A következő két részben a weboldaltervezés és -szerkesztés során követendő elveket, szabályokat és módszereket tárgyalom és a webszerkesztő programokat mutatom be. A dolgozat további részeiben a weboldalak szabványosságának kritériumait és értékelési módszereit fogalmazom meg. Végül a böngészőprogramok jellemzésével, használati arányainak bemutatásával zárul a főrészt. Az összefoglaló megállapítások után fogalomtár és rövidítésjegyzék segíti az olyan szakkifejezések és rövidítések megértését, amelyek a téma feldolgozása során nem kerülnek kifejtésre.

Munkám során elsősorban a magyar és külföldi szakirodalmat, folyóiratcikkeket, internetet, az érintett programok felhasználói kézikönyveit, és nem utolsósorban a tanulmányaim során szerzett ismereteimet használtam fel.

Szakedolgozatom elkészítése során elsősorban a tanulmányaim során megszerzett ismereteimet és tapasztalataimat, valamint a magyar és külföldi szakirodalmat, újságcikkeket és az internetes szakanyagokat használtam fel.

A bemutatások és leírások mellett több területen végeztem összehasonlító és elemző munkát. A téma terjedelme, szerteágazó volta miatt természetesen nem törekedhettem a teljességre. Mégis remélem, hogy ez a munka a weblapszerkesztés iránt érdeklődő olvasó számára segítheti a kérdéskör áttekintését, egyfajta összefoglalást, iránymutatást adhat.

*Köszönöm a hasznos tanácsait, a dolgozat elkészítésében nyújtott segítségét és szakmai útmutatásait témavezetőmnek, **dr. Boda István** tanszékvezető egyetemi docensnek.*

„A Web-nek egy univerzális médiumnak kell maradnia, nyitottnak és semlegesnek. Azt remélem, hogy ahogy a technológia egyre erősebbé, elterjedtebbé válik, és egyre több eszközön lesz elérhető, megtanuljuk olyan csatornaként használni, mely az együttműködést és a félreértések eloszlátását szolgálja minden téren.”
(Tim Berners-Lee)

1. A WORD WIDE WEB

Az **Internet** a számítógépes hálózatok világméretű összekapcsolásával létrejött hatalmas rendszer, amely az USA Védelmi Minisztériumának technológiájával és felszerelésével kezdte meg a működését még a hetvenes években. Napjainkban csaknem minden ország csatlakozik erre a legnagyobb kiterjedésű ún. világhálózatra. Az egymástól esetenként teljesen különböző hálózatok egymásnak átlátható módon tudnak állományokat továbbítani, elektronikus leveleket cserélni. [5]

A genfi CERN¹-nél szoftvermérnökként dolgozó Tim Berners-Lee és kollégája, Anders Berglund a '80-as évek végére egy olyan információkezelő rendszert dolgozott ki, amelyben a szövegek hivatkozásokat tartalmazhattak más dokumentumokra, így az olvasó könnyen átugorhatott egyik dokumentumról a másikra. Ezt a szolgáltatást az internet felhasználásával úgy biztosították, hogy lehetővé tették az egymástól távol dolgozó fizikusok számára a szakmai elképzeléseik, publikációik könnyű és gyors megosztását. A CERN szervezetén belül tehát az információt „jól darabolható”, egymásra ún. hyperlinkekkel hivatkozó dokumentumokba szervezték. Tim Berners-Lee az általa hypertextként (hiperszövegként) elnevezett tartalmak megjelenítéséhez készített egy kiszolgálót (egy böngésző- és szerkesztő programot) is, amelyet World Wide Web-nek nevezett el. 1990 decemberében a kiszolgálóval elsőként mutatott be sikeres együttműködést webes kliensek és szerverek között az interneten. A szoftver már 1991-ben megjelent, azonban a forráskódja csak 1993-ban vált szabadon hozzáférhetővé. Ez utóbbinak köszönhető, hogy sok vállalat és egyetem indított projektet a különböző böngésző programok fejlesztésének irányába. A tartalmak gyarapodása és a böngészők megjelenése következtében robbanásszerűen megnőtt az internet népszerűsége.

¹ CERN European Nuclear Research Facility, azaz CERN Európai Részecskefizikai Laboratórium (Ebben a laboratóriumban készítették el napjainkban a világ legnagyobb részecskegyorsítóját, hadronütköztető alagútját.)

1994-ben Tim Berners-Lee a massachusetts-i Technológiai Intézetnél megalapította a W3C²-t a CERN, a DARPA³, valamint az Európai Bizottság támogatásával. A W3C elsődleges célja a különböző technológiák és protokollok szabványosítása. Egy olyan Web felépítését kívánta ezzel biztosítani, amelyen az információ a Föld népességének lehető legnagyobb része számára elérhetővé válik, függetlenül attól, hogy milyen hardverrel, szoftverrel, hálózati infrastruktúrával, anyanyelvvel, kultúrával, sérültséggel rendelkezik. Ennek érdekében szoftverajánlásokat alkot a világhálóra, melyek alkalmazását a szoftvergyártókra bízta. A gyártók az ajánlást viszont kötelesek betartani, amennyiben a terméküket meg akarják jelölni, hogy a W3C feltételeinek megfelel. A W3C célként tűzte ki, hogy a WWW egyéb készülékekkel is (telefonnal, kézi számítógéppel, interaktív televízióval stb.) olyan egyszerűen, könnyen és kényelmesen legyen elérhető, amennyire egy asztali számítógépről.

Hazánkban először az egyetemeken és főiskolákon volt elérhető az internet. A '90-es évek elején az otthoni internethasználat megfizethetetlen volt. 1994-ben elindult a közösségi internet a teleházak révén. A nagy internetszolgáltatók megjelenése (pl. DataNet, Matáv stb.) és ezzel együtt a jelentős technikai fejlesztések megkezdése 1995-re tehető. Ekkor kezdődött el a tömeges elterjedése, és mára otthon, iskolában vagy munkahelyen minden harmadik-negyedik honfitársunk hozzáfér a világhálóhoz.

A **World Wide Web**, magyar fordításban világméretű nyitott háló, az interneten belüli grafikus felhasználói felülettel rendelkező superhálózat. Röviden az angol elnevezéséből képzett WWW mozaikszóval nevezhetjük. Megjelölésére gyakran használják a rövidült web és a világháló alakot is. Az internet sok szolgáltatása közül a legismertebbek a WWW, valamint az elektronikus levelezés, az e-mail. A web egy nyílt rendszer, amely az átlagfelhasználó számára főként szöveges és képi információ-hozzáférési lehetőséget jelent. A Web az interneten működő, egymással hiperlinkekkel összekötött dokumentumok rendszere.

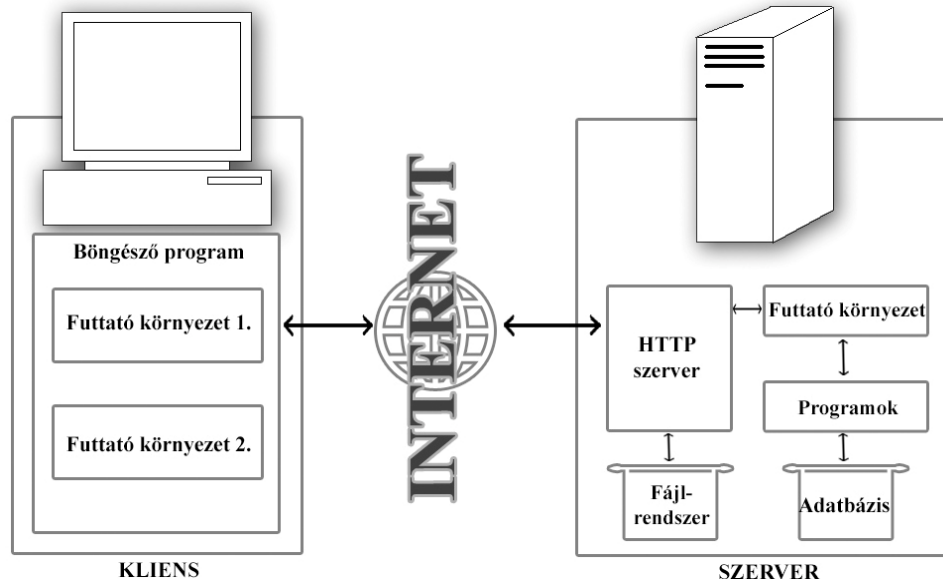
A webbel kapcsolatban több angol nyelvű szakszó keletkezett, melyek közül a leggyakrabban a web page és a web site kifejezéseket használjuk. A **web page** definícióját a World Wide Web Consortium (W3C) a következők szerint határozta meg: *„Egy vagy több webes forrásból származó információk összessége, amelyet egyidejű feldolgozásra szánnak,*

² World Wide Web Consortium: konzorcium, amely nyílt szoftverajánlásokat (szabványokat) alkot a világhálóra, melyek alkalmazását a szoftvergyártókra bízta.

³ Defense Advanced Research Projects Agency (Fejlett Védelmi Kutatási Projektek Ügynöksége) az USA védelmi minisztériumának kutatásokért felelős részlege.

és egy egyszeri URI határoz meg. Jobban mondva, egy weblap egy webes forrás, amely további webes forrásokat foglalhat magába egyetlen közös egységként való feldolgozásra, és azon egyedüli webes forrás URI-je hivatkozik rá, amely nem foglaltatik benne.”⁴ A web site fordítása a webhely, amely az egész, az önálló tájékoztató egységnek tekinthető tartalmat jelenti, beleértve az esetleges szolgáltatásokat is. A web site tehát több logikailag összekapcsolt web page együttese.

A web page magyar fordításaként a weblap, weboldal vagy a honlap használatos (bár ez utóbbi a Homepage jelentése). A web site kifejezésnek a webhely a magyar megfelelője. A felsorolt szavakat a magyar nyelvben egymás szinonimájaként, ugyanakkor többféle értelemben is használjuk. Az egyik értelemben egyetlen fájl jelöl, mely meghatározza a weblap alakját, és egy hipertextet közöl egy webböngészőnek, hogy aztán a weblap tartalmát a felhasználó számára vizuálisan értelmezhető formába adja. Egy másik értelmezés szerint az előbbiek szerinti weblap (főoldal) és a hozzá szorosan kapcsolódó fájlok, melyek elérési útvonalát mindig tartalmazza. Magába foglalja mindazon fájlokat (hipertext, kép, hang stb.), melyeket a formátumának megfelelően a feldolgozó böngésző program beépít a weboldalba annak struktúrája szerint.



1. ábra: Webes tartalomszolgáltatás architektúrája

⁴Az eredeti angol szöveg: „A collection of information, consisting of one or more Web resources, intended to be rendered simultaneously, and identified by a single URI. More specifically, a Web page consists of a Web resource with zero, one, or more embedded Web resources intended to be rendered as a single unit, and referred to by the URI of the one Web resource which is not embedded.”

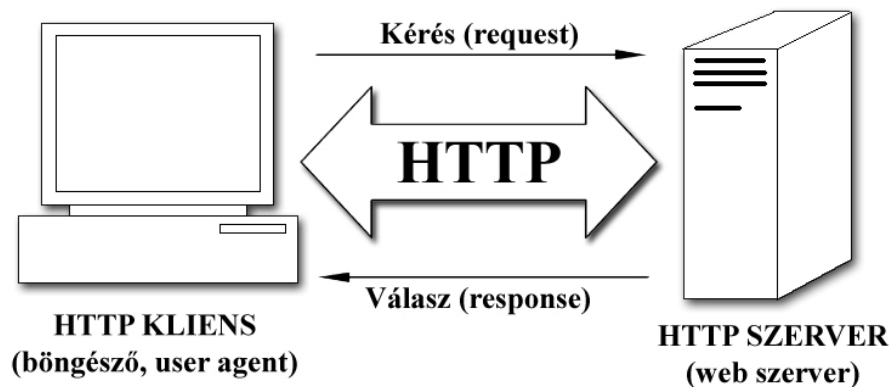
Forrás: <http://www.w3.org/1999/05/WCA-terms/#Link>

A fájlokat az internet hálózatára kapcsolt úgynevezett web szervereken, kiszolgálókon helyezik el, hogy aztán egy bizonyos protokollal (HTML fájlok esetén általában HTTP vagy HTTPS) egy kliens számítógépre letöltse a böngésző a kiszolgálóról. A harmadik értelmezés szerint az olyan weblapok összessége, amelyek szorosan kapcsolódnak egymáshoz. Leggyakoribb formáját adják ennek a kapcsolatnak az interneten egy domain alatt megtalálható weboldalak. A fentiek összefoglalásaként megállapítható, hogy a web három szabványra épül:

- az URL-re, amely leírja, milyen egyedi címmel kell rendelkeznie az egyes oldalaknak,
- a hipertext átviteli protokollra (http-re), amely megszabja, hogy hogyan küld egymásnak információt a böngésző program és a kiszolgáló,
- a hiperszöveg-leíró nyelvre (pl. HTML-re), amely az információkódolás eljárása (szabványa).

A **honlap** a webes szerver és a web-dokumentumok külön névvel ellátott együttese, melyek egyetlen URL segítségével érhetők el. Az interneten található címmel ellátott hely, amelyen keresztül azokhoz a weboldalakhoz lehet hozzáférni, amelyek URL címek alatt készültek. A honlap kifejezés tehát leginkább a címoldalt vagy kezdőoldalt jelenti. Egy közelítő tipológia szerint a weboldalak csoportjai a következők: tudományos, ismeretterjesztő; kereskedelmi; politikai, kulturális; hírszolgáltatások; személyi.

A weblapok azonosítása a webcímükkel, vagy más néven URL-jükkel, vagy URI-jükkel történik. Ez biztosítja, hogy a weboldalakra minden felhasználói környezetben egységes módon hivatkozzunk. A webcím szintaxisa az alkalmazott protokoll meghatározhatja. A felhasználók többsége a HTTP protokollt használja, amelynek esetében a webcím a *http://* szöveggel kezdődik. A honlapok karbantartói többnyire az FTP adatátviteli protokollt használják, és ekkor a cím *ftp://* szöveggel kezdődik.



2. ábra: A HTTP tranzakció lépései

2. A WEBLAP-SZERKESZTÉS TÖRTÉNETE, JELENTŐSÉGE

Korábban a Web nem volt ennyire látványos megjelenésű és könnyen kezelhető, mint manapság. Az olyan oldalak, melyekkel napjainkban találkozhatunk, a hipertext alapú protokollok létrehozása óta, vagyis a '90-es évek elejétől léteznek. Nagy segítséget jelent az, hogy az oldalakat, dokumentumokat egy hivatkozáshálózat szövi át, melynek segítségével könnyen átléphetünk a dokumentumnak egy másik pontjára, vagy egyik oldalról egy másikra navigálhatunk. Ezt a szolgáltatást a hipertext alkalmazásának köszönhetjük.

A web-dokumentumok nyelvének az őse az IBM cégtől elindulva a '80-as évek elején keletkezett. A nyelv azon a felismerésen alapult, hogy a legtöbb szövegnek számtalan közös része van (címe, alcíme, főrésze stb.), és ha az egyes részekre vonatkozó formázóutasításokat a szövegbe lehetne belefoglalni, akkor könnyen lehetne hardver- és szoftverfüggetlen, formázott szövegeket létrehozni. Az IBM kifejlesztette az előbbieknak megfelelő nyelvet, amelyet GML⁵-nek nevezett el. A tartalomról információkat hordozó címkék (angolul tagek) használatának köszönhetően jól szerkeszthető és kereshető dokumentumokat lehetett általa definiálni. Kezdetben a GML-t csak az IBM hatalmas nyomdarendszere alkalmazta. A '80-as évek elején az ANSI⁶ felkérte a GML megalkotóit egy szabványos, általános dokumentum leíró nyelv kidolgozására. Így született meg az SGML.

A weblapok készítésének alapvető célja nem más, mint az információközlés, amely esetleg valamilyen interaktív tevékenységgel is kiegészül. Ez megvalósítható azzal, hogy a meglévő tartalmainkat (szöveg, kép, hang, videó stb.) egyszerű módon „felrakjuk” egy oldalra, de készíthetünk egy híreket, fórumot is tartalmazó oldalt, vagy esetleg egy webportált. A webportál a weboldalaknál összetettebb, testreszabható rendszerek. Számos modul beépítésével a funkcióhoz és a közvetített tartalomhoz igazítható a szerkezete (pl. hírlevél, fórum, chat, blog, webáruház, feltöltés, szavazás stb.).

⁵ Generalized Markup Language - általános jelölőnyelv

⁶ American National Standards Institute - Amerikai Nemzeti Szabványügyi Hivatal

3. A LEÍRÓ JELÖLŐ NYELVEK KIALAKULÁSA

A szövegek valamilyen célból történő megjelölése még a számítógépek megjelenése előtt elterjedt. A nyomdászatban a szedők számára már szinte a kezdetektől fogva jelzésekkel látták el a kéziratot, jelezve a nyomtatvány végső formáját meghatározó betűkészletet, stílust, margót stb. Ezeket a jelzéseket általában a szerkesztők, korrektorok és grafikusok helyezték el a kéziratban.

Az adatok formázott dokumentumként való tárolása már a '60-as évek óta megoldott, azonban ezek módszerének az alapvető célja az volt, hogy a dokumentum „jól mutasson” monitoron vagy nyomtatásban. Sajnos az ilyen dokumentumok programmal történő feldolgozása nagyon nehézkes volt, mert nem támogatták az adatok strukturálását, az adatok közötti egyszerű keresést. A különböző számítógépes rendszerek és programok közötti információcsere a hálózatalapú alkalmazások fejlődésével egyre bonyolultabbá vált. Alapvető követelményként jelentkezett az információkkal szemben, hogy önleíróak legyenek, annak érdekében, hogy a kliens program önállóan képes legyen feldolgozni azokat. Ezt olyan jelölési rendszer kidolgozásával sikerült megoldani, amely a dokumentumban ún. metaadatokat helyez el. A **metaadat** egy másik adatot jellemez, leírja az adatot, vagyis Caplan megfogalmazása szerint adat az adatról.⁷ A metaadatok önmaguk is adatok, róluk is születhetnek további metaadatok. A metaadattal összekapcsolt tartalmat tartalomcsomagnak nevezzük. A metaadatot az eredeti tartalomhoz csatoljuk például a digitális tartalom állományába kódolva, vagy mutatók, linkek teremtenek kapcsolatot a tartalom és a leíró adatok között. A metaadat használatának alapfeltétele, hogy egyértelműen elválasztható legyen magától az adattól.

A **jelölés** olyan módszer, amely sajátos karaktereket (meta karaktereket) ágyaz be szöveges fájllokba, és ezek a karakterek közlik a böngészővel, hogyan kell kezelnie a fájl tartalmát. A metaadatok leírására nem programozási, hanem „cédulázó”, jelölő nyelveket (pl. HTML, XML) használunk. [2]

A **jelölő nyelv** mesterséges nyelv, speciális karakterek és azok képességeinek formális csoportja, amelyet arra használnak, hogy rögzítsenek egy speciális módszert az olyan fájlok megjelenítésére, amelyek a jelölő nyelvet tartalmazzák. A jelölő nyelv elnevezés onnan származik, hogy az információt annak tartalma, illetve szerkezete alapján jelöli meg.

⁷ Caplan, P.: Providing access to online information resources: A paper for discussion. 1992.

Segítségével adott szöveget egy meghatározott cél érdekében (pl. megjelenítés, számítógépes értelmezés, strukturálás stb.) speciális jelölésekkel láthatunk el. Jelölőnyelv segítségével lehet egy weboldal tartalmi felépítését és struktúráját is egyértelműen meghatározni. A jelölő nyelveket tehát elsősorban a tartalom leírására tervezik. A jelelő nyelv konkrét alkalmazásainak fejlődése során persze megjelentek olyan nyelvi elemek is, amelyek megjelenésre helyezik a hangsúlyt, azonban a tapasztalat azt mutatta, hogy az ilyen elemekkel átszőtt dokumentumok karbantartása komoly megpróbáltatást jelent a felhasználó számára. A megjelenítés finombeállítására később kifejlesztették a stíluslapokat. Az előbbi gondolat *Eric Tilton* szerint „... az alábbi szabályban összegezzhető:

- *ha dokumentumunkban az információt aszerint jelöljük, ami, ahelyett, hogy aszerint jelölnénk, hogyan kellene megjeleníteni,*
- *akkor a böngészők megfelelő, és professzionális külalakot adnak neki.*”⁸(1995)

⁸ Az eredeti angol szöveg: „*The trust you must have can be summarized by the following rule:*

- *if you mark up a document so that your information is labeled as what it is instead of as how it should be displayed,*
- *then browsers will render it in a way that is appropriate and professional-looking.* „

Forrás: <http://www.ology.org/tilt/cgh/>

4. A WEBLAP-SZERKESZTÉS LEÍRÓ JELÖLŐ NYELVEI ÉS ALKALMAZÁSAI

A jelölő nyelveknek kifejlesztették az általános változatait, amelyeket szabványok írnak le, és azoknak konkrét alkalmazásait is. Az általános jelölő nyelvek (pl. SGML, XML, stb.) esetében a bennük használható jelöléseknek (tageknek) csak a szintaktikája meghatározott, a szemantikája viszont nincs rögzítve. A nyelv szintaxisa azt a szabályrendszert jelenti, amely megszabja, hogy a nyelvben hogyan lehet nyelvi elemeket létrehozni. A szintaxis tehát megadja a használható kulcsszavakat és meghatározza a nyelvi elemek felépítésének szabályait. A szintaxis leírására a metanyelvek szolgálnak. A szemantikát a kulcsszavak jelentését definiáló szabályok összessége alkotja. Az általános jelölő nyelvek segítségével újabb jelölő nyelveket lehet definiálni (például az SGML-ből a HTML, az XML-ből a XHTML stb.). Az általános jelölő nyelvek konkrét alkalmazásai már az egyes jelölésekhez adott jelentést kapcsolnak, vagyis a jelölések értelmét az adott alkalmazás szabja meg. A jelölések értékkészlete véges, és ezt a nyelv specifikációja határozza meg. A szintaktikai kötöttségek lehetővé teszik, hogy a dokumentum egy általános célú értelmezővel feldolgozható (beolvasható, kiírátható, ellenőrizhető).

4.1. SGML

Az SGML (Standard Generalized Markup Language) egy szabványos (ISO 8879:1986 szabvány) **általános** kiterjesztett jelölő nyelv. A nyelvet leíró szabvány 1986-ban került kiadásra, amely szintaktikai szabályokat biztosít a szöveg hierarchikusan rendeződő elemeinek formális leírásához. Általa elvben bármilyen dokumentum leírható, függetlenül az azt tároló és megjelenítő számítógépes környezettől. Kidolgozásának elsődleges oka az volt, hogy az erősen számítógépesített vállalatokhoz elektronikusan érkező dokumentumok egységes gépesített feldolgozása gyakorlatilag lehetetlen volt, nemcsak a formátumbeli eltérések, hanem a belső szerkezetük heterogenitása és a dokumentumok elemeinek azonosíthatatlansága miatt sem. A szabvány alkotók elsődleges célja a strukturált szöveg és információ megosztásának biztosítása az Interneten keresztül, lehetővé téve ezáltal az elektronikus formában létező szöveg alapú dokumentumok egységes feldolgozását, cseréjét és

közzétételét. Azáltal, hogy a különböző szövegek egymással kompatibilis formátumban kerülnek kódolásra, biztosíthatók azok SGML-t támogató szoftverekkel történő analízisa. Megszületése óta meglehetősen stabil, ami a nyelv tulajdonsággazdagságának és rugalmasságának köszönhető. A weboldalak lapleíró nyelve lett. Bonyolultsága miatt azonban soha nem vált népszerűvé. Az SGML az 1986-os megjelenését követő 10 évben gyakorlatilag változatlan maradt. [2]

4.1.1 Az SGML szabványról

A szabvány két fő része: az információ szerkezetének megadására szolgáló metanyelv és az SGML dokumentumok tárolási formátumának leírása. Ezt meghaladóan kitér több technikai kérdésre, pl. az SGML deklarációra, amely az alkalmazás konfigurálására, az alapértékek felülírására használatos. Az SGML szabvány alkalmazásaiban a metaadat elválasztására az adattól ”<” jelpárt használjuk. A szövegrögzítés célja határozza meg azt, hogy milyen jellegű metaadatot érdemes rögzíteni. Szöveges adatok esetében olyan jellemző tulajdonságot kell rögzíteni, mely képes kifejezni az adott szegmentum viszonyát a dokumentum többi szegmentumához. A szabvány rögzíti, hogy a definiált szegmentumok milyen számítógépes eljárással azonosíthatók vagy dolgozhatók föl, ezt a szövegfeldolgozó programrendszerek feladatává teszi. Feltételezi egy olyan programnak a használatát, amely kapcsolatot teremt a szegmentumok, a szegmentumok definíciója, és a szöveg lehetséges felhasználása között.

Az SGML szabvány nem rendelkezik az információ megjelenítéséről, hiszen a filozófiájának lényege a tartalom és forma elválasztása. Az információ független az ábrázolástól. Ez biztosítja, hogy az információ annak bármiféle változtatása nélkül a mindenkori technikának és médiának megfelelő módon legyen ábrázolható. Ez a stíluslapok elkészítésével lehetséges, és a tartalmi elemekhez a megjelenítési információ hozzárendelésével történik.

A stíluslapok alkalmazásának legjelentősebb előnye az, hogy egyetlen stíluslappal az adott dokumentumtípushoz tartozó összes (ezer vagy millió) dokumentum megjelenítését egységesíthetjük. Egy dokumentumtípushoz viszont tetszőleges számú stíluslapot lehet rendelni, ami lehetővé teszi, hogy változtatás nélkül különböző formában (pl. olvasási vagy

nyomtatási nézetben), sőt különböző tartalommal (teljes terjedelemben vagy részben) jeleníthető meg.

A szabvány második része a dokumentumok tárolási formátumát rögzíti. Az elemek kezdetét `<elemnév>`, a végét `</elemnév>` taggel (címkével) jelöljük. Az SGML formátumban a 7 bites ASCII karaktereket alkalmazzuk. Ez garantálja a tökéletes hordozhatóságot az ASCII és EBCDIC platformok között. A többi karaktert, így pl. az ékezetes magánhangzókat entitásként (entity) lehet megadni, pl. az "ö" és "ő" karakterek reprezentációi: `ö` és `ő`. Az ismert nyelvek egyedi betűinek és sok egyéb általánosan használt speciális karakternek ISO szabványosított entity-neve van. A Latin 2 kódtábla karaktereinek ISO neveit tartalmazza az *1. sz. melléklet*. Amennyiben más nevet vagy egyéb karaktert akarunk entitásként megadni, akkor ezt a DTD-ben jelezni kell. A karakterek helyes megjelenítése már az SGML-t feldolgozó szoftver feladata.

4.1.2 DTD – dokumentum típusdefiníció

Az SGML technológiában a leglényegesebb feladat a dokumentumok típusokra osztása és a típusok szerkezetének leírása, azaz a DTD elkészítése. Az SGML a dokumentum felépítésének szabályait a DTD-ben tárolja. A DTD (Document Type Definition - dokumentum típusdefiníció) egy olyan módszer, amellyel az adatelemekre és a szervezetre hivatkozhatunk. A dokumentumtípus összetett adatszerkezet, melyet formálisan mindig a feldolgozni kívánt szöveg alkotóelemeivel és azok szerkezetével definiálunk, a típusdefiníció pedig leírja, hogy az egy dokumentumtípushoz tartozó dokumentumok milyen elemeket, attribútumokat, értékeket és hivatkozásokat tartalmazhatnak. A DTD-vel adjuk meg a dokumentum hierarchikus szerkezetét is, amely akár faszerkezetben is ábrázolható. A DTD egy elemdeklarációkból (attribútumok hivatkozások a dokumentumon belüli és kívüli objektumokra) és jellemződeklarációkból felépített szerkezet, amely egy dokumentum értelmezési szabályait tartalmazza. A DTD tartalmazási modellje megadja, hogy az adott típusú dokumentum milyen kötelező vagy választható elemekből áll. Az elemekre sorrendi és ismétlődési megkötések szabhatók. DTD adja meg a dokumentumban használható entity-ket, melyek speciális karakterek, hosszabb sztringek, állományon kívüli szövegrészek, elemcsoportok, képek, táblázatok stb. jelölésére szolgálnak.

A DTD-nek a következők a jellemzői:

- Egy „fa” szerkezet, amely lehetőséget ad komplex, a relációs modellben nehezen leírható struktúrák megadására.
- A fa gyökere adja meg a dokumentum típusát.
- A fa, és így a dokumentumok elemei az SGML filozófiájának megfelelően tartalomra és nem formára vonatkoznak.
- Az elemek nevei szabadon választhatók.
- Könnyen feldolgozható ASCII fájl, amely önmaga is SGML formátumú.
- A dokumentum tartalmazhat olyan részeket, amely nyomtatásban nem jelennek meg.

Minden SGML dokumentumhoz tartoznia kell DTD-nek, amely elhelyezkedhet a magában a dokumentumban, de gyakoribb, hogy a dokumentum csak egy hivatkozást tartalmaz a DTD-re, amely külső fájlként létezik.

Az SGML alkalmazások meghatározó feladata a DTD tervezés, amely komoly szaktudást igényel. A tervezéskor a meglévő dokumentumokat, a jövőben várható igényeket, a feldolgozás folyamatát, az információ hasznosítását és mozgását is figyelembe kell venni. Egyes DTD tervezési feladatok hónapokat, esetleg éveket vesznek igénybe. A munkát segítheti a több szabványosított, vagy annak tekintett DTD. Ezek a DTD-k közvetlenül is átvehetők, vagy módosításokkal testre szabhatók. Ilyen például az ISO 12083 Book DTD, amely általános könyvek leírására szolgál.

A saját készítésű vagy szabványos DTD-k a következő kulcsszavakat vagy szintaktikai egységeket tartalmazhatja:

- deklarációk:
 - = ELEMENT – elem-típus deklarációk,
 - = ATTLIST – attribútum-lista deklarációk,
 - = ENTITY – entitás deklarációk,
 - = NOTATION – adatformátum deklarációk,
 - = paraméter-egyed definíciók és hivatkozások.
- feldolgozási utasítások
- megjegyzések

A deklarációk szintaxisa:

```
<!kulcsszó név minimalizálási szabályok (tartalmi modell)>
```

Az elem-típus deklarációval új elemet adhatunk meg, és meghatározhatjuk a megengedett elemtartalmat. Elemként a dokumentum szerkezeti összetevőit jelöljük. Szerencsés, ha az elemek neve a tartalmukra utal.

```
<!ELEMENT gyerekek (gyerek1, gyerek2, gyerek3)>
```

Attribútumként az olyan tulajdonságokat jelöljük, amelyek az elemet többletinformációval egészítik ki. Egy elemhez több attribútum is rendelhető. Az attribútumok nevének nem kell egyedinek lenniük a dokumentumban, csak az adott elemre vonatkozó attribútum-listában. A DTD-kben a attribútumokat az „ATTLIST” deklarációval adhatjuk meg. Az attribútum-deklaráció szintaxisa:

```
<!ATTLIST elem-név attribútum-név attribútum-típus alapérték>
```

Az attribútum-típus megjelölésére meghatározott kulcsszavak közül adunk egyet, amely meghatározza, hogy attribútum milyen értéket vehet fel. A 2. sz. *melléklet* összehasonlításban tartalmazza az összes lehetséges kulcsszót. Az alapérték az attribútum alapértelmezett értékének jelölésére szolgál, mely a következő lehet:

- #REQUIRED (kötelező): kötelező megadni valamely értéket,
- #IMPLIED (hallgatólagos, bennfoglalt): nem kötelező értéket megadni,
- #FIXED (fix, rögzített): a megadott érték fix, nem változik,
- #CURRENT (aktuális, legutóbbi): hiányesetben az ilyen típusú elemnél legutóbb megadott attribútum érték kerül felvételre.

```
<!ATTLIST kuldo ceg CDATA #FIXED "Microsoft">
```

Az entitás (egyed) az SGML.-ben a jelölt dokumentum valamely azonosított, névvel ellátott részét jelenti, függetlenül bármely szerkezeti megfontolástól. Az entitások olyan „változók”, melyek más szövegelemekre mutatnak. Az entitás lehet egy karakterfüzér, vagy akár egy szöveges állomány is. Az entitást entitáshivatkozással helyezzük a dokumentumba. Az entitásoknak az alábbi típusait különböztetjük meg:

- általános entitások: ennek a hivatkozása belső vagy külső forrásra utaló lehet
 - = *belső entitás-deklarációval megadott*: belső forrásra utaló, szintaxisa

```
<!ENTITY entitás-név "entitás-érték">
```
 - = *külső entitás-deklarációval megadott*: külső forrásra utaló, szintaxisa

```
<!ENTITY entitás-név SYSTEM "URI/URL">
```
- paraméter entitások: a DTD egy összetett részének több helyen történő szerepeltetésére használható, amelynek deklarációja után az egyedre a DTD-n belül bárhol lehet hivatkozni.

Példa a belső entitás deklarációra:

```
<!ENTITY szerzo „Vass Albert”>
```

A NOTATION deklarációval meghatározzuk az adatok formátumát és az azokat feldolgozó alkalmazást. Amennyiben valamely egyed nem SGML adatot tartalmaz, fel kell ismernie az objektum típusát valamint azt, hogy miként ágyazza be a dokumentumba. Ismert és definiált formátum használata esetén a PUBLIC kulcsszót célszerű alkalmazni az egyértelmű azonosítás céljából. Például:

```
<!NOTATION BMP PUBLIC "+//ISBN 0-7923-9432-1::Graphic  
Notation//NOTATION Microsoft Windows Bitmap//EN">
```

A megjegyzések rendszerint a dokumentum adott részével vagy forrásával kapcsolatos információk. A programozás során gyakrabban használják, de a szövegjelölésnél is előnyösen használhatók. Szintaxisa:

```
<!-- Megjegyzés szövege -->
```

4.1.3 Az SGML felhasználási területei, előnyei és hátrányai

Az SGML nyelv célszerűen használható digitalizált szövegek hosszú távú archiválása során, mivel a visszakeresést és feldolgozást csak ilyen, a hardver- és szoftvereszközöktől független formátum tudja biztosítani. Mindaddig, amíg majdnem emberi intelligenciával rendelkező számítógépeink nincsenek, valamilyen módon kénytelenek leszünk megjelölni a szoftverek számára az információkat annak érdekében, hogy azokat megtalálják. Különösen fontos ez a jelölés akkor, ha egy-egy szónak többféle értelme is van (pl. Java – amely jelölhet egy szigetet vagy egy programozási nyelvet). A következő példa a gép számára összetettebb értelmezési feladatot jelent. Legyen egy e-mail üzenet a következő:

```
Feladó: Naár János  
Címzett: Nagy István  
Tárgy: megrendelés  
Üzenet: Postai utánvétellel megrendelem Öntől a hirdetésében szereplő Siesta  
gázkályhát a 3958 Hercegkút, Petőfi u. 33. címre.
```

Az e-mail üzenet géppel is feldolgozható vagy kereshető módon az alábbiak szerint írható le:

```
<felado>Naár János</felado>  
<cimzett>Nagy István</cimzett>  
<targy>megrendelés</targy>  
<uzenet>Postai utánvétellel megrendelem Öntől a hirdetésében  
szereplő Siesta gázkályhát a 3958 Hercegkút, Petőfi u. 33.  
címre.</uzenet>
```

Az SGML ezt a jelölést is jól ellátja. [1]

A dokumentumok újrahasznosítása során problémát jelenthetnek a különböző formátumok és a hozzá rögzített megjelenés is. Újrahasznosítás céljára jól megfelel az SGML állomány – ellentétben a különböző szövegszerkesztőkkel előállított mesterien megformált dokumentumokkal –, mivel a dokumentum formai jellemzőit egy, az SGML állományokon kívüli fájlban tárolja. Az utóbbi évekig WYSIWYG jelzettel ellátott irodai szövegszerkesztők egyáltalán nem törődtek a tartalom és dokumentumban található struktúra jelölésével. A szövegszerkesztő programjának ezt a fogyatékoságát orvosolandó, a Microsoft 2005-ben bejelentette, hogy a 2006-ban megjelenő Word változat már az SGML egyszerűsített változatát (XML nyelvet) fogja használni belső formátumként. (Már a 2003-as változat képes volt XML-dokumentumként menteni.) [1]

A fentiek összefoglalásaként szolgáljon az alábbi felsorolás, amely az SGML előnyeit igyekszik kiemelni:

- Az információk leírására, tárolására elfogadott gyártófüggetlen szabvány.
- A szabvány vagy annak része átültethető más nyelvekbe is (pl. HTML).
- Alkalmas az információ szerkezetének leírására.
- Az általa leírt információ számítógéppel könnyen feldolgozható, a könnyű és gyors feldolgozhatóság nem függ az alkalmazott számítógépes platformtól.
- A tárolt információ sokoldalúan újrafelhasználható, vagyis magában hordozza a más formátumokba való konverziós lehetőséget.
- A tárolt információ hordozható, azaz bármilyen számítógépes környezetben használható.
- A tárolási formátum révén a keresett információt gyorsan, pontosan megtalálható.

Nem szabad eltitkolni azt sem, hogy az SGML használatának az előnyök mellett hátrányai is vannak.

- A nyelv nehezen integrálható, mivel egy SGML dokumentum több fájlból áll. Értelmezéséhez szükséges egy dokumentumtípus-definíciót tartalmazó fájl (DTD) és szükség lehet egy speciális stíluslapra is (XLS, CSS).
- Egységes formában feldolgozott dokumentumokat csak közös DTD-vel készíthetünk.
- Bonyolult, ezért a szabvány megismeréséhez sok idő és alapos háttértudás szükséges.
- Az SGML feldolgozó programcsomagok drágák.

- Napjainkban a legtöbb szoftvert, feldolgozó alkalmazást már nem az SGML-hez, hanem az utódjának számító és egyszerűsített formájának tekinthető XML-hez fejlesztik.

4.2. HTML

A HTML (HyperText Markup Language, magyarul hiperszöveg-leíró nyelv) az SGML-ből származó leíró jelölő nyelv, az abban meghatározottak egyik nyelvmegvalósítása, amelyet weboldalak készítésére használunk. A „hypertext” elnevezés a HTML fájlok kapcsolhatóságaira (linkelés) utal. A HTML gazdag szókinccsel és szintaxissal rendelkezik. A HTML állomány olyan szövegfájl, amely a szövegen kívül tartalmaz ún. "HTML-tag"-eket (formázóutasításokat), melyek befolyásolják a dokumentum megjelenítését, és megjelenítendő, letöltendő, stb. objektumokra történő hivatkozásokat. A tageket a böngésző értelmezi és hajtja végre. [4]

Bíró [1] a HTML nyelvet az SGML „gyermekeként” említi. A HTML dokumentum egy speciális jelöléseket (elemeket) tartalmazó szövegfájl, amelyben a jelölések körülveszik a dokumentum szövegeit, és megadják, hogy a webböngészők hogyan értelmezzék a dokumentum megjelölt részeit. Az HTML nyelvet eredetileg szöveges és képi tartalom gyors, platformfüggetlen közzétételére fejlesztették ki, amely napjainkra multimédiás tartalommal bővült. A HTML dokumentum kiterjesztése a `.html` vagy `.htm`.

4.2.1 A HTML története – a különböző szintek jellemzése

A HTML nyelvnek az évek során számos szintje, hivatalos verziója jelent meg. Ezen verziók mindegyike szabvány.

A **nulladik** szinten (HTML 1.0) alakult ki az alapvető szerkezet, a nyitó és záró címek, a fejléc és a törzs. Már biztosította a kezdőcímke előtti dokumentumtípus deklarációt (DTD). A tartalom a dokumentum törzsében helyezkedik el. Már létre lehetett hozni bekezdéseket, hivatkozásokat, fejléceket, felsorolást. Az eredeti SGML alapján épül fel, lehetővé téve más fájlokra való hivatkozásokat is. Ezt a szintet minden böngészőprogram

támogatja. A szakirodalomban több szerző a HTML 1.0 verziónak a Tim Berners-Lee által létrehozott változatot tartják.

Az **első** szint (HTML 2.0) nem hozott túl sok változást, de innentől különül el a böngésző által szabályozott és a dokumentum szerkezetében létrehozott formázás. A HTML 2.0 volt az első publikált hivatalos verzió, amely 1995 novemberében került kiadásra. A korábbi változat kiegészült azzal, hogy lehetővé tette bizonyos szövegrészek megkülönböztetését, színének, formájának megváltoztatásával. Új elemként jelent meg, hogy lehetővé vált képek elhelyezése a dokumentumban, azonban a képet nem lehetett körbefolytatni szöveggel, azaz a kép mellett semmilyen szöveges információ nem lehetett. Új formázási lehetőségként jelent meg pl.: a `<code>` ami forráskód jelölésére, az `<EM>` ami a kiemelésre szolgált. A karakterformázási lehetőség is kibővült: a `<b>` – félkövér, `<i>` – dőlt karakterek, a `<TT>` az azonos betűszélességet jelenti.

A **második** szint a HTML 2.0 kiegészítése. Bevezetésre kerültek az űrlapok, az ezen belüli többsoros szövegbevitel és a kiválasztható opciók (listából). Az űrlapokkal lehetővé tette az olvasó számára az adatbevitelt, vagyis az adatok kétirányú áramlását, azaz egy minimális interaktivitást is biztosított. Ez a verzió szabvány lett, melynek lezárására 1996-ban került sor, és a későbbi fejlesztések alapjául szolgált.

Az első igazán elterjedt **harmadik** szintet jelentő HTML 3.2 verziót 1996-ban fogadta el a W3C, ami jelentős változásokat hozott. Tovább bővültek a lehetőségeink: lehetővé vált a táblázatok, matematikai képletek és vezérlőelemek használata, innentől van lehetőségünk Java appletek⁹ beágyazására, és scripteket is innentől használhatunk. Itt vált először lehetővé az alsó és felsőindex használata, ami a tudományos témájú publikációk fontos kelléke. A képek körüli szöveg körbefolyás is ezen a szinten jelent meg. A verzióba bekerült néhány böngésző-specifikus tag, ugyanis a böngészőprogramok (elsősorban a Mosaic és Netscape Navigátor) fejlesztői számos alkalommal eltértek az eredeti specifikációtól azzal, hogy új tageket és paramétereket vezettek be. Egy általános statikus HTML oldal ma is nagy részben ehhez a szinthez tartozó jelöléseket használ.

A **negyedik** szintet jelentő HTML 4.0 verziót 1997 végén tette hivatalos ajánlássá a W3C, majd némi módosítás után 1998 áprilisában kiadta a végleges verziót, mely megfelel az ISO 8879:1986 szabványnak. Ez már igazából egy SGML alkalmazás. Ez a verzió a

⁹ Java programozási nyelven írt programkód, amely a szerver oldalon tárolódik, böngészőből indítva a felhasználó számítógépén fut le.

következőkkel egészítette ki elődjait: stíluslapok, keretek (frame-ek) használata, továbbfejlesztett szöveg és táblázatirány meghatározás, űrlapok, fogyatékos felhasználók számára elérhetőség, objektumok beágyazása, továbbfejlesztett táblázatok, a nemzetközi karakter készletek, a jobbról balra olvasás támogatása. Lehetővé tette a tartalom és a megjelenítés szétválasztását lehetővé tevő stíluslapok használatát. Talán napjainkban az egyik leggyakrabban használt verzió. A HTML 4.01-nek három doctype-ja lehet: Strict, Transitional és Frameset. Ez a verzió 1999 decemberében jelent meg, amely néhány javításban különbözött elődjétől, és úgy tűnt, hogy ezzel be is fejeződik az SGML nyelven alapuló HTML nyelv fejlesztése. Az SGML nyelv helyett az XML szabvány került előtérbe. Jelentős lépésnek számított, hogy 2008 januárjában a W3C nyilvánosságra hozta a HTML 5 első vázlatát. Kidolgozásában az összes jelentős böngészőgyártó aktívan részt vett, ezáltal a jelenlegi kompatibilitási problémák minimalizálásra kerültek. Lényeges újítás, hogy a HTML ezután már csak tényleg a tartalomra és felépítésre szorítkozik, mindenféle formázás és stíluselem CSS-be kerül. A HTML 5-öt a fejlesztők igyekeznek kompatibilissé tenni a HTML 4.01-gyel és az XHTML 1.0 és 1.1 verziókkal is.

A HTML 4 verzió XML alapokra helyezésével létrejött az **ötödik** szint, az XHTML, amely jelenlegi és jövőbeli dokumentumtípusok és modulok családjába sorolható.

4.2.2 A HTML elemek szintaxisa

A HTML kód logikailag két részből áll: az egyik maga a megjelenítendő szöveg, a másik pedig azok az elemek, amelyek formázzák a szöveget, egyéb objektumokat ágyaznak be, vagy hivatkozásokat hoznak létre. A dokumentum minden egységének (bekezdés, kép, táblázat stb.) tulajdonságait szabványos kódokkal határozhatjuk meg. Az elemek többségének van nyitó (első) és záró (befejező) tagje, amelyek szöveget fognak körül. A tag egy "<" vagy ">" jel és egy ">" közötti azonosítóból áll, pl. <title>, <head>, <body> stb. A záró tag azonosítója megegyezik a nyitó tagjével, csupán abban különbözik, hogy "</" jel vezeti be. Például a <title> záró párja a </title>.

A HTML elemek többsége a következő szerkezetű:

```
<utasítás paraméter1="érték1" paraméter2="érték2">szöveg</utasítás>
```

A nyitó tagek az utasításon (azonosítón) kívül paramétereket (attribútumokat) és ezek értékeit tartalmazhatják. Az attribútumokkal szabályozható a körülfogott szöveg formázása.

Az utasítás és paraméterei a szövegre vonatkozik.

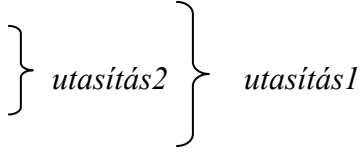
Egy konkrét példa:

```
<p align="center">Ez a bekezdés középre igazítottan jelenik meg</p>
```

A szabvány szerint néhány elem esetében nem kötelező a záró tag használata, de a jobb áttekinthetőség érdekében célszerű használni. Néhány elemnek nincs záró tagje, pl. `<br>` amely egy sortörést eredményez stb. A tagnél nem különböztetjük meg a kis- és nagybetűket.

A HTML dokumentumban az elemek egymásba ágyazhatók, és az ajánlás szerint nem lapolhatják át egymást (de lehetséges), ami azt jelenti, hogy az utoljára megnyitott utasítást kell legelőször bezárnunk. Az egymásba ágyazást szemlélteti az alábbi általános példa:

```
<utasítás1>  
  <utasítás2>  
    Szöveg  
  </utasítás2>  
</utasítás1>
```



4.2.3 A HTML adattípusai

Ebben a részben azokat az adattípusok kerülnek ismertetésre, amelyek az elemek tartalmi részében vagy az attribútumok értékeiként szerepelhetnek.

URL—Egy URL-t (címet) jelent.

Szöveges adat

Név: Betűvel kell kezdődik és tetszőleges számú karakterből állhat.

Szöveg: Tetszőleges szöveget tartalmazhat (speciális karakterkódokat is). Attribútumként használva az előtte/utána lévő szóközöket a böngésző nem veszi figyelembe.

Szín – Kétféle módon adhatjuk meg.

Hexadecimális módon, pl. #808080

Előre definiált név segítségével, pl. gray.

Számadatok – Ezek a számadatok általában valamilyen méretet jelölnek.

Abszolút: Képpontokban adjuk meg, pl. "100".

Relatív: A felhasználható terület egy részét szeretnénk kihasználni. Ezt százalékos adatban adhatjuk meg, pl. "50%".

Nyelvkódok – Egy emberi nyelvre utaló kódot tartalmaznak ezek az adatok.
hu – magyar, *de* – német, *fr* – francia, *en* – angol, *en-US* – amerikai angol

Egyszerű karakterek – Egyes paramétereknek csupán egyszerű karakterekre van szükségük, amelyek lehetnek szimpla betűk, de ugyanúgy írhatunk ide speciális kódot is.

Tartalom típusok – dokumentumokba beágyazható objektumok. Leggyakrabban használt típusok: *"text/html"*, *"image/png"*, *"image/gif"*, *"video/mpeg"*, *"text/tcl"*, *"text/javascript"*, *"text/vbscript"*.

Dátumok és Idők – *YYYY-MM-DDThh:mm:ssTZD*

YYYY – év (négy számjegy), *MM* – hónap (01-től 12-ig), *DD* – nap (01-től 31-ig), *hh* – óra (00-tól 23-ig), *mm* – perc (00-tól 59-ig), *ss* – másodperc (00-tól 59-ig), *TZD* – időzóna jelző.

Ha az időpontot nem ismerjük pontosan, akkor a perc, másodperc stb. helyére 00-t kell írni.

Keret célnevek – A következőkön kívül minden keretnévnek betűvel kell kezdődnie.

_blank – egy új névtelen ablakba kell betöltődnie

_self – az aktuális dokumentumot tartalmazó keretbe kell betöltődnie

_parent – az aktuális dokumentumot tartalmazó keret szülőkeretébe kell betöltődnie

_top – minden keretet felülírva az eredeti ablakba kell betöltődnie

Médium-leírók – Meghatározzák, hogy aktuális elem milyen médiához készült.

screen – számítógép képernyő (nem lapozó)

tty – fix karakterkészlettel rendelkező, grafikus képességekkel nem rendelkező eszközök

tv – televízió jellegű berendezések

projection – kivetítő készülékek (projektorok)

handheld – kézi megjelenítők (kis képernyő, bittérképes grafika, korlátozott sáv szélesség)

print – lapozott, nyomtatási kép a képernyőn

braille – Braille-berendezés

aural – felolvasó egységek, beszédszintetizátorok

all – minden fenti eszköz

Hivatkozás-típusok – Különböző hivatkozási típusok.

Alternate – valamiben különböző (pl. más nyelvű), de lényegében hasonló információkat tartalmazó dokumentumra mutat.

Stylesheet – egy külső stíluslap leírásra mutat.

Start – egy aktuális dokumentumrendszer első oldalára mutat.

Next – egy dokumentumsorozat következő oldalára mutat.

Pre – dokumentumsorozat előző oldalára mutat.

Contents, ToC – dokumentum(ok) tartalomjegyzékeként szolgáló dokumentumra hivatkozik.

Index – a dokumentum indexére mutató hivatkozás.

Glossary – a dokumentum szójegyzékét tartalmazó dokumentumra mutat.

Copyright – az adott dokumentumra vonatkozó jogvédelmi információt tartalmazó dokumentumra mutat.

Chapter – egy fejezetre utaló hivatkozás.

Section – egy bizonyos részletre mutató hivatkozás.

Subsection – egy alrészletre mutató hivatkozás.

Appendix – a dokumentum hozzáfűzéseit tartalmazó dokumentumra mutat.

Help – segítséget nyújtó dokumentumra mutat.

Bookmark – egy könyvjelzőre mutató hivatkozás. Lehetnek például kulcspontok egy nagyobb dokumentumon belül.

4.2.4 A HTML dokumentum alapszerkezete, alapvető elemei

A HTML dokumentum egy ASCII karakterekből álló olyan szöveges fájl, amit egy `<html>` és egy `</html>` tag ölel körül, és amelyet a böngésző programok értelmezni képesek. [4]

A HTML dokumentum az alábbi három szerkezeti egységre bontható:

1. A DOCTYPE meghatározás, amely arról informálja a böngészőprogramot, hogy a HTML nyelv melyik verziójában íródott a dokumentum.
2. Fejléc elem (`<head> ... </head>`), amely az egész dokumentumra vonatkozó információkat tartalmaz. Paraméter értékeként megadható az a külső fájl, amelyben a meta tagok találhatóak, amely azzal az előnnyel jár, hogy nem kell minden dokumentumban megadnunk a meta sorokat. A fejléc elembe ágyazva helyezzük el a `<title>Cím</title>` elemet, és ennek tartalma jelenik meg a böngészőben az ablak címként.
3. A dokumentum törzsébe (test elemébe) a `<body> ... </body>` elemek közé kerül az a tartalom, amelyet szeretnénk megjeleníteni a böngészőben. Ez tartalmazza a

dokumentumnak a böngészőben látható részét. A törzs elem általános felépítése a paraméterei jelölésével a következő (a {} jeleket az alkalmazás során elhagyjuk):

```
<body>{alink, background, bgcolor, link, onload, onunload, text, vlink}>...</body>
```

A <body> elem egyes paramétereinek értelmezése:

alink – az éppen aktív hivatkozások (link-ek) színe rákattintás esetén

background – egy háttérképet rendelhetünk az oldalhoz a kép URL-je megadásával

bgcolor – az egész oldal háttérszíne állítható be vele

link – a még meg nem látogatott linkek színe

onload – a dokumentum betöltődésekor végrehajtódó szkriptparancs (paraméter érték)

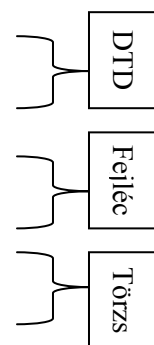
onunload – a dokumentum bezárásakor végrehajtódó szkriptparancs (paraméter érték)

text – az oldalon megjelenő szövegek színét állítja be

vlink – a már meglátogatott linkek színét állítja be

Egy egyszerű HTML-dokumentum a következő, melynek szerkezeti egységei külön jelölésre kerültek:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Minta oldal</title>
  </head>
  <body>
    <h1>Itt szerepelhet a szöveg</h1>
  </body>
</html>
```



4.2.5 A HTML szövegformázó elemei

Az információ egyik fő hordozója a HTML dokumentumokban a szöveg. Szöveges dokumentumok esetében különösen fontos jelentőséggel bír annak formája. A HTML dokumentumokban is van lehetőség a szövegszerkesztőkben megszokott szövegformázásra. A szövegformázásnak három területét kell kiemelnünk: a karakterformázást, a sor- és bekezdésformázást, valamint a szakaszolást.

A **fontok** beállítására a tag-et használjuk, melynek általános felépítése a következő: Az attribútumok formája és értelmezése: **size="méret"** – a font nagyságát állítjuk be az alábbiak szerint:

- = abszolút módon – ekkor 1-7 közötti szám lehet a méret, pl. `<font size="3">`
- = relatív módon – megadhatjuk, hogy az alapértékhez képest mennyivel nagyobb legyen, pl. `<font size="+1">`, `<font size="-2">`

`basefont size="méret"` – az alap fontméret (alapérték) beállítására szolgál

`color="színkód"` – a font színét határozza meg (a HTML nyelvben a szín hexadecimális kóddal, pl. `"#0000FF"` vagy definiált névvel, pl. `"blue"` adható meg)

`face="név"` – a font típusát (betűcsaládot) adja meg (pl. Arial, Verdana stb.)

Nem ajánlott a különleges egyedi betűtípusok használata, mert ha nincs az adott gépre felinstallálva, akkor nem fog megjelenni.

A **karakterek** formázására további elemek is használhatók, ezek közül a jelentősebbek:

`<b>...</b>` – félkövér (bold) karakter

`<blink>...</blink>` – villogó szöveg

`<i>...</i>` – dőlt (italic) karakter

`<s>...</s>` – áthúzott karakter

`<strike>...</strike>` – aláhúzott karakter

`<small>...</small>` – kisméretű betűformátum

`<sub>...</sub>` – alsó index

`<sup>...</sup>` – felső index

`<u>...</u>` – aláhúzott (underlined) karakter

Az elemek kombinált használatát mutatja be a következő példa:

`<b><i><u>Ez egy félkövér dőlt aláhúzott szöveg</u></i></b>`

A `<P>...</P>` elem segítségével a szöveget **bekezdésekre** (paragrafusokra) oszthatjuk.

A böngészők a bekezdések előtt és után nagyobb térközt hagynak. A tag általános alakja: `<p{align}>`. Az `align` (igazítás) paraméter értékei lehetnek: `left` – balra, `right` – jobbra, `center` – középre, `justify` – sorkizárt.

A `<br{clear}>` tag segítségével **sortörést** kényszeríthetünk ki. Az elemet nem kell lezárni (üres elem). A böngésző a `<br>` elemhez érve az értelmezéskor az utána következő szöveget új sorban a bal margónál kezdi. Például, ha egy postai címet akarunk megadni, a következők szerint tehetjük:

```
<p>Naár János<br>
Herceghút<br>
Petőfi Sándor utca 33.<br>
3958</p>
```

Amennyiben tördelhetetlen szóközt igényel a szöveg, akkor az egyszerű szóköz helyett alkalmazzuk a ` `; különleges karaktert. Ilyen lehet pl. egy intézménynév, mondjuk a "Kazinczy Ferenc Általános Iskola", kódolva:

`Kazinczy Ferenc Általános Iskola`

A szöveg felosztható több logikus (logikai egységet képező) **szakaszra**, amelyeket be kell vezetni megfelelő címsorral. Hat címsor szint definiálható: `h1`, `h2`, `h3`, `h4`, `h5` és `h6` (a magasabb szintűtől az alacsonyabbig). Általában a `h1` az egész lap címsora szokott lenni, a `h2` szakaszokra bontja lapot, a `h3` alszakaszokra és így tovább. A `h1` szintű fejléc jelenik meg a legnagyobb, a `h6` pedig a legkisebb betűmérettel. A lap ilyen fajta szakaszolása érthetőbbé teszi a lapot a képernyőolvasók és az indexelő motorok (pl. Google) számára. A szakaszolás általános alakja: `<hx{align}>`, amelyben x egy szám 1 és 6 között. Az `align` paraméter lehetséges értékei megegyeznek a `<p>` tagnél leírtakkal.

Példa a címsorok összerendezésére:

```
<h1>HTML adattípusok</h1>
<h2>Szöveges adat</h2>
<h2>URL</h2>
<h2>Szám adatok</h2>
  <h3>Abszolút szám adat</h3>
  <h3>Relatív szám adat</h3>
<h2>Egyéb adatok</h2>
```

A webes dokumentumokban **idézhetünk** más dokumentumokból. A HTML kódban a hosszabb idézeteket a `blockquote` elemmel jelölhetjük. A `blockquote` elem nem tartalmazhat szöveget, hanem csak egy másik blokk szintű elem lehet benne. Az idézetben is ugyanazokat a blokk szintű elemeket kell használni, mint az eredeti környezetben. Egy bekezdés idézése esetén, a paragrafus elemet kell használni, egy lista elemeinek idézésekor, listát kell készíteni, és így tovább.

Amennyiben az idézetet egy másik weboldalról származtatjuk, akkor megadható a forrás (vagyis az idézendő dokumentum helye) a `cite` attribútumban az alábbi példa szerint:

```
<p>Ez a saját szövegünk első része, amely után idézünk</p>
<blockquote cite="http://www.keri-ujhely.hu/aktualis/">
<p>A szövegünk folytatása az idézet után</p>
</blockquote>
```

Előre **formázott szöveget** is el lehet helyezni a dokumentumban a `<pre{width}>...</pre>` elem segítségével. Ekkor a szöveg pontosan olyan tagolásban (sortöréssel, szóközökkel) jelenik majd meg, mint ahogyan begépeztük. Az előformázott szövegek megjelenítése általában azonos szélességű betűket tartalmazó betűtípussal történik.

Vonalak létrehozására a `<hr {width, size, color, align}>...</hr>` elemet használhatjuk. A `width` és a `size` attribútummal a vonal hossza (pixel vagy %) és vastagsága (pixel) adható meg. A `color` paraméter a vonal színét, az `align` pedig az igazítását határozza meg.

A szövegszerkesztők felsorolás és számozás bekezdés formázási elemeihez hasonlóan a HTML nyelvben is kétféle **listát** tudunk létrehozni: az egyszerű és a sorszámozott listát. Az egyszerű lista akkor használható, ha a listaelemek sorszámozása nem indokolt. Ennek az általános alakja: `<ul {compact, type}>...</ul>`. A `type` attribútummal a felsorolási jel állítható be, melynek értékei "square" (■), "circle" (○) és az alapértelmezett "disc" (●). A felsorolás jel mögött a listaelemeket a `<li {type, value}>...</li>` elemmel tudjuk elhelyezni. A számozott listában a listaelemek sorszámot kapnak, amely a következő elemmel írható le: `<ol {compact, start, type}>...</ol>`. A `type` attribútummal megadható a számozás típusa, pl. `<ol type="a" >...</ol>`, ekkor a listaelemek a, b, c stb. sorszámot kapnak (az alapértelmezett lista számozása 1, 2, 3 stb.).

A következő példa egy lista kódját és megjelenését tartalmazza:

```
<body>
Iskola
<ol><li>Technikai dolgozók</li>
  <li>Nevelők
    <ol type="a">
      <li>Osztályfőnökök</li>
      <li>Szaktanárok</li>
    </ol>
  </li>
  <li>Tanulók
    <ol type="a">
      <li>Alsó tagozatos</li>
      <li>Felső tagozatos</li>
    </ol>
  </li>
</ol>
</body>
```

Iskola

1. Technikai dolgozók
2. Nevelők
 - a. Osztályfőnökök
 - b. Szaktanárok
3. Tanulók
 - a. Alsó tagozatos
 - b. Felső tagozatos

4.2.6 Képek beágyazása a HTML dokumentumba

A HTML formátumú dokumentumba képeket (grafikákat) is elhelyezhetünk. A képek beágyazására az IMG elemet használjuk, melynek általános alakja a következő:

`<img {align, alt, border, height, hspace, ismap, src, units, usemap, vspace, width}>`.

A finombeállításra használható attribútumok funkciója a következő: `align` – a kép igazítása (lehetséges értékei: top, middle, bottom, left, right), `alt` – azt az alternatív szöveget adja meg, amely nem grafikus böngészők esetén a kép helyett megjelenik, `border` – a képkeret képpontokban jelzett vastagsága adható meg (0 esetén nincs keret), `height` és `width` – a kép magasságát és szélességét adja meg a `units` által meghatározott egységben (pixel vagy %), `hspace` és `vspace` – a vízszintes és a függőleges térköz megadására használhatók, `ismap` és `usemap` – a kép különböző területeihez különböző hipertext hivatkozásokat rendelhetünk, `src` – a képfájl helyét és azonosítóját (URL-jét) adja meg. A képfájl beágyazása többféle forrásból lehetséges és emiatt a `src` paraméter értéke is más-más módon adható meg:

- a HTML dokumentumot tartalmazó könyvtárban, ekkor elég a fájl azonosítót megadni, pl. `kep1.jpg`
- egy másik könyvtárban, ekkor az útvonalat is meg kell adni, pl. `images/kep1.jpg`
- egy másik webcímen, akkor a teljes http címet megadjuk, pl. `http://www.hercegkut.hu/kepek/cimer.gif`

A kép és a szöveg viszonya is beállítható a dokumentumokban, nevezetesen az, hogy a szöveg körülfolatása milyen legyen. A kép helyzete az `align` attribútummal igazítható a bal vagy a jobb oldali margóhoz. Az alábbi megadással a kép az oldal bal oldalára igazodik, a szöveg pedig a kép jobb oldalán olvasható.

```
<p> Ez a szöveg a kép jobb oldalán olvasható.</p>
```

Azt is kikényszeríthetjük, hogy a szöveg a kép alatt folytatódjon, például:

```
<p> Ez a szöveg a kép jobb oldalán olvasható.
<br clear="all"> A szöveg új sorban folytatódik a kép alatt.</p>
```

4.2.7 Táblázatok a HTML dokumentumban

A táblázatok numerikus és/vagy szöveges információkat hordozhatnak. Az oldalaink rendezett megjelenésében is sokat segíthetnek a táblázatok, mivel a böngészők a cellákat mindig pontosan egymás mellé és alá helyezik. Jól használhatók többek között az oldal tartalmának rendezéséhez, menüsorok megjelenítéséhez, nagyobb darabolt kép összeillesztéséhez. A táblázatok oszlopokba és sorokba rendezett cellákból épülnek fel. A cellák tartalmazhatnak szöveget, képet vagy akár új táblázatot is. A táblázatok méretezhetők

és formázhatók. A HTML dokumentumokban a `<table>` és a `</table>` tag-ek közé zárt részt tekintjük egy táblázatnak. A táblázat elem általános alakja:

```
<table{align, bgcolor, border, cellpadding, cellspacing, summary, width}>...</table>
```

Az attribútumok a táblázat egészére vonatkozó paramétereket adják meg, melyek a következők: `align` – vízszintesen igazítja a cellák tartalmát (lehetséges értékei: left, center, right, justify, decimal), `bgcolor` – a háttérszint adja meg, `border` – a rácszat szélességét határozza meg (0 esetén nincs keret), `cellpadding` – pixelben meghatározza az üres hely nagyságát a cellákon belül az adatok körül (az adat és a cella széle közötti távolság), `cellspacing` – pixelben meghatározza a cellák közötti üres hely nagyságát, `summary` – szövegesen leírjuk a táblázat struktúráját az olyanoknak, akik nem látják a táblázatot, `width` – a táblázat szélességét adjuk meg (pixel vagy %).

A táblázat **sorait** a `<tr{align, bgcolor, height, valign}>...</tr>` elemmel hozzuk létre, azok konténereként viselkedik. Az eddig még nem ismertetett attribútumok funkciója: `height` – a sor magasságát állítja be (pixel vagy % adható meg), `valign` – a sor tartalmának függőleges igazítását állítja be (top – tetejéhez, bottom – aljához, center – középre, baseline – alapvonalhoz).

A `tr` elemek között kell a táblázat **celláit** meghatározni a `td` elemmel a következő módon: `<td{align, bgcolor, colspan, height, nowrap, rowspan, width, valign}>...</td>`.

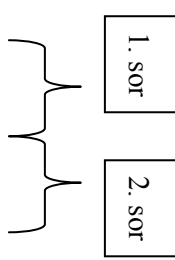
Az új attribútumokkal a következők állíthatók be: `colspan` – adott sorban összevont cellák száma, `nowrap` – sortörés tiltása a cellában, `rowspan` – az adott oszlopban összevont cellák száma.

A táblázatnak **címet** adhatunk, amely alapértelmezetten a táblázat fölött jelenik meg középre igazítottan. Közvetlenül a nyitó `<table>` elem után és az első `tr` elem előtt kell megadni a következő módon: `<caption{align}>...</caption>`. Az align attribútummal a cím jobbra (right), balra (left), fölé (top) vagy alá (bottom) igazítható.

Fejléc létrehozására a `th` elemmel történik, amely a táblázat első adatsora fölött helyezkedik el, melyben a szövegek félkövér karakterekkel jelenik meg. Az elem általános alakja: `<th{attribútumai azonosak a td elemével}>...</th>`.

Az ismertetett elemek egymásba ágyazása egy 2 sor $\times$ 1 oszlop méretű HTML táblázat esetén a programtörzs a következő lehet:

```
<body>
  <table>
    <tr>
      <td>...</td>
    </tr>
    <tr>
      <td>...</td>
    </tr>
  </table>
</body>
```



Az ismertetett elemeken túl még továbbiak is használhatók a táblázatok leírására, melyek közül a fontosabbak a következők: `thead` – a fejlécsorok csoportját határozza meg, `tfoot` – a lábléc-sorok csoportját definiálja, `tbody` – a táblázat egy részére (törzsére) vonatkozó formázási utasítás, `col` – a táblázat egy oszlopát adhatjuk meg, `colgroup` – a táblázat több oszlopát hozza létre.

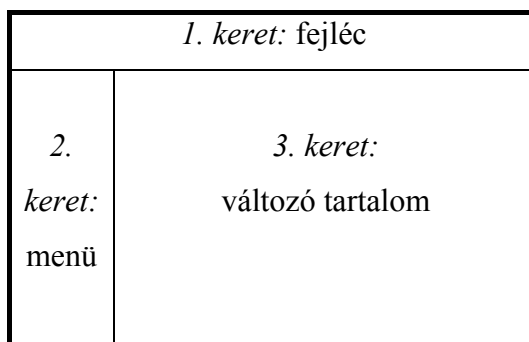
A következő példa egy 2 sor $\times$ 2 oszlop méretű címmel (táblázat fölött középen) és fejléccel (középre igazított szöveggel) ellátott formázott (az 1. sor adatai jobbra, a 2. soré balra igazítottak) táblázat (piros 1 pontos vonalakkal szegélyezett, 10 pontos belső cellamargóval) kódját tartalmazza:

```
<table border="1" bordercolor="red" cellpadding="10" width="80%">
<caption>Ez a táblázat címe</caption>
  <tr align="center">
    <th>1. oszlop</th>
    <th>2. oszlop</th>
  </tr>
  <tr align="right">
    <td>1.sor 1.cellája</td>
    <td>1.sor 2.cellája</td>
  </tr>
  <tr align="left">
    <td>2.sor 1.cellája </td>
    <td>2.sor 2.cellája </td>
  </tr>
</table>
```

4.2.8 HTML keretek – a dokumentumablak felosztása

Olyan esetekben, ha egy weboldalnak csak bizonyos részei változnak, míg a többi része változatlan marad, az oldalt célszerű ún. keretekre (régiókra) felosztani. Ekkor a

böngészőnek csak a változó ablakot kell újra tölteni, miközben a többi változatlan marad. A weboldalnak egy gyakori keretes ablakszerkezete látható a következő ábrán:



3. ábra: Egy weboldal tipikus keretszerkezete

Az ilyen weblapok több fájlból állnak. Keretenként léteznie kell egy-egy HTML dokumentumnak, amely az adott ablakot írja le, továbbá még egy HTML fájlnak, amelyben azt adjuk meg, hogy az egyes keretek hol és milyen méretben jelenjenek meg az oldalon. Ez utóbbi fájlban használjuk a `<frameset{cols, onload, rows}>...</frameset>` elemet. Ennek a fájlban a `<body>` elem helyett használjuk, mivel ennek az oldalnak nincs megjelenítendő tartalma. A `cols` és a `rows` attribútumok az egyes keretek függőleges és vízszintes méretét (pixelben vagy %-ban) határozzák meg. A maradék képernyő területre *-gal lehet hivatkozni. Az `onload` paraméter esetén végrehajt egy megadott JavaScript kódot, amikor a böngésző befejezi egy ablak, vagy egy `frameset` összes elemének betöltését. Egy `<frameset>` elemmel csak vagy függőlegesen vagy vízszintesen osztható fel a képernyő. Mindkét irányú felosztás igénye esetén (mint az előző ábrán is látható) a `<frameset>` elemeket egymásba kell ágyazni.

A definiált keretbe a `frame` elemmel töltjük be a megfelelő objektumot (egy teljes HTML fájlt vagy annak egy részét, képet) és határozzuk meg annak viselkedését. Egy `frameset` elem annyi `frame` elemet tartalmaz, ahány részre osztottuk az oldalt. Az elem formája: `<frame{name, scr, scrolling, marginwidth, marginheight}>...</frame>`. Az opciók funkciója: `name` – a keretnek nevet ad (a fejlécben megadott `<base target="név">` utasítás a `name="név"` opcióval elnevezett keretbe irányítja a hivatkozásokat), `scr` – betölti a megadott objektumot, `scrolling` – a keret szkrollozását szabályozza (no – letiltja, yes – engedélyezi), a `marginwidth` és `marginheight` – a kereten belüli margókat állítja be.

A `<frameset>`, `</frameset>` tagekkel határolt területnek meg kell előznie a dokumentumtörzset. Továbbá egy `<noframes>` elemmel kell jelezni a dokumentum azon részének kezdetét, amelyet akkor jelenít meg a böngésző, ha nem kezeli a keretutasításokat.

A 3. ábrán látható ablakkereteket az alábbi kód írja le:

```
<html>
  <frameset rows=150,*>
    <frameset cols=150>
      <frame src=logo.jpg scrolling=no name="fejlec">
    </frameset>
    <frameset cols=25%,*>
      <frame src=menu.html name="menu">
      <frame src=tartalom.htm name="tartalom" marginheight=10
        marginwidth=10>
    </frameset>
  </frameset>

  <noframes>
    <center>
    <body bgcolor="#FFFF00">
    <font color="#FF3333">
      <h1>Sajnos az Ön böngészője nem támogatja a keretek
        használatát!
      </h1>
    </font>
    </body>
  </noframes>
</html>
```

4.2.9 Hiperhivatkozások a HTML dokumentumban

A hiperhivatkozások (linkek és könyvjelzők) a weboldalak alapvető részei, a HTML formátum lényegét (hypertext) jelentik. Segítségükkel tudunk a világhálón található dokumentumok között mozogni, illetve a dokumentumok adott pontjaira ugrani. A dokumentum bármely részét hivatkozásokat helyezhetünk el, melyeket aktivizálva, a hivatkozott szöveg(rész)hez juthatunk el. A legegyszerűbb esetnek azt tekintjük, amikor a hivatkozás egy fájl távolabbi részére – egy könyvjelzőhöz – mozdítja a böngésző ablakot. A link mindig egy másik fájlra mutat. A linknek két fajtáját különböztetjük meg:

- abszolút link: a megcélzott fájlt teljes elérési útvonalával adjuk meg,
pl. <http://www.kazinczyoktatasisimuhely.hu/>, `C:/Dokumentumok/weblap`
- relatív link: a hivatkozó oldaltól kiindulva adjuk meg az elérési utat a DOS szabályai szerint, pl. `/hirek/20081020/012hir`

A hiperhivatkozások konkrét formája sokféle lehet a célobjektumtól függően. A létrehozására használható elem általános alakja a következő:

`<a{accesskey, charset, cords, href, hreflang, name, onblur, onfocus, rel, rev, shape, tabindex, target, type}>...</a>`. Az `<a` és `/a>` tagek közrezárhatnak szöveget, szövegrészt, képet stb, melyet a böngésző program eltérően jelenít meg (más színnel, aláhúzással stb.). Amint egérrel a közrezárt rész fölé érünk, annak mutatója megváltozik.

Az elem fontosabb attribútumainak jelentése a következő: `charset` – a hivatkozott tartalom karakterkódolása, `hreflang` – a hivatkozott tartalom nyelve, `name` – a könyvjelző hivatkozáscíme, `rel` – kapcsolat a cél felé, `rev` – kapcsolat a cél felől, `target` – céldokumentum helye a keretben (lehetséges értékei: `_blank`: új böngészőablakban, `_top`: azonos ablakban keretek nélkül, `_parent`: a fremaeset-ben megadott szülőablak helyén, `_self`: aktuális keretben), `type` – a hivatkozott tartalom médiatípusa. Lehetőség van egy kép egészének vagy részeinek a kijelölésére linkfelületként, amelynek beállítására valók a `chords`, `shape`, `onfocus`, `onblur` attribútumok.

A legfontosabb attribútum a `href` (hypertext reference), amely a hivatkozás címét tartalmazza. Formája a következő: `<a href="protokoll://elérési út/fájlnév.kit"></a>`. A következő példák közül az elsőben az „Petőfi Sándor Tagintézmény” szövegre, a másodikban az „ikon.jpg” képre kattintva linkelhetünk a megjelölt oldalakra. Az utóbbi példa relatív hivatkozást tartalmaz.

```
<a href="http://kazinczyfaltisk.extra.hu/petofi.html">Petőfi Sándor  
Tagintézmény</a>  
<a href="/zene.html">  </a>
```

A dokumentum (oldal) adott pontjára való hivatkozásnak előfeltétele, hogy a kellően szakaszolt dokumentumban könyvjelzőket helyezünk el. A könyvjelzők elhelyezése a `name` attribútummal lehetséges. Ez azonosítja a hivatkozási célt. A következő példa ezt mutatja be:

```
<a name="1_resz">1. Az optikai tárolók</a>
```

A könyvjelzőkkel ellátott oldal egyes pontjaira a `<a href` alakkal hivatkozhatunk. Erre példa:

```
<a href="#1_resz">Ugrás az 1. Az optikai tárolók c. részre</a>
```

A következő példa azt mutatja be, hogy szakaszolt szöveg esetében hogyan oldható meg a szövegen belüli hivatkozás.

A következő példa azt mutatja be, hogy szakaszolt, címekkel és tartalomjegyzékkel ellátott szöveg esetében mennyivel egyszerűbb a szövegen belüli hivatkozás.

```
<h1><a name="egysejtuek">Egysejtűek</a></h1>  
<h2><a name="rovarok">Rovarok</a></h2>  
...
```

```
<ul>
  <li><a href="#egysejtuek">Egysejtűek</a></li>
  <li><a href="#rovarok">Rovarok</a></li>
  ...
</ul>
```

A címsorokat egy-egy hiperhivatkozás potenciális céljává úgy tehetjük, hogy tartalmukat az `<a name >` és `</a>` közé zárjuk. A tartalomjegyzék most már tartalmazhat olyan hiperhivatkozást, mely ezt a nevet (egysejtuek, rovarok) használja. Ha a cél egy másik oldalon van, akkor ennek az oldalnak a webcíme meg kell hogy előzze a # karaktert, pl.

```
<a href="http://www.kazinczyisk.hu/#rovarok">Rovarok</a>
```

A hivatkozás speciális fajtája, amikor az `<a href` alakkal e-mailt küldhetünk az oldalról a következő példa szerint:

```
<a href="mailto:jnaar@freemail.hu">Levél küldése</a>
```

Az elem értelmezésekor a böngésző megnyitja az alapértelmezett levelező programot és egy jnaar@freemail.hu-ra megcímezett üres e-mail ablakot kínál fel. A levél többi részét is megadhatjuk (tárgy, a szöveg egy részét stb.), sőt több címet is megjelölhetünk.

4.2.10 Űrlapok a HTML dokumentumban

Az űrlapokkal interaktívvá tehetjük a weblapot azzal, hogy a felhasználó adatokat vihet általuk be és ezek az adatok továbbküldhetők, hogy a szerveren egy nem HTML-alapú program (PHP, CGI, ASP stb.) feldolgozza azokat, vagy e-mailben a címzettnek megküldhetők. Lehetőség van arra is, hogy az oldalon lévő egyéb program (általában JavaScript nyelvű) dolgozza fel a bevitt adatokat.

Az űrlap alkotórészeit (szövegablak, adat- és jelszóbeviteli mező, gombok, választógomb csoportok stb.) és szerkezetét, amelyekből többet is használhatunk egy űrlapon, valamint azokat a folyamatokat, amelyek vele kapcsolatosak le kell írni. Az űrlappal kapcsolatos elemeket a `<form>` és `</form>` elemek között kell elhelyezni. A keretként funkcionáló elem általános alakja a következő:

```
<form{action, enctype, method, name, onreset, onsubmit, target}>...</form>
```

Az attribútumok funkciója a következő: `action` – az itt megjelölt címre küldi az űrlapot (szerver oldali program vagy „mailto:” jelzettel egy e-mail cím), `enctype` – a küldéskor használt tartalomformátumot állítja be (alapértelmezés: „application/x-www-form-

`urlencoded`”, `method` – a küldés módját adja meg (alapértelmezett értéke: „get”, e-mail esetén „post”), e-mail esetén: „text/plain”), `onreset` és `onsubmit` – a törléskor, illetve küldéskor indítandó scripteket adja meg, `target` – az űrlapra adott válasz kerete (amelyikre hatással lesz).

Az űrlapba számos elem közül válogathatunk, melyek főbb csoportjai: a adatbeviteli mezők, többsoros szövegbeviteli mezők, kapcsolók és nyomógombok. Az űrlapok elemeit az `<input{type}>...</input>` elemmel írhatjuk le. A `type` attribútum értékétől függ, hogy milyen elem kerül beillesztése az űrlapba. Ezek a következők lehetnek:

– adatbeviteli mezők:

`text` – egysoros szöveg bevitele (a `size` opció határozza meg az ablak szélességét, a `maxlength` pedig a bevihető szöveg maximális hosszát)

`password` – jelszó bevitele (nem jelenik meg a bevitelkor)

`hidden` – rejtett szöveg bevitele (nem jelenik meg a bevitelkor)

`file` – fájl feltöltése (a megadott fájl az `accept` kiegészítő opcióval megadott MIME módon csatolódik az elküldendő kérdőívhez)

– kapcsolók:

`checkbox` – kijelölő kapcsoló (több is kiválasztható egyszerre, a `checked` opció bekapcsolja)

`radio` – választókapcsoló vagy rádiógomb (egyszerre csak egyet lehet kiválasztani, a `checked` opció bekapcsolja)

– nyomógombok:

`submit` – adattovábbító gomb (a `value` opcióval feliratot helyezhetünk a gombra)

`reset` – adatok törlése (az adatbeviteli mezőket alapértékkel tölti fel, a `value` opcióval feliratot helyezhetünk a gombra)

`button` – egyéb nyomógomb

`image` – képből készített nyomógomb (kiegészítő opciója az `src`, melynek a kép elérési útvonala az értéke)

Hosszabb, többsoros szöveg bevitelét írja le az alábbi elem:

`<textarea{name, rows, cols, value}>...</textarea>`.

Az elem beviteli ablakot nyit `rows` magasságban és `cols` szélességben, a `value` az alapértelmezett szöveget adja meg, a `name` értékeként megadott néven azonosítja a mezőbe bevitt adatokat.

Kérdésre adandó válasz menüből történő kiválasztását teszi lehetővé a `<select name="név" size="sor">...</select>` elemmel létrehozott szerkezet, melynek menüpontjai az `<option>` elemmel adhatók meg. A `size` opció meghatározza, hogy hány sorban jelenjenek meg a választéklista. Megadásával szkrollozható, elhagyva legördülő menüt kapunk. A `multiple` opciót használva több menüpont jelölhető ki egyszerre. Az alapértelmezett választást az `<option selected>` formájú szerkezet adja meg.

A következő példa egy űrlapot és annak kódját tartalmazza:

- Levélírás Naár Jánosnak -

Név:	név
e-mail:	e-mail cím
honnan:	---- innen ---- ▼
Honnan2:	☐ innen ☐ onnan
szöveg:	
Elküld Töröl	

```
<form name="mail_form" action="mailto:nevem@domainom.com"
method="post" enctype="text/plain">
<input type="hidden" size="0" name="web-üzenet" />
<table width="250" bgcolor="#ffface" bordercolor="#ff0000"
border="2" align="center">
  <caption><b>- Levélírás Naár Jánosnak -</b></caption>
  <tr>
    <td>Név:</td>
    <td><input name="név" type="text" size="20" value="név" /></td>
  </tr>
  <tr>
    <td>e-mail:</td>
    <td><input name="e-mail" type="text" size="20" value="e-mail cím"
/></td>
  </tr>
  <tr>
    <td>honnan:</td>
    <td>
<select name="Honnan:">
  <option name="innen"> ---- innen ---- </option>
  <option name="onnan">onnan</option>
</select>
    </td>
  </tr>
  <tr>
    <td>szöveg:</td>
    <td>
<div style="border: 1px solid black; height: 60px; width: 100%;>
  </div>
    </td>
  </tr>
  <tr>
    <td colspan="2" style="text-align: center;">
<input type="button" value="Elküld">
<input type="button" value="Töröl">
    </td>
  </tr>
</table>
```

```

</tr>
<tr>
  <td>Honnan2:</td>
  <td>
    <input type="radio" name="honnan" value="innen">innen<br />
    <input type="radio" name="honnan" value="onnan">onnan<br />
  </td>
</tr>
<tr>
  <td>szöveg:</td>
  <td>
    <textarea name="Szöveg" rows="5" cols="30"
value="szöveg"></textarea>
  </td>
</tr>
<tr>
  <td> </td>
  <td>
    <input type="submit" value="Elküld" />
    <input type="reset" value="Töröl" />
  </td>
</tr>
</table>
</form>

```

4.2.11 A HTML dokumentumok egyéb elemei

Bármilyen MIME-típusú objektumot elhelyezhetünk az oldalon, amit a webböngésző képes értelmezni, például beágyazott oldal, plugin által kezelt kód (pl. Flash animáció, Java applet, hanganyag stb.). Objektum elhelyezése a következő formában történhet, melynek típusát a type jellemzővel adjuk meg.

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

A következő elem csak az object elemen belül jelenhet meg.

```
<param>...</param>
```

A name és a value jellemzők használatával, minden egyes `<param/>` elemmel az objektum egyes paramétereit tudjuk megadni, pl.: width, height, font, background colour stb., attól függően, hogy a beágyazott object elemben mit engedélyeztek a fejlesztők.

Az `<embed>...</embed>` elempár egy plugin-kezelőt hív meg a type jellemzőben megadott típushoz. Ezt használhatjuk Flash és hangfájlok és további objektumok beágyazásánál. Ezt a HTML elemet a Netscape vezette be. A W3C által javasolt standard módszer a `<object>` elem használata.

A `<noembed>...</noembed>` elemmel adhatjuk meg, hogy mi jelenjen meg abban az esetben, ha a beágyazott objektum nem működik.

Az `<applet>...</applet>` utasítás Java appletet helyez el a weboldalon, de helyette inkább az `<object>` használata javasolt.

A kódban elhelyezett megjegyzések (kommentek) általában saját magunk segítségére valók. Jól el tudjuk különíteni velük a HTML kódot, és általuk jobban áttekinthetővé tehetjük a kódszerkezeteket. Néhány példa kommentere:

```
<!-- Megjegyzés 1 -->
<!-- Megjegyzés 2 -- -- Megjegyzés 3-->
<!-->
<!--> Megjegyzés 4 -->
<!-->
```

Az alábbi pedig egy jegyzet megjelenítése:

```
<note>Ez itt egy jegyzet</note>
```

Egy HTML formátumú szövegfájl a tartalmazhat lábjegyzeteket. Az `<fn id="azonosító">` és a `</fn>` utasítások között szerepel a lábjegyzet szövege. Az így definiált lábjegyzetszövegre hivatkozik a szövegnek az `<a href="#azonosító">` és az `</a>` utasításokkal jelölt része.

Példa: ehhez a sorhoz `<a href="#az">lábjegyzet</a>` tartozik.

```
<fn id="az">A jelzéshez tartozó lábjegyzet</fn>
```

4.3. XML

Az XML az eXtensible Markup Language kifejezésből alkotott mozaikszó, magyarul bővíthető jelölő nyelvet jelent. A W3C által ajánlott **általános célú leíró nyelv**, speciális célú leíró nyelvek létrehozására.

Az SGML alapvetően egy eszköz volt saját markup nyelvek definiálására. Terjedni kezdett, de széleskörű felhasználásra túl bonyolult volt. A generikus kódolásnak a HTML definiálása adott lendületet. Egyszerűbb volt használni, de visszalépést is jelentett, mivel sok, formázó taget alkalmazott, ezáltal pont a strukturáltság veszett el, és nem lehet saját tageket definiálni. A webnek köszönhetően azonban hamar elterjedt.

Az eredeti generikus alapokhoz való visszatérés jegyében jött létre az XML. Olyan nyelvet akartak létrehozni, amely lehetővé teszi markup nyelv definiálását kifejezetten a

használni kívánt adatokhoz, egyértelműen interpretálható, egyszerű, a lehető legtöbb hiba ellenőrzést biztosítja, és elválasztja a megjelenítést a leírástól. Az XML lényegében az SGML leszűkítése a sokak számára felesleges részek elhagyásával. A HTML ezzel szemben egy SGML-ben definiált markup nyelv. [1]

Az SGML-lel ellentétben nem szabvány, hanem ajánlás. Az ajánlás egy olyan szintaxist ad meg, amelyet betartva különböző jelölő nyelvek (pl. XHTML) hozhatóak létre. Ezért metanyelvnek, nyelvleíró nyelvnek is nevezik. Az XML specifikáció azt definiálja, hogy milyen módon írható le egységesen a dokumentum. Az XML nyelvet nagyobb mennyiségű adat elektronikus továbbítására fejlesztették ki. Az W3C SGML munkacsoportja által kidolgozott XML 1.0 ajánlás 1998-ban született meg. A kidolgozás során az SGML-t tekintették alapnak, megfogalmaztak néhány egyszerűsítő megszorítást, és követelményként kezelték, hogy az XML dokumentum SGML konform legyen.

4.3.1. Az XML jellemzői

Az SGML és az XML hasonlítanak egymásra, mindkét nyelv célja az információcsere és információleírás platformfüggetlen megvalósítása. Az SGML-ben sokkal több nyelvi elem megengedett, mint az XML-ben, például nincs különbség a kis- és nagybetűk között, az attribútumokat nem kell mindig idézőjelbe tenni, az elemeket nem kell mindig lezárni. Az XML olyan nyelv, melyben a jelölésre használt címkék szabadon definiálhatók és szemantikai értelmet adnak a dokumentumban foglaltaknak. Az SGML egyszerűsített részhalmaza, mely különböző adattípusok leírására képes. Strukturált információkat tartalmazó dokumentumok számára kifejlesztett leíró nyelv. A strukturált információk egyrészt valós tartalmat (numerikus, szöveg, kép) másrészt információkat jelentenek a tartalom struktúrájáról (például, hogy az adott helyen levő szöveg fejléc, cím vagy dátum stb.). Az adatelemek közötti kapcsolatokat is képes leírni. Például egy cím irányítószámot, településnevet, utcanevet és házszámot tartalmaz. Az XML a célrendszer számára nem csak azt adja meg, hogy az adatfolyamban irányítószám, településnév stb. található, hanem azt is jelölje, hogy mindezek az elemek együtt egy címet alkotnak.

Sok XML alapú vagy XML dokumentumot használó alkalmazást fejlesztenek. Az XML kapcsán a "dokumentum" szó mögött nem a hagyományos értelemben vett

dokumentumot kell érteni, hanem más XML adatformátumok sokaságát (vektorgrafika, matematikai egyenlet, objektum metaadat stb.) [1]

A XML jelentős változást jelent, mivel túllép a jelölő nyelvek korábbi problémáin: az SGML túl komplex, a HTML megjelenítés orientált. A HTML-ben mind a tagek jelentése, mind a tagek halmaza fix és rögzített. Az XML sem szemantikát, sem a tagek halmazát nem határozza meg. Az XML lehetővé teszi, hogy tageket definiáljunk és meghatározzuk a strukturális viszonyait. Az XML dokumentumnak a szemantikáját a feldolgozó alkalmazás által vagy stíluslapként már korábban definiálni kell. [8]

Megoldást kínál a webes kereskedelem számára is, mert segítségével lehetővé válik, hogy a megrendelő és az áruház között a megrendelések forgalmát bonyolítva értelmezni lehessen például, hogy mely szám jelöli a megrendelés azonosítóját és melyik a dátumát. DTD létrehozásával leírhatjuk a megrendelést feldolgozó adatelemeinket, és egy kulcsszó hozzáadásával a DTD-re hivatkozhatunk az XML-ben ezen adatelemek felhasználásakor. A kulcsszót ahhoz az XML utasításhoz kapcsoljuk, amely a DTD-re hivatkozik. [3]

Az XML előnyösen használható a következő tulajdonságai miatt:

- konfigurációs vagy egyéb átmeneti állományok leírása egyszerű,
- adatok cseréje biztosítható XML dokumentumokon keresztül (adatbázissal történő kommunikáció, platform független webszolgáltatások stb.),
- megjelenítés semleges, de külön megadható formázás a dokumentumokhoz (stíluslapok, CSS),
- a dokumentumok transzformálhatóak a megjelenítéshez (pl. XML -> HTML),
- biztosított az adatintegritás és az adatellenőrzés (parser, validálás),
- strukturált dokumentumok leírására jól használható,
- nyitott ajánlás.

4.3.2. Az XML felépítése

Az XML dokumentumot a *Document prolog* vezeti be. Meghatározza a használt XML verziót, a kódolást, illetve ha az XML dokumentum több részből áll, akkor a parsernek tudnia kell erről. Szükség van egy gyökér (legfelső szintű) XML elemre, ami az összes többit tartalmazza.

```
<?xml?>
<?xml version="1.0"?>
<?xml version='1.0' encoding='US-ASCII' standalone='yes'?>
<?xml version = '1.0' encoding= 'iso-8859-1' standalone ="no"?>
```

Az XML elemek a dokumentum építőelemei. Tartalmazhatnak további XML elemeket és/vagy szöveget. Ezeket tag-ek határolják.

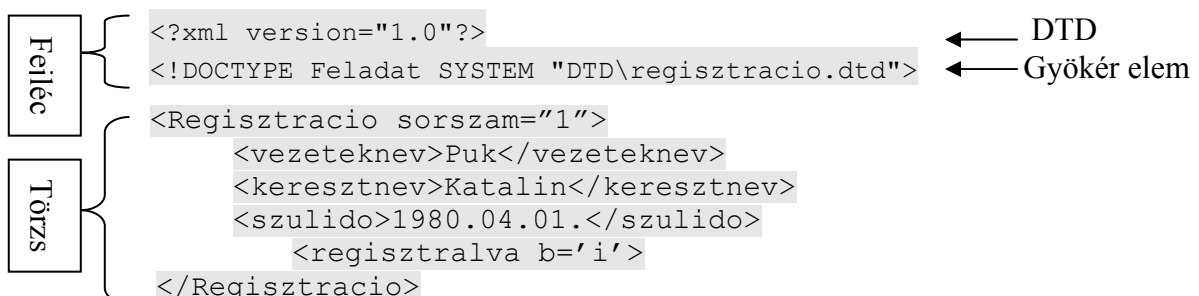
- Üres elem: `<xref linkend="abc"/>`
- Konténer elem: `<p>Ez egy paragrafus.</p>`
- Deklaráció: `<!ENTITY author "Naár János">`
- Feldolgozási előírás: `<?print-formatter force-linebreak?>`
- Komment: `<!-- ide irogatom a megjegyzéseimet -->`
- CDATA szekció: `<![CDATA[Amperands galore! &&&&]]>`

Minden nem üres XML elemhez szükséges egy nyitó és egy záró tag. Minden üres XML elemet jelölő tagnek tartalmaznia kell /-t a záró > előtt. A tagek beágyazási sorrendjének megfelelőnek kell lennie. Minden attribútum értéknek idézőjelek között kell szerepelnie. Minden saját tagnek az előírt formátumot teljesíteni kell. Egy XML dokumentum akkor jól használható, ha teljesíti a jól formáltsági követelményeket. A jól formáltság minden XML dokumentum fontos tulajdonsága, mert a szigorú szintaktika teszi lehetővé az egyszerű parserek írását. [1]

Az XML feladata az adatok strukturált leírása, s ebben nagy segítség, hogy nyelvezete nagyon minimális formai követelményeket követ, mely a következő példában is látszik:

```
<Regisztracio sorszam="1">
  <vezeteknev>Puk</vezeteknev>
  <keresztnev>Katalin</keresztnev>
  <szulido>1980.04.01.</szulido>
</Regisztracio>
```

Az előző példa kicsit kibővítve DTD-vel és egyéb elemekkel:



Az XML egyik nagy erőssége a dokumentum validáció. Többféle nyelv áll rendelkezésre ahhoz, hogy megadhassunk betartandó formai szabályokat. A legelterjedtebb a

DTD és az XML Schema, de létezik még sok egyéb nyelv is (pl. RelaxNG, Schematron). A szabályok teljesülését az XML feldolgozó ellenőrizni tudja. Hasznos megadni a használt nyelvtant, de ez nem kötelező, köszönhetően az XML szigorú szintaktikájának. [3]

4.3.3. DTD

A DTD fontos tényezője a saját nyelvtan, a saját elemkészlet, melyek alapján kell a dokumentumunkat elkészíteni. Ezen kívül a hibaellenőrzés is rendelkezésünkre áll a séma ellenőrzésére. Példa:

```
<!ELEMENT Pelda:Regisztracio (Pelda:vezeteknev, Pelda:keresztnev,
Pelda:szulev)>
    <!ELEMENT Pelda1:vezeteknev (#PCDATA)>
    <!ELEMENT Pelda1:keresztnev (#PCDATA)>
    <!ELEMENT Pelda1:szulido (#PCDATA)>
```

Módosító operátorok lehetnek:

- **#IMPLIED** – Nem feltétlenül szükséges.
- **#REQUIRED** – Szükséges.
- **#FIXED** – Kötelező a jelenléte és az értéke egyaránt.
- **"alapértelmezett érték"**

Az SGML-től örökölt deklarációk:

- *Dokument típus deklaráció:* A DTD fájl helye.

```
<!DOCTYPE RootElement SYSTEM "filename.dtd">
```
- *Elem típus deklaráció:* XML elem leírása.

```
<!ELEMENT Name (#PCDATA)>
```
- *Attribútum lista deklaráció:* Egy elemtípus attribútumait definiálja.

```
<!ATTLIST ElementName Name Datatype Default>
```
- *Egyed deklaráció:*

```
<!ENTITY Entity Name "text">
```

A DTD leírások önmagukban is lehelyezhetők egy XML dokumentumban, illetve külső állományban is. Külső állományban így néz ki az előző példa:

```
<?xml version='1.0' encoding='UTF-8'?>
  <!ELEMENT Pelda:Regisztracio (Pelda:vezeteknev, Pelda:keresztnev,
  Pelda:szulev)>
    <!ELEMENT Pelda1:vezeteknev (#PCDATA)>
    <!ELEMENT Pelda1:keresztnev (#PCDATA)>
    <!ELEMENT Pelda1:szulido (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Regisztracio SYSTEM "Regisztracio.dtd">
  <Regisztracio>
    <vezeteknev>
      Puk
    </vezeteknev>
    <keresztnev>
      Katalin
    </keresztnev>
    <szulido>
      1980.04.01.
    </szulido>
  </Regisztracio>
```

Beágyazva az XML-be:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Regisztracio [
<!ELEMENT Pelda1:vezeteknev (#PCDATA)>
<!ELEMENT Pelda1:keresztnev (#PCDATA)>
<!ELEMENT Pelda1:szulido (#PCDATA)>
]>
  <Regisztracio>
    <vezeteknev>
      Puk
    </vezeteknev>
    <keresztnev>
      Katalin
    </keresztnev>
    <szulido>
      1980.04.01.
    </szulido>
  </Regisztracio>
```

4.3.4. XML Schema definition Language (XSD)

Az XML elemkészletet és nyelvtant definiál, szintaxist használ (SAX, DOM). A legtöbb programnyelvben használt adattípusokat használhatjuk itt. A névterek használata lehetővé teszi több séma egybeolvasztását.

W3X szabvány. Olyan XML dokumentum, ami más XML dokumentum felépítéséről tárol információkat. Célja megegyezik a DTD-vel, de itt az XML Schema specifikációban megadott tagek felhasználásával XML dokumentumot hozunk létre. A DTD-vel szemben ez a leírás nem ágyazható be az XML dokumentumba. Sok beépített adattípust tartalmaz.

Alapelemei:

- Document prolog

- Séma elem:

```
<xsd:schema xmlns:xsd= "namespace">
```

```
<xsd:schema xmlns: xsd="http://www.w3.org/2001/XMLSchema" />
```

- Elem deklaráció:

```
<xsd:element name="name">
```

- Attribútum deklaráció:

```
<xsd: attribute name="name" type="type">
```

A document prolog szabvány XML. A gyökérelemként szolgál a séma elem leírás. Minden más elemet tartalmaz, illetve megadja a használt névteret. A névtér egy URL, ahol a használt XML szótár részletei találhatóak.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Az xsd: -al jelölt elemek ehhez a névtérhez tartoznak. Ha csak egy névteret használunk, akkor nem kell ezt a jelölést használni.

Kétféle elem deklarációról beszélhetünk: egyszerű és komplex deklarációról. Egyszerű esetben nem tartalmazhat az elem attribútumokat és további elemeket (csak szöveget), míg összetett esetben igen.

Egy egyszerű elem deklaráció:

```
<xsd:element name="date" type="xsd:date"/>
```

A komplex elemeknél a következő négy tartalmi elrendezés lehetséges.

- *Szöveg*: Csak szöveget tartalmaz az elem.
- *Üres*: Nem tartalmazhat sem szöveget, sem egyéb elemet (attribútumot igen).

```
<xsd:element name="source">
  <xsd:complexType>
    <xsd:attribute name="yearsInService"
      type="xsd:positiveInteger"/>
  </xsd:complexType>
</xsd:element>
```

- *Elem*: Gyermekek elemeket tartalmazhat.

```
<xsd:element name="bookInfo">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="title"/>
      <xsd:element ref="author"/>
      <xsd:element ref="publisher"/>
      <xsd:element ref="isbn"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

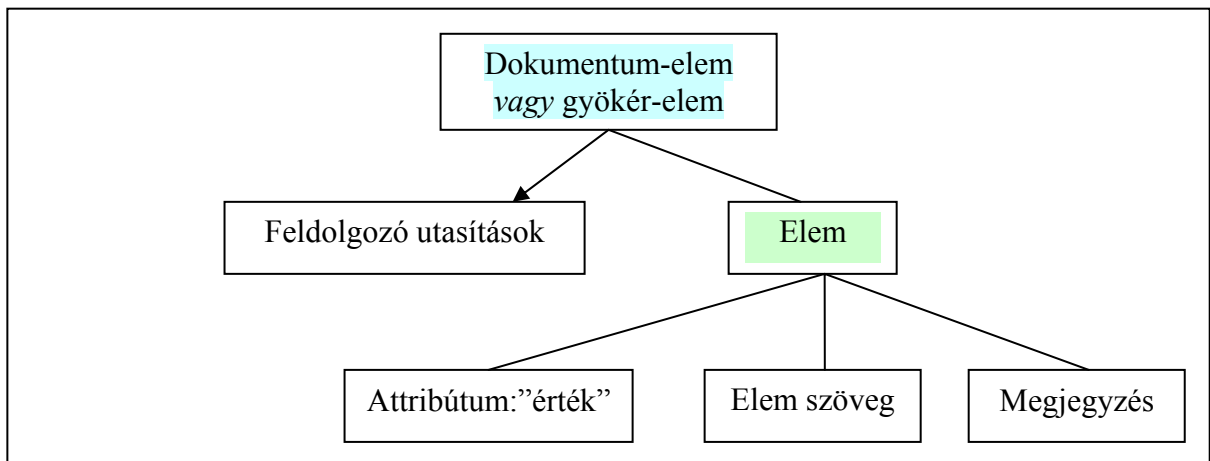
A `sequence` elem előírja, hogy az elemek ebben a sorrendben forduljanak elő; a `choice` esetén bármelyik bezárt elem előfordulhat; az `all` pedig nem ír elő semmit.

- *Kevert*: Lehetnek benne elemek, és az elemek között szövegek is. [3]

```
<xsd:element name="confirmOrder">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element ref="opening"/>
      <xsd:element ref="fullName"/>
      <xsd:element ref="date"/>
      <xsd:element ref="title"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

4.3.5. DOM – Document Object Modell

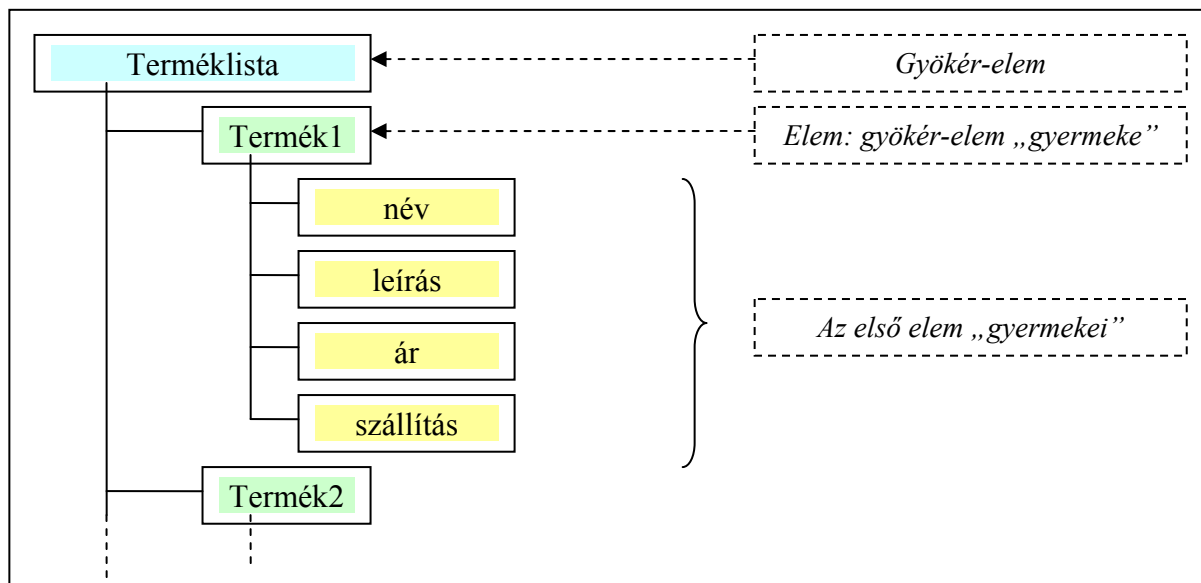
Az XML dokumentumokat egy XML feldolgozó alakítja át programokban felhasználható formátumúra. Ehhez ad egységes programozási interfészt a DOM. A DOM vizuálisan egy fával ábrázolható. Az XML dokumentum alkotóelemei a fában csomópontként jelennek meg.



4. ábra: Az XML dokumentum elemszerkezete

Az XML dokumentum tartalma elemeknek nevezett összetevőkre bontható. Az elem egy logikai egysége a dokumentumnak, és tartalmazhat további elemeket. Azt az elemet, amely mindegyik másikat tartalmazza, ezt gyökér-elemnek vagy dokumentum-elemnek nevezzük. A dokumentum-elemhez kapcsolódik a feldolgozási utasítás. Minden elemnek van azonosítója és lehet attribútuma (több is). Az attribútumok az elemre vonatkozó

tulajdonságokat tárolhatják. Ezt a hierarchikus elemszerkezetet (fa-struktúrát) nevezzük logikai szerkezetnek. A következő ábra egy konkrét elem-szerkezetet mutat be.



5. ábra: Egy konkrét példa a hierarchikus elemszerkezetre

Az XML alapú dokumentumnak van fizikai szerkezete is. Lehetőség van arra, hogy a dokumentum részeit (egyedeit) elnevezzük, és külön tároljuk olyan adatállományokban, amelyekből az információ újra kinyerhető és hivatkozásokkal, nem XML típusú (bináris) adatokkal is (pl. kép, hang stb.) bővíthető. [3]

4.3.6. XML és HTML kapcsolata

Az XML egy szabályrendszer (mint pl. az SGML), melynek segítségével leíró nyelveket definiálhatunk. A HTML ugyanakkor egy SGML-en alapuló leíró nyelv, melynek megadott elemkészlete van. Az XML az SGML szűk részhalmaza. Az XML létrejöttében jelentős szerepe volt a HTML nyelvvel szemben felmerült kifogásoknak. A HTML folyamatos bővítése miatt a kódba bekerülő formai elemek tartalmi szempontból szinte teljesen feldolgozhatatlanná tették a dokumentumokat. A XML kiterjeszthetősége lehetővé teszi a dokumentumok tartalmi szempontok alapján történő leírására. XML-ben deklarálhatunk több olyan tartalmilag különböző elemet, amelyeknek a megjelenítése akár ugyanaz is lehet. Az XML dokumentumokkal szemben támasztott követelmény, hogy emberi olvasásra is alkalmasak legyenek. Az XML döntően internetes alkalmazásra készült, ezért jól

használható dokumentumok webes publikálására. Egy dokumentumnak alapformátuma lehet az XML leírás, amely jól konvertálható más, pl. HTML, PDF stb. formátumokba.

Az XML és a HTML dokumentumok közötti formális különbségek a következőkben foglalhatók össze:

Jellemzők:	HTML	XML
Elemnevek (tag-ek)	rögzített (pl. br, img, style, table, ...)	szabadon definiálhatók (kiterjeszthető nyelv)
Kis- és nagybetűk	nem tesz különbséget köztük	különböznek egymástól
Elem és attribútum nevek	kis- és nagybetűvel is írhatók	csak kisbetűvel írhatók
Elemek lezárása	nem kötelező	kötelező (a párja, ha nincs /)
Átlapolás	lehet	tiltott
Deklaráció	nem kötelező	kötelező (karakterkódolás, verzió)
Attribútumok értékei idézőjelek között	nem kötelező	kötelező
Attribútumok minimalizációja	lehetséges	nem támogatott

Az XML és a HTML közti lényegi különbség elsősorban abban jelentkezik, hogy a HTML az adatok megjelenését, formátumát is definiálja, míg a XML csak a tartalom struktúráját, biztosítva az alkalmazások közötti egyszerű adatcsere lehetőségét. XML-ben léteznie kell egy gyökér (root) címkének (elemnek), és adatok csak ezen belül helyezhetők el. Az egyes címkék hierarchikus viszonyban állnak egymással.

4.3.7. Az XML felhasználási területei és lehetőségei

Az XML alkalmas adatformátum a különböző médiákon terjesztett, közvetlenül megjelenített strukturált szövegek számára. Az XML formátumú információ „önleíró” – a megcélzott felhasználó és a használt média követelményeihez igazodóan használható, jól alakítható és formázható anélkül, hogy újra a kiszolgálóhoz kellene kapcsolódni.

XML alapokon leíró nyelvek definiálhatók. Ezek a nyelvek bővíthetők, tetszőleges saját elemek deklarálhatók bennük. Az elemek szabadon egymásba ágyazhatóak. A legmagasabb logikai szinten minden elemet egyetlen dokumentumelembe (gyökér-elembe) kell zárni. A dokumentumelem egy konkrét előfordulását nevezzük XML-dokumentumnak.

Az XML specifikáció meghatároz egy szintaxist az elemek jelölésére, és még egyet a szabályok leírására (DTD). Az XML elemek alkotják a definiált nyelv szókincsét. Az elemek tartalmát és az elemek kapcsolódását rögzítő szabályok adják a nyelv nyelvtanát. A szintaxis betartásával saját nyelv készíthető és az azzal készített dokumentum a megfelelő eszközzel ellenőrizhető, feldolgozható.

Az XML megjelenése óta komoly sikereket ért el az elektronikus kereskedelemben, a tudományban, a számítástudományban és az ipari alkalmazásokban. Sok ipari DTD létezik. A legjelentősebb szoftvergyártók számos XML-t támogató termékek fejlesztettek. Bonyolult adatformátumok két program közötti mozgatásához nagyon jól alkalmazható az XML formátum. Az XML relációs adatbázisok rendszerek közötti mozgatására is alkalmazható, melynek során az XML elemek határolóként funkcionálnak (a rekordok, mezők és kapcsolatok átvitelekor). [8]

Az XML szabványok a könyvtári informatikában is megjelentek. Megkezdődött a MARC formátum XML-ben történő leírására (pl. XMLMARC), amelynek eredményeként lehetővé válhat a könyvtári adatok kezelése, visszakeresése XML-re épülő eszközökkel. Az XML-t használják már a tezauszépítésben és -szerkesztésben is (Virtual HyperGlossary).

Az alábbi felsorolás a fontosabb XML-re épülő alkalmazásokat veszi számba, és rövid ismertetés ad róluk:

- **XHTML**: A HTML XML alapokon újradefiniált változata.
- **Docbook**: Dokumentációk, könyvek strukturált leírására szolgáló nyelv.
- **TEI**: Szövegek elektronikus feldolgozása céljából kifejlesztett nyelv. Szépirodalmi és tudományos szövegek markup-olására használható.
- **XMLNews**: a hírek tartalmának és metaadatainak leírását teszi lehetővé. Hírügynökségek használják hírszolgáltatásra (pl. Reuters, AFP).
- **XFRML**: Pénzügyi jelentések leírására szolgáló nyelv.
- **WML**: A WAP specifikációt támogató eszközökhöz tervezett leíró nyelv (a WAP-os mobiltelefonok nyelve).
- **WEB-DAV**: A HTTP-re épülő XML szintaxisú protokoll. A Windows-ban lévő "Web-mappa" is ezt használja.
- **SVG**: Két dimenziós, statikus és mozgó vektorgrafikák meghatározására szolgáló XML alapú szöveges leíró nyelv. A W3C által definiált nyílt szabvány.

4.4. XHTML – a HTML megújulása XML alapokon

Az XHTML (eXtensible HyperText Markup Language - bővíthető hyperszöveg lapleíró nyelv) a HTML egy fajtája (ötödik szintje), mely kompatibilis az XML nyelvvel. Gyakorlatilag a HTML 4.01-nek az XML-alapokon nyugvó kiegészítése, kiterjesztése az XHTML 1.0 verziója. A korábbi SGML nyelv helyett az XML szabvány kerül előtérbe általa, amely egyszerűsíti a webes adatbázisok közötti kommunikációt is, és nagy előnye, hogy bővíthető. Az XHTML 1.0-s verziója 2000 januárjában, az 1.1-s verziója 2002 augusztusában került publikálásra. A W3C HTML Munkacsoport 2005 májusában adta ki az XHTML 2.0 legújabb munkatervét, amely a leírónyelvet és moduljait írja le. Ennek a verziónak a célja a látványos, hordozható web-alapú alkalmazások létrehozása. Az XHTML 2.0 inkompatibilis a HTML 4.01, a XHTML 1.0 és 1.1 verziójával.

Az XHTML nyelv funkcionalitása a HTML és az XML képességeit ötvözi. Egyaránt alkalmas a szöveges és képi tartalom gyors közzétételére és nagyobb mennyiségű adatok továbbítására. A mobil eszközök közül ma már sok rendelkezik a XHTML nyelven megírt lapok megjelenítésének képességével. Az XHTML 1.0 elemkészlete megegyezik a HTML 4.01-es verzióéval, viszont az elemek használatának módja jelentős mértékben megszigorodott. A szigorítás oka abban rejlik, hogy míg a HTML egy SGML alkalmazás, addig az XHTML egy XML-en alapuló leíró nyelv. Mivel az XML az SGML egyszerűsítéséből és szigorításából jött létre, ebből következően az XHTML-nek is szigorúbbnak kell lennie a HTML-nél. [9]

4.4.1. Az XHTML dokumentum alapszerkezete

Az XHTML dokumentum az XML deklarációval kezdődik, amely azonban elhagyható, ha az oldal UTF-8 kódolásban készült. Az XML deklaráció megadja az XML verziót és a karakterkódolást. Ezt követően egy stíluslapot csatolhatunk az oldalhoz. Ezután következik a DOCTYPE deklaráció. A DTD-nek az alábbi három változata van:

Szigorú - strict:

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

Az elnevezése a szigorú, pontos megfeleltetésre utal. A dokumentum mindazon tulajdonságának meglétét megköveteli, amit a specifikáció kötelezőként jelöl meg (pl. DOCTYPE, xmlns elem, gyöker-elem stb.). Minden megjelenéssel kapcsolatos utasítást lépcsős stíluslapokkal kell megadni.

Átmeneti - transitional:

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

Az átmeneti megfeleltetés olyan oldalak esetében használatos, ahol a megjelenést csak részben írja le CSS. Ez biztosítja, hogy az oldalhoz olyan böngészők is hozzáférnek, amelyek csak részben támogatják a stíluslapok.

Keretkészletes - frameset:

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

A keretkészletes változat használatával lehetővé válik az oldal vonalakkal történő felbontása névvel azonosított keretekre (pl. fejléc, menü, szöveg stb.)

A helyes DOCTYPE meghatározás azért fontos, meg a böngésző innen tudja meg, hogy milyen módon jelenítse meg a weblapot, azaz milyen üzemmódba kapcsoljon. A html, a head és a body elemeknek szerepelniük kell a dokumentumnak, a title elemnek pedig a head elemen belül kell elhelyezkednie.

A html címkének kötelező megadni az xmlns=<http://www.w3.org/1999/xhtml> attribútumot, amely az XML névteret állítja be. Itt kell megadni a weboldal fő nyelvét az xml:lang="nyelv kód" attribútummal. A főnyelvtől eltérő szöveg esetén a következő módon válthatunk át más nyelvre:

```
<p>Magyar szöveg <span xml:lang="de">valami németül </span> magyarul
ismét</p>
```

Az alábbi példa egy egyszerű XHTML 1.0 dokumentum alapszerkezetét mutatja be:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/css" href="style.css"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="hu">
<head>
<title>Kölcsey Ferenc: Emléklapra</title>
</head>
<body>
<p>Négy szócskát üzenek, vésd jól kebeledbe, s fiadnak
```

```
Hagyd örökül ha kihúnysz: A HAZA MINDEN ELŐTT.</p>
</body>
</html>
```

Az XHTML 1.1 verzió a Strict továbbfejlesztése, ezért a megjelenést már csak stíluslappal lehet beállítani. Ebben a verzióban már csak egyfajta DOCTYPE-ot használhatunk.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

4.4.2. Az XHTML elemei

Dokumentum felépítés: body, head, html, title

Sztruktúra: address, blockcode, block quote, div, hn, p, pre, section, separator

Szövegkezelés: abbr, cite, code, dfn, em, kbd, l, q, samp, span, strong, sub, sup, var

Lista: dl, dt, dd, label, nl, ol, ul, li

Hypertext: a

Táblázat: caption, col, cplgroup, summary, table, tbody, td, tfoot, th, tr

Metaadatok: meta, link, object, param, standby

Jellemzők: xml:lang, edit, src, style, datatype, about, src, dir, rel, property, class, id,

4.4.3. Az XHTML és a HTML összehasonlítása

Az XHTML és a HTML közötti főbb különbségek – melyek XML kompatibilitási követelményből fakadnak – a következők:

- minden elem és attribútum kisbetűvel írandó (az XML szabvány megkülönbözteti a kis- és nagybetűket),
- minden elemet kötelező lezárni, pl. `<p> Ez az első bekezdés</p>`, ugyanakkor az olyan elemek esetében, melyeknek nincs a HTML-ben záró párjuk (area, base, bgsound, br, col, frame, hr, img, input, isindex, keygen, link, meta, option, param), / karaktert kell tennünk, pl. `<br />`,

- az elemek egymásba ágyazása során a megnyitásnak fordított sorrendjében kell a záró elemeket elhelyezni pl. `<b><u>Félkövér, aláhúzott szöveg</u></b>`
- minden elemet a `<html>` elemen belül kell elhelyezni, vagyis jól formázott dokumentumot kell létrehozni,
- a paramétereket és az attribútumokat idézőjelbe kell tenni, pl. `<table width="150">` vagy `<input checked="checked" />`,
- az attribútumok minimalizálása (rövidítése) tiltott, (A 3. sz. melléklet a HTML-ben minimalizálható attribútumokat és az XHTML megfelelőjét tartalmazza.)
- kötelező entitással helyettesíteni a szövegben előforduló `<` és `&` karaktereket (még a JavaScript-ben és a CSS-ben is, ha az nem külső) és ajánlott kódolni a `>` és `'` karaktereket is,
- csak az előző öt névvel ellátott entitás (`&`, `>`, `<`, `"`, `'`) alkalmazható biztonsággal, ezeken túl számmal kell hivatkozni (pl. ` `;
- az elemazonosításra és navigálásra alkalmazott „name” helyett „id” attribútumot kell használni (régebbi böngészők miatt érdemes mindkét attribútumot használni, ugyanazon értékkel),
- az adott elem nyelvének megadására a „lang” mellett az „xml:lang” elemet is szerepeltetni kell, pl. `<div lang="hu" xml:lang="hu">Magyar vagyok</div>`.

A XHTML-nek az előnyei a HTML-lel szemben a következők:

- A kevesebb szabály miatt könnyebb olyan algoritmust írni, ami XML fájlt és nem SGML fájlt kezel.
- Az XHTML fájlt könnyebb feldolgozni, mint a HTML fájlt, mert az XML egyszerűbb nyelv.
- Az XHTML fájlt csak akkor jeleníti meg a böngésző, ha az szabványos. A hibás oldal helyett csak egy hibaüzenet jelenik meg. A HTML oldal hibásan is megjelenik valahogy. Az XHTML-ben dolgozó webszerkesztőt a szabályosságra készíti.
- Az XHTML weboldalba más XML-re épülő nyelvek is használhatók, pl. SVG (Scalable Vector Graphics – méretezhető vektorgrafikák: leíró nyelv kétdimenziós, statikus és mozgó vektorgrafikák meghatározására), MathML (Mathematical Markup Language – matematikai leíró nyelv). [9]

4.4.4. Az XHTML oldal hibátlanságának alapvető kritériumai

Ahhoz, hogy az XHTML oldal hibátlan legyen, két kritériumnak kell megfelelnie: a jól formázottság (well-formedness) és az érvényesség (validity) kritériumnak. A jól formázottság lényegét tekintve azt jelenti, hogy valamennyi elemnek vagy lennie kell záró tagjének, vagy pedig speciális módon kell leírni, továbbá az összes elemnek beágyazottnak kell lennie. Amennyiben XHTML oldalunkat `application/xhtml+xml` típusban (és nem `text/html` típusban) küldjük, akkor a jól formázottság lényeges követelmény, különben a böngésző az oldal helyett csak egy hibaüzenetet mutat. Az érvényesség követelményének akkor felel meg a kód, ha jól formázott és a kiválasztott DTD szabványnak megfelel. Az érvényesség követelményének való megfelelés azt biztosítja, hogy minden böngészőben ugyanaz jelenjék meg. Az érvényesség követelménynek nem kötelező megfelelnie az oldalnak, de mindenképpen ajánlott. A jól formázottság és az érvényesség követelményének való megfelelését egy oldalnak a „W3C validator”-ral¹⁰ könnyen ellenőrizhetjük.

4.5. DHTML

A DHTML (Dynamic HyperText Markup Language - dinamikus hiperszöveg jelölési nyelv) a HTML nyelv dinamikus használatra módot adó kiterjesztése, mely egy weboldalt aktívabbá tesz JavaScript segítségével manipulált HTML elemeken keresztül. weboldalak tartalmának dinamikus megjelenítését szolgálja ábrák, szövegek és egyéb oldalelemek mozgatásával. Ellentétben az egyszerű információközlő statikus weboldallal, a dinamikus oldalra az interaktivitás és a változás jellemző. Segítségével a dokumentumok tartalma, stílusa, az egyes elemek elhelyezkedése interaktívan, percről perce változhat. [7]

A webágensek (webes események előidézői) hatására bizonyos feladatok akkor kerülnek végrehajtásra, mikor egy-egy meghatározott típusba tartozó esemény (pl. a felhasználói felület vagy tartalom változása, tartalom betöltése, illetve a működési környezettől származó utasítás) bekövetkezik. Egyes jelölőnyelvek lehetővé teszik adott típusú események bekövetkeztekor scriptek - úgynevezett **eseménykezelők** - végrehajtását. Az eseménykezelő egy az egyben adott eseményhez van rendelve, amennyiben az elemet az

¹⁰ A <http://validator.w3.org> oldalon érhető el.

elemjelölő vagy a DOM az adott elemhez rendeli. A „beágyazó eseménysor” olyan programozási eljárás, amikor adott eseménykezelő bizonyos eseményt egynél több elemhez rendel. Ebben az esetben az eseménykezelők nincsenek egy az egyben hozzárendelve az események által érintett elemekhez (eltekintve attól, ha az esemény bekövetkezte egyetlen elemtől függ).

A HTML, stíluslapok, DOM (Document Object Model - Dokumentum-objektum Modell) és scriptek együttes használata esetén általában "dinamikus HTML"-ről vagy DHTML-ről beszélünk, viszont egyetlen W3C-szabvány sem határozza meg a DHTML fogalmát (sőt, a W3C nem ajánlja a DHTML használatát).

A DHTML előnyét a kliensoldalon reagáló oldalak jelentik. Hátránya viszont az, hogy sokan a teljes menürendszert ebben készítik, viszont csak Explorer alatt működik.

4.5.1 JavaScript

A JavaScript a Sun és a Netscape cég által kifejlesztett szkriptnyelv (parancsnyelv) egyszerűbb webes programozási feladatok céljára (pl. interaktív funkciók, dinamikus HTML elemek stb.). A Java programozási nyelvre csak hasonlító JavaScript parancsok beírhatók a HTML állományokba és a Web-kliens programok futásidőben soronkénti értelmezés és fordítás után végrehajtják azokat. Azért parancsnyelv, mert a forrásprogramot az interpreter (parancsértelmező) hajtja végre. A Java és a JavaScript nem ugyanaz, mert míg a Java a szerveren kerül értelmezésre, fordításra végrehajtás előtt, addig a JavaScriptet a böngésző értelmezi. Szintaktikailag és felépítésében nagyon hasonlít a Java programozási nyelvre, lehetőségei azonban sokkal korlátozottabbak. A JavaScript támogatja a függvényeket, amelyeket speciális deklarációs követelmények nélkül lehet használni. A függvények lehetnek objektumok elemváltozói, amelyeket "laza" típusú módszerrel hajtanak végre. A JavaScript programok leggyakoribb felhasználása az űrlapokhoz kötődik. Egyes régebbi böngészőprogramok eltérő módon támogatják, vagy nem ismerik a JavaScript nyelvet, ezért ez DHTML oldalak futtatásakor problémákat okozhat.

A JavaScript úgy működik, hogy megkeresi a célzott HTML elemet, és ezzel csinál valamit. A JavaScript alkalmazásával a HTML oldalakban különböző funkcionalitást adhatunk a weblapunkhoz, melyek közül a leggyakrabban alkalmazottak a következők:

- a felhasználó által beírt adat ellenőrzése (helyesen adta-e meg vagy sem),

- a weblaphoz drag and drop („fogd és vidd”, adatmozgatás) funkcionalitást adhatunk,
- lecserélhetjük vele gombnyomásra a lap stílusát,
- animálhatók a lap elemei (például a menük, képek),
- lekezelhetők a gombok eseményei stb.

A JavaScript forráskódokat a HTML-fájlba kell elhelyezni, amit az alábbiak szerint tehetünk meg:

- utasításokat és függvényeket a SCRIPT tag-ben lehet használni;
- eseménykezelőket a HTML tagben lehet használni (HTML tag a HTML formázó utasítás, a JavaScript utasításaitól való elkülönítés miatt);
- külön fájlba hivatkozással;
- JavaScript-es entitásba;
- internetes címekben a „javascript:” előtaggal.

Beágyazó szintaxis:

```
<SCRIPT LANGUAGE="JavaScript">
<!--
program
/-->
</SCRIPT>
```

Példa egy egyszerű szkriptre:

```
<html>
<head>
<script language="JavaScript">
document.write("Ez egy szöveg a head részben.")
</script>
</head>
<body>
Ez egy szöveg a body részben.
</body>
</html>
```

Ekkor ez jelenik meg:

Ez egy szöveg a head részben. Ez egy szöveg a body részben.

Példa egy függvénnyel beágyazott szkriptre:

```
<html>
<head>
< script language ="JavaScript">
<!--
    function square(i) {
        document.write("Paraméterként ", i ,"-t adtunk a függvénynek. ")
        return i * i
    }
    document.write("A függvény ",square(5),"-el tért vissza.")
```

```

-->
</script>
</head>
<body>
<br>
Kész!
</body>
</html>

```

Ekkor ez jelenik meg:

Paraméterként 5-t adtunk a függvénynek. A függvény 25-el tért vissza.
Kész!

A JavaScript programban előforduló adatok csak a következő négy különböző típusba tartozhatnak: objektum, számérték (egész vagy valós számok), szöveg (karakter sorozat) és logikai érték (*true* és *false*).

Változókat a következő módon hozhatunk létre:

```
var változónév [= érték]
```

Függvények definiálására a

```
function függvénynév ( paraméterek ) { ... return érték }
```

forma szolgál. A függvény visszatérési értékének típusát nem kell definiálni, ez dinamikus, és attól függ, hogy a *return* milyen értéket ad vissza.

A JavaScript objektum <tulajdonság, érték> párok halmaza. Az egyes tulajdonságokhoz (*property*) az

```
object.property
```

szintaxissal hivatkozhatunk, az értékét használhatjuk, módosíthatjuk.

Objektum létrehozása az alábbi példában:

```

function tanulo (vnev, knev, kor)
{
  this.vezeteknev = vnev;
  this.keresztnev = knev;
  this.kora       = kor;
}
Kata = new person("Minta", "Kata", 20);

```

A JavaScript objektumokat asszociatív tömböknek is tekinthetjük, ahol a tulajdonság nevekhez értékek tartoznak, amelyek a nevek alapján hozzáférhetők.

A HTML lapok különböző elemeihez magához a laphoz, az egyes hivatkozásokhoz, űrlap elemekhez a böngésző az egér mozgatásával, a billentyűzet használatával kapcsolatban különböző eseményeket rendel. Az egyes események feldolgozására az új eseménykezelők (*event handler*) szolgálnak, amelyeket a megfelelő HTML elem definíciójánál lehet megadni.

Az **eseménykezelő** egy tetszőleges JavaScript utasítássorozatot tartalmazhat, leggyakrabban egy függvény meghívását tartalmazza.

<i>onLoad</i>	betöltünk egy oldalt a böngészőbe
<i>onUnload</i>	kilépünk egy betöltött oldalról
<i>onFocus</i>	az űrlap valamelyik eleme a fókuszba kerül, azaz a billentyűzetről érkező karaktereket fogadja
<i>onBlur</i>	a fókusz elmozdul az űrlap valamelyik eleméről
<i>onSelect</i>	az űrlap valamelyik bemeneti mezőjét kiválasztották
<i>onChange</i>	az űrlap valamelyik elemének (szöveg, választás) értékét megváltoztatták
<i>onClick</i>	rákattintottak egy űrlap elemre (pl. nyomógombra), vagy hivatkozásra
<i>onSubmit</i>	elküldenek egy űrlapot a szerver felé (a <i>submit</i> gombra kattintottak)
<i>onMouseOver</i>	az egér egy hivatkozás, vagy dokumentum címke fölé kerül

Példa az eseménykezelésre egy kalkulátor képében:

```
<html>
<head>
  <script language="JavaScript">
    <!--
    function compute(form)
    {
      var operator = form.op.options[form.op.selectedIndex].text
      form.result.value = eval(form.x.value + operator + form.y.value)
    }
    //-->
  </script>
</head>
<body>
  <form name="kalkulator">
    <input type="text" NAME="x" SIZE=8>
    <select name=op>
      <option>+ <option>- <option>* <option>/
    </select>
    <input type="text" name="y" size=8>
    <input type="button" value=" = "
      onClick="compute(document.calculator)">
    <input type="text" name="result" size=8>
  </form>
</body>
</html>
```

És az eredmény:

	+		=	
----------------------	--	----------------------	--	----------------------

Példa új ablak nyitására:

```
<html>
<head>
< script language ="JavaScript">
```

```

<!--
function WinOpen() {
    msg=open("", "DisplayWindow", "toolbar=no,directories=no," +
"menubar=no,resizable=no,width=300,height=200");
    msg.document.write("<head><title>Hahó!</title></head>");
    msg.document.write("<body>");
    msg.document.write("<center><H2 align=center>Ez itt egy új
ablak!</H2></P>");
    msg.document.write("<br><br>");
    msg.document.write("<form>");
    msg.document.write("<input type='button' value='Becsuk!' " +
                        onclick='window.close()' '>");
    msg.document.write("</form>");
    msg.document.write("</body>");
}
//-->
</script>
</head>
<body>
<form>
<input type="button" value="Katt IDE!" onclick="WinOpen()">
</form>
</body>
</html>

```

Az összes C-ben használható **operátor** itt is használható, az eltérések a következők:

- >> egyszerűen kettővel osztja a számot, az előjelet megtartva.
- >>> jobbra tolja a számot, és balról nullákkal tölti fel (tehát az előjel megváltozhat)
- a logikai kifejezések nem feltétlenül értékelődnek ki teljesen. Tehát ha az első operandusból lehet következtetni az eredményre, akkor a második operandus kiértékeletlen marad.
- kétféle idézőjel használható: a " és a '. Ezeket felváltva használva ágyazhatunk egymásba idézőjeleket. Egyébként egy \-t írva elé hasonló hatást érhetünk el.
- az idézőjelek közé zárt szöveg automatikusan egy string objektum, tehát tetszőleges műveletek végezhetők vele.

Különleges kifejezések:

- **true** és **false** konstansok igaz és hamis értékekkel.
- **this** ez mindig az aktuális objektumra hivatkozó változó.
- **null** a semmilyen objektumra hivatkozik.
- **void(kifejezés)** kiértékeli a kifejezést, majd elnyeli az értékét, aminek következtében a teljes kifejezésnek nem lesz értéke.
- **typeof(objektum)** egy stringet ad vissza, ami a kifejezés típusát jellemzi.

- **new típus(paraméterek)** értéke egy új objektum a megadott típusból, a megadott paraméterekkel (ezeknek jelentése típusonként változó).
- **return [kifejezés]** függvényből visszatér a megadott kifejezés értékével, vagy érték nélkül.
- **function név(paraméterek) { utasítások; }** definiál egy adott nevű függvényt, a paraméterek a függvény belsejében mint változók használhatók.
- **var név [= kifejezés]** definiál egy változót. Ha még nincs ilyen nevű, akkor a var elhagyásával egy értékadás is definiálhatja.
- **eval(string)** kiértékeli a megadott Javascript kifejezést
- **parseInt(string)** egész számmá konvertálja a stringet
- **parseFloat(string)** lebegőpontos számmá konvertálja
- **isNaN(szám)** megadja, hogy a szám érvénytelen érték-e

4.5.2 Stíluslapok - CSS

A HTML nyelv formázó utasításaival leírható a dokumentum tartalma, például címsor, bekezdés, rövidítés, kiemelt szöveg stb. A böngészőprogramok fejlesztése során több ponton túlmentek a HTML szabványon és új tageket, paramétereket vezettek be, amelyekkel a dokumentum megjelenésére helyezték a hangsúlyt. Ezek idővel a szabványba (újabb verzióba) is bekerültek, így ma már a HTML-ben nehéz szétválasztani a tartalmat a megjelenéstől. Egy hagyományos webes dokumentum kódjának nagyobb részét alkotják a formázási utasítások és kisebb rész a tulajdonképpeni tartalom. [7]

Szükségessé vált a formázások egyszerűsítése és egységesítése a dokumentumon belül. Ennek a megoldására a W3C kidolgozta és felügyeli a CSS (Cascading Style Sheets - lépcsőzetesen egymásra épülő stíluslapok) specifikációját, amely biztosítja a tartalom és a megjelenés elkülönítését, egyben szebbé tehető a megjelenés. A CSS tehát egy stílusleíró nyelv, amely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le. Ezen túl használható bármilyen XML alapú dokumentumok stílusának leírására is. A CSS-sel a weboldalnak egységes megjelenést adhatunk, és annak módosítása is egyszerűbbé vált.

Kidolgozásakor az elsődleges szempont az volt, hogy elkülönítsék a dokumentumok struktúráját (ezt HTML vagy hasonló leíró nyelvben lehet leírni) a megjelenésétől (ezt lehet CSS-sel megadni). Ez a megoldás növeli a weblapok használhatóságát, rugalmasságát és a

megjelenés kezelhetőségét, másrészt csökkenti a dokumentum tartalmi struktúrájának komplexitását. A CSS előtt a HTML dokumentumok csaknem minden megjelenéshez kapcsolódó része a HTML kódon belül volt, amelyek gyakran ismétlődtek. A CSS használatával a megjelenítéssel kapcsolatos információk átkerülhetnek a stíluslapra, mely így egy sokkal egyszerűbb, kevésbé redundáns HTML kódot eredményez. A CSS arra is alkalmas, hogy a dokumentum stílusát a megjelenítő eszköz vagy módszer alapján adja meg, így más-más a dokumentum formája a képernyőn, nyomtatott lapon, hangos böngészőben (beszédszintetizátoros felolvasó), vagy braille készüléken megjelenítve. [4]

A CSS 1 specifikációja 1996-ban készült el, ugyanakkor három évnek kellett eltelnie ahhoz, hogy egy böngészőben (Internet Explorer) megjelenjen a teljes megvalósítása. A fejlettebb utasításkészlettel rendelkező CSS 2 1998-ban jelent meg, de még 2004-ben sem volt olyan böngésző, amely teljes mértékben támogatta volna. Jelenleg a CSS 2.1 szintet használjuk, amely a lényegében a 2 verziónak a nem működő részekről megtisztított változata.

A CSS szintaxisa egyszerű, hiszen csak néhány kulcsszót használ a stílusok leírásához. A stílust leíró szabályok sora maga a stíluslap. Minden szabály (utasítás) két részből épül fel a következők szerint: `szelektor {tulajdonság: érték; tulajdonság: érték; ...}`

1. *szelektor (kiválasztó)*: alapvető esetben egy HTML elem, amelyre a stílusutasítás vonatkozik (pl. `p`, `h2`, `td`, `table`, stb.)
2. *deklarációs szakasz*: kapcsos zárójelek között pontosvesszővel elválasztott deklarációkat tartalmaz. A tulajdonság a formázási utasítás neve (amit módosítani akarunk), a tulajdonság értéke (amire módosítunk, a több szavas értéket idézőjelbe kell tenni).

Pl.: `p {color: blue}, p {font-family: "sans serif"; color: #808080},
h2 {align: center; font color: blue; font-family: Verdana}`

A CSS megszületése előtt a webszerkesztők, ha módosítani akarták a `h2` elemek jellemzőit, akkor a HTML font elemét kellett használniuk a címsor minden használatakor. Egy középre igazított, kék, Verdana betűtípusú címsorhoz a következőt kellett használni:

```
<h2 align="center"><font color="blue" face="Verdana, serif">  
<em>CSS nélküli megoldás</em></font></h2>
```

A prezentációs jelölések miatt a HTML dokumentum bonyolult lesz, és nehezen karbantartható. Amennyiben minden címsort ilyen megjelenésben szeretnénk látni, akkor mindegyikre külön-külön használni kell ezt a kiegészítést. A dokumentumot olvasó felhasználónak nincs módja változtatni a megjelenésen, ha például zöld színű címsorokat szeretne, mivel az oldal készítője egyértelműen megadta a címsorok színét.

A CSS használatával a h2 elem a HTML dokumentumban csak strukturális információt hordozhat, hozzá a stíluslap adja a megjelenítési tulajdonságokat. Az alábbi stílus definiálja az összes h2 címsor fenti példa szerinti megjelenését a teljes dokumentumban, esetleg webhelyen:

```
h2 {
  text-align: center;
  color: blue;
  font-family: "Verdana", serif
}
```

h2 címsor stílusa:
= *szöveg igazítása középre*
= *színe kék*
= *Verdana betűtípus, talpas helyettesítő betű*

A megjelenés így elvált a tartalomtól, tehát a HTML csak a strukturális, míg a CSS a prezentációs információkat írja le. Az előző példában az utasítást a jobb áttekinthetőség érdekében úgy tagoltuk, hogy egy-egy sorban csak egy deklarációt írtunk.

Több elemre vonatkozóan egyszerre is kiadhatunk utasítást, ekkor az elemeket vesszővel választjuk el egymástól. Egy ilyen példa következik:

```
p, td, h3, ul {
  font-family: arial;
  color: yellow;
  background-color: black
}
```

bekezdés, cella, 3 szintű címsor, egyszerű lista formátuma:
= *Arial betűtípus*
= *sárga színű betű*
= *fekete háttér*

Amennyiben a dokumentumban többször előforduló elemre (pl. p, td, table) többféle utasítást akarunk kiadni jól használhatjuk a stílus osztályokat (class). Elsőként a stíluslapon definiáljuk az azonosított elem stílusát, pl. egy középre igazított bekezdését

```
p.kozepre {text-align: center; font-weight: bold}
```

amelyre a HTML dokumentumban így hivatkozunk:

```
<p class="kozepre">Ezt a dőlt betűs szöveget középre igazítja.</p>
```

Amennyiben a stílus osztályt önállóan használjuk, úgy az a teljes dokumentumra vonatkozó stílust határozza meg, például ha mindent Verdana betűtípussal kívánunk megjeleníteni az oldalon, akkor a `.betu {font-family: verdana}` szabályt használjuk.

Egyetlen HTML elemre vonatkoztatott utasítás esetén az egyedi kiválasztót használhatjuk, melynek a # (kettőskereszt, ID) a jele. A következő példa a p elemre vonatkozó egyedi kiválasztót (bevezetes) mutatja be:

```
p#bevezetes
{
  font-family: Garamond;
  text-align: left;
  font-weight: bold;
  font-size: 110%;
  color: blue
}
```

„bevezetes” nevű bekezdés stilisztikai tulajdonságai:
= *Garamond betűtípus*
= *szöveg igazítás balra*
= *félkövér betű*
= *betűméret 110%-os*
= *színe kék*

Látszólagos osztályokat is definiálhatunk, amelyekkel az elemek stílusát csak annak valamilyen állapotában határozzuk meg. A linkek megjelenése a legjobb példa erre. Az elem után kettőspontot írunk, majd az elem egy lehetséges állapotát írjuk és ezt követi a deklaráció.

Ilyen példa a következő:

a: active	{color: red }	aktív link – vörös színű
a: hover	{color: yellow }	kijelölt link - sárga színű
a: visited	{color: orange}	látogatott link – narancs színű
a: link	{color: blue }	nem látogatott link - kék színű

A CSS stíluslapok megjegyzéseket is tartalmazhatnak, melyeket a böngésző nem hajt végre, viszont egy későbbi kódmódosítás során jó szolgálatot tesz. Segítségével bizonyos elemeket ideiglenesen kizárhatunk a végrehajtásból. Példák a megjegyzések használatára:

```
/* bekezdés formázása */
P {
  color: white;
  /* text-align: center */
}
```

← ezt a deklarációt átmenetileg kizárjuk

A stílustulajdonságok átöröklésére is van lehetőség olyan módon, hogy egy szelektor átveszi a befoglaló elem azonos tulajdonságát. A következő példában a prezentáció módját a body elemre definiáljuk, amelynek következtében ez minden body elemen belüli további elemre is vonatkozik (átöröklődik).

```
<html>
  <head>
    <style type="text/css">
      body {color: blue; font-weight: bold }
    </style>
  </head>
  <body>
    <p> Minden szöveg kék színű és dőlt lesz </p>
  </body>
</html>
```

← stílusleíró (stíluslap)

A CSS nyelven készített stíluslapoknak a dokumentumhoz csatolása az alábbi négyféle módon lehetséges:

1. A `style` elem elhelyezésével, amely a teljes dokumentumra érvényes. Ekkor a stílusutasításokat beépítjük a HTML oldalak kódjába a `<head>` és `</head>` közé. A style elem paraméterezése: `<style type="text/css">`. Az előző példában ezt a megoldást láthatjuk.
2. A stílusnak egy adott elemhez történő hozzárendelésével. Akkor alkalmazzuk, ha egy stílusutasításra csak egy adott helyen van szükség. Ekkor a HTML kódba a `style` paraméter értékeként deklaráljuk a stíluselemeket. Ezt mutatja a következő példa:

```
<p style="font: normal 12px arial" >Bekezdés 12 pontos normál Arial betűvel</p>
```

3. Külső fájlként megírt stíluslapra történő hivatkozással. Ekkor a stílusutasításokat egy .css kiterjesztésű szöveges fájl tartalmazza. Ebben nem használhatunk HTML kódot és nem szükséges a <style> és </style> elem pár sem. Ezt a fájlt a HTML dokumentum fejrészában elhelyezett link elem fogja meghívni és a beírt stílussorok szerint fogja felépíteni a dokumentum stílusát. Ezzel a módszerrel valóban kettéválik a tartalom (HTML) és a megjelenés (CSS). A következőkben látható példaként egy CSS állomány és egy ezt felhasználó HTML állomány.

stilus.css

```
table {
font-family: Garamond;
color: orange;
font-weight: bold;
border: 3px dotted blue
}
```

Táblázat stílusleírása:

= *Garamond betűtípus*
 = *narancs színű betű*
 = *félkövér betű*
 = *3 pixeles kék szaggatott keret*

oldal.html

```
<html>
  <head>
    <link rel="stylesheet" href="stilus.css"
      type="text/css">
  </head>
  <body>
    <table>
      <tr><td>Szegélyezett szöveg</td></tr>
    </table>
  </body>
</html>
```

← *stíluslap meghívása*

Az eredmény:

 Szegélyezett szöveg

4. Külső stíluslapok importálásával. Egyidejűleg több stíluslapot is importálhatunk, valamint alternatív stíluslapokat megadhatunk, a felhasználóra bízva a választást. A megjelenítés helyétől függően különböző stílusok alkalmazására van lehetőség, például a nyomtatási stílus teljesen különbözhet a képernyőn megjelenő változattól. Amennyiben ugyanazon elem tulajdonságát több utasítás szabályozza, a következő prioritás érvényesül:
 - ha nincs egyéb utasítás, a külső stíluslap szabálya érvényesül,
 - ha a HTML oldalban van saját stílusutasítás blokk, illetve elem tulajdonságaira és külső stíluslap is szabályozza, úgy az utóbbit állítja be a böngésző,

- ha a HTML oldalban az elemre soron belüli stílus meghatározás van, ez felülírja a külső stíluslap szabályát.

Az előzőek alapján megállapítható, hogy a stíluslapok használata számos előnnyel jár a webszerkesztő és a használó számára. Ezek közül a fontosabbak a következők:

- Egy honlap több oldalához kapcsolódhatnak azonos stílustulajdonságok, ezáltal a formázást csak egyszer kell megtenni. Az eredmény az egységes dokumentum formátum lesz.
- Több lap vagy akár egy teljes webhely stílusait egy helyen lehet tárolni, így gyorsan és könnyen frissíthető.
- A dokumentum mérete és komplexitása csökken, mivel nem tartalmaz információkat a megjelenítéshez, ezáltal gyorsabban letöltődik (különösen, ha több oldalon használunk azonos stíluslapot), és mivel a böngészők gyakran tárolják a CSS stíluslapokat a gyorsítótárban, ezért a hálózati forgalom is jelentősen csökkenhet.
- Ha utólag változtatnunk kell a megjelenésen, csak egy helyen kell ezt megtennünk, s minden oldalon meg is változik.
- Mind a HTML, mind pedig a CSS kód sokkal átláthatóbb lesz, ami által könnyebbé válik a weboldalak karbantartása.
- Sok olyan formázási lehetőséget biztosít, amik csak HTML-el egyáltalán nem oldhatók meg.
- Különböző felhasználókhöz különböző stílusokat lehet rendelni: például könnyen olvasható, nyomtatható stílusok.

5. WEBOLDALTERVEZÉS

A tervezés szakaszában javasolt számos – az adott témához illeszkedő – oldal megtekintése, melyekből sok-sok ötletet meríthetünk, esetleg elkerülendő hibás megoldásokat ismerhetünk meg.

A legalkalmasabb domain név kiválasztása és lefoglalása nagy jelentőséggel bír, hisz ez lesz az a cím, amit a böngészőbe beírva a látogatók elérhetik a weboldalt, valamint a céges e-mail címek is ezt a domain nevet viselik. A domain név kiválasztásakor arra kell törekedni, hogy az kifejező és könnyen megjegyezhető legyen, és a szóbeli átadása se okozzon nehézséget (pl. telefonon történő honlap vagy e-mail cím megadásakor). A névben kerüljük az ékezetes betűk használatát, mert ezeket több böngésző még nem támogatja. Arra is figyelemmel kell lenni, hogy ékezetek nélkül a domain név hogyan cseng, és van-e olyan megváltozott tartalma (pl. vicces, trágár stb.), amely miatt elvetendő. A választott domain név lefoglalhatóságáról (szabad / foglalat) a <http://www.domain.hu/domain/szabad-e/> címen tájékozódhatunk. [6]

A honlapot webszerveren (kiszolgálón) kell elhelyezni. A szerverek a nap 24 órájában üzemelnek és az interneten keresztüli elérésük folyamatos. Webtárhelyet erre szakosodott cégektől bérelhetünk. Az szolgáltatás feltételeinek részleteit jelentik például az igényelt tárhely mérete, szükséges e-mail fiókok száma és mérete, szkript és php futtatási lehetőség, adatbázis lehetőségek stb. Amint a webhely szerkesztett oldalai elkészülnek, elkezdhetjük az előkészített szöveges, képi és egyéb tartalmak feltöltését, ellenőrzését, esetleges módosításokat, javításokat.

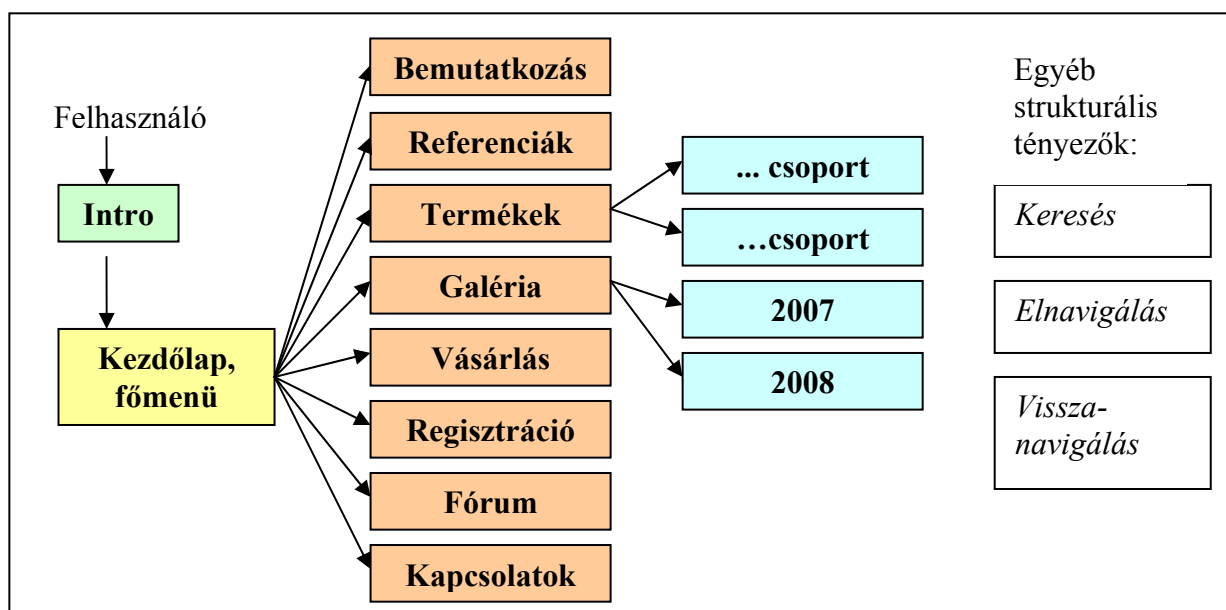
A honlap akkor funkcionál, ha a látogatók megtalálják, vagyis biztosítanunk kell a jelenlétét és elérhetőségét. Annak érdekében, hogy minél több látogatónk legyen, a webhelyet regisztrálnunk kell a különböző kereső programokban (pl. Google, Yahoo!, Heureka stb.), katalógusokban, linkgyűjteményekben (pl. lap.hu). A webhely látogatottságát jól mérhetjük az oldalakra telepített látogatást figyelő szkriptekkel. Ezek egy független szerveren gyűjtik a látogatottságot jellemző statisztikai adatokat, melyek az oldal tulajdonosa számára elérhetőek.

5.1. Tartalomtervezés

Az új látogatók informálása céljából a bejelentkező oldalon (vagy onnan közvetlenül elérhetően) tájékoztatni kell a honlap tulajdonosáról, aki általában egyben felelős is a tartalom

közléséért. Fontos, hogy aki nem ismeri a honlap tulajdonosát, (mert például keresővel jutott oda, vagy csak a címet kapta meg) olyan információkhoz jusson, amiből eldöntheti, hogy érdekli-e egyáltalán a honlap, érdemes-e neki tovább navigálni a belső oldalakra. Cég vagy egyéb szervezet honlapja esetén akkor jó a bemutatkozás, ha megismerjük a cég rövid történetét, profilját, filozófiáját, a honlap létrehozásának világosan meghatározott célját, a webhely egyértelműen meghatározott profilját, a megtalálható információk egyértelmű leírását, valamint a honlap fenntartásáért felelős szervezetet (ha az nem a honlap tulajdonosa).

A webhely egyik legfontosabb tartalmi szempontja, hogy kinek legyen hasznos a honlap, és mennyiben szolgálja a látogatói célcsoport igényeit. A tartalom legyen tárgyhoz tartozó és lényeges. A számítógépvásárlást tervező számára például fontos lehet, hogy a korszerű hardver és szoftver kínálatról, valamint ezek paramétereiről – összehasonlítva más gyártók és kereskedők termékeivel – egy helyen jusson információhoz. A kitűzött célok általában a következők lehetnek: marketing, értékesítés, új vevők szerzése, kapcsolattartás a vevőkkel, termékkövetés, szolgáltatásokhoz kapcsolódó segítségnyújtás, információk közlése a cégünkről stb. Az oldalakon közölt információnak a célcsoport számára közérthetőnek, célszerűen rendezettnak és elegendőnek kell lennie. A cél meghatározása után határozhatók meg azok a tartalmak, amelyek alapján elkészíthető a honlapot alkotó oldalak struktúrája, a navigálást segítő menüpontok, a körvonalazhatók a szöveges és képi tartalmak. A következő ábra egy honlap lehetséges struktúráját mutatja be.



6. ábra: Egy egyszerű példastruktúra

Az ábrán jól megfigyelhető a funkciók, illetve az ezeket megvalósító oldalak hierarchikus kapcsolódása. Az egyes oldalakat a tárhelyünkön érdemes önálló mappával ellátni és a honlap struktúrájának megfelelő mappaszerkezetet kialakítani a kellő átláthatóság miatt. A mappák és a fájlok elnevezésekor nem használhatunk hosszú ékezetes betűelemeket, vagy ha mégis szeretnénk, akkor a megfelelő helyettesítő karaktereket kell használnunk (pl. ő=ő ü = ű stb.). A lapunkat megtekintő felhasználónak természetesen biztosítanunk kell a szabad elnavigálás lehetőségét is.

A webhelyen közölt tartalomnak azon túl, hogy hasznosnak és korrektnek, kellően meggyőzőnek is kell lennie ahhoz, hogy a látogatók hitelesnek tekintsék. Ezt szolgálhatja a szöveges információk megfelelő stílusa. Törekedni kell a képek informatív használatára. Meg kell jelölni azt is, hogy amennyiben további információk szükségesek a hitelességünk növeléséhez (pl. tanúsítványok, versenyen elért eredmények stb.), azok hol szerezhetők be.

Amennyiben a megcélzott látogatói csoport egy szegmense nem beszéli a magyar nyelvet, úgy többnyelvűvé kell tenni az oldalakat, de legalább biztosítani kell a minimális szintű érthetőséget több nyelven is (pl. legalább a címeket, menüket, rövid tartalmi összefoglalókat fordítsuk).

A célt szem előtt tartva kell összegyűjteni és elkészíteni a weboldalra feltöltendő szöveges, képi, audio, videó stb. anyagokat. Az anyaggyűjtést ajánlott előbb elvégezni, mert a honlap kivitelezése közbeni módosítgatás, vagy a későbbi átalakítások felesleges és elkerülhető munkával járnak.

A honlappal kapcsolatban talán a legfontosabb kérdés, hogy hogyan hat a látogatókra, és arra készíti-e őket, hogy visszatérjenek. Vannak-e olyan elemek, amelyek negatív hatást váltanak ki a felhasználóban. Alapvetően ezek határozzák meg a honlapunk sikerességét.

5.2. Interaktivitás

A honlap által biztosított interaktivitás azt jelenti, hogy a látogató nem csak passzív szemlélője a közzétett tartalomnak, hanem olyan funkciókat is elérhetővé teszünk az oldalon, amely a felhasználó aktív közreműködését kívánja. Ezeknek a funkcióknak a honlap létrehozásának célját (reklám, marketing, szolgáltatás, ismeretterjesztés stb.) kell szolgálniuk. Lehetőséget adunk a látogatónak a kapcsolatteremtésre, a kommunikációra (pl. információk kérésére, véleményének kifejtésére, javaslatainak megtételére). Adott esetben ösztönözzük a

felhasználót az információk megosztására, kérdésfelvetésre, esetleg vitára (pl. fórum). Interaktivitást szolgáló szerkesztési eszközként gyakran használjuk az űrlapokat. Az interaktív eszközök használatára esetenként megfelelő eszközökkel ösztönözni kell a felhasználókat (pl. egy fogyasztói szokásokat felmérő kérdőív kitöltésére valamilyen nyeremény kilátásba helyezésével).

Egy termékeket bemutató honlap értékét növeli, ha arra vonatkozó adatokat is tartalmaz, hogy hol szerezhető be az adott termék, ki a termék személyes területi képviselője, esetleg linket tartalmaz a forgalmazó oldalára. Ügyelnünk kell arra, hogy a honlapon közzétett elérhetőségek pl. levélcím, székhely, telephely, e-mail cím, telefon, fax stb. valóban megtalálhatók legyenek, pontosak legyenek. Tájékoztatnunk kell, hogy speciális kérdésekkel (műszaki, technológiai, pénzügyi, jogi stb.) kihez fordulhat a látogató. A bizalmat erősítheti, ha a kapcsolattartók fényképes bemutatkozása is megtalálható az oldalon.

Biztosítani kell a felhasználó számára, hogy az oldalunkon megtalált számára hasznos információt megoszthassa másokkal is. Ennek egyik lehetősége, ha a lapot vagy annak egy tartalmi egységét elküldheti egy általa megadott e-mail címre, vagy a tartalmat praktikus formában egyszerűen kinyomtathatja.

A célszerűen szervezett navigációval, jól elhelyezett linkekkel lehetővé kell tenni a látogató számára, hogy könnyen (adott számú kattintással) eljuthasson az általa preferált tartalmat (pl. műszaki cikkek, hétvégi programok) prezentáló oldalakra. A honlaphoz készített kereső eszközzel kulcsszavak alapján juttathatjuk el a felhasználót a kívánt tartalomhoz.

5.3. Oldaltervezés

A webdesign kialakításakor figyelembe kell vennünk, hogy az általunk megjelenített tartalmat több különböző eszközön, környezetben és módon fogják olvasni a látogatók. Néhány látogatónk talán a mobiltelefonján kívánja elolvasni az oldalunkat, valaki esetleg ki akarja nyomtatni, és az is lehet, hogy rosszul lát, és emiatt fel kell nagyítania az oldalakat a jobb olvashatóság érdekében. Ilyen okok miatt a weblap tervezésekor a böngészőnek csak útmutatásokat tudunk adni, hogy milyen módon szeretnénk megjeleníteni a tartalmakat. A webdesignerek persze mindezeket figyelmen kívül hagyva gyakran megpróbálnak ráerőltetni egy design. Ennek egyik módszere az, amikor a szövegeket képként helyezik el fix keretekben.

5.3.1. Oldalszerkezet kialakítása és design

Kiemelkedő tervezési szempont a honlap megjelenése, arculata (design), és különösen fontos az első benyomás, amit a bejelentkező oldal a felhasználóban kelt. Jelentős szerepet játszik abban, hogy a látogató elolvassa-e az oldalt vagy sem. Fontos, hogy a színek harmonizáljanak, kellemesek és nyugtatóak legyenek és igazodjanak a tartalomhoz. Ez adja meg azt az alapot, amelyben a szép képekkel, fotókkal, ötletes animációkkal kiemelhetjük, kihangsúlyozhatjuk a fontos tartalmakat.

A design szempontjából meghatározó a szöveg megjelenése. Fontos, hogy a betűtípus és a betűk színe az oldal stílusával harmonizáljon, megfelelő méretű legyen, biztosítva ezzel a könnyű olvashatóságot. Nem szerencsés a kurzív (kézíráson alapuló) betűtípusok használata. Az előbbieket összegzéseként megállapítható: különösen fontos, hogy a honlapot alkotó minden oldalnak egységes elvek alapján megtervezett és egyéni arculata legyen. Nem lehet azonban a design öncélú, vagyis nem ronthatja a weboldal alapfunkcióit és paramétereit, például nem gátolhatja a navigációt, nem lassíthatja indokolatlanul a letöltést. Az egyedi arculat kialakítását nagyban felgyorsítja, ha már van egy kidolgozott cégarculat terv (logó, dokumentumsablon, választott színek, egyéb jellemzők), amely rendkívül jó kiindulópontul szolgálhat.

A kezdőoldalon ajánlott tudatni a látogatóval, hogy a honlap oldalait milyen monitorfelbontásra terveztük és milyen böngészőhöz optimalizáltuk. Ennek alapján már ekkor átkapcsolhatja gépét megfelelő felbontásra, illetve átválthat más böngészőre (amennyiben többet is feltelepített a gépére).

A honlap használhatóságának fontos tényezője a felépítés, a szerkezet és ezen belül a menü szerkezet. Biztosítani kell, hogy a látogató könnyen eljusson a tartalom megfelelő eleméhez, ami a könnyen áttekinthető honlap szerkezettel és a jó navigálhatósággal érhető el. A többszintű hierarchikus oldalszerkezetű honlap akkor jó felépítésű, ha a belső oldalakon járva a képernyőn is jól kitűnik, hogy hol jár a látogató a struktúrában, melyik menüpontból jutott oda, és hová tud tovább-, illetve visszalépni. Egy adott oldalon belül könyvjelzők elhelyezésével kell biztosítani a könnyű navigálás lehetőségét. Új ablak csak nagyon indokolt esetben kerüljön megnyitásra a navigáció során, ugyanis a túlzottan sok megnyitott ablak átláthatatlan és kezelhetetlen.

Honlaptérkép (oldaltérkép, sitemap) készítésével a tapasztaltabb felhasználók számára nyújthatunk segítséget a tartalmak struktúrájában való eligazodásban és a navigálásban. A jó

menüszerkezet és honlap térkép alapján látszik a webhely teljes szolgáltatásválasztéka, tehát nem csak véletlenre bízunk, hogy a látogató rátalál-e a kívánt tartalomra. Kiegészítő funkciókkal, például keresés az oldalakon vagy listákban (terméklista, könyvek stb.), tárgymutató és egyéb lehetőségek beépítésével segíthetjük a navigációt vagy az információ megtalálását.

Az információ értékét jelentősen befolyásolja annak aktualitása, frissessége. Amennyiben egy honlap nincs megfelelő rendszerességgel frissítve, akkor az onnan nyert információ már nem új ismeret, hanem ismert tény lesz, vagy már téves információ. A honlap rendszeres karbantartásával biztosítani kell az információ naprakészségét. Erre mind az új, illetve a visszatérő látogatók, mind pedig a keresőgépek (Google, Yahoo! stb.) igényei miatt van szükség. Sok esetben egyes információk akkor is fontos lehetnek, ha már régiek, de ekkor az archívumban a helyük. A folyamatos és könnyű, HTML ismeret nélküli frissítését teszi lehetővé, ha az oldal tartalomkezelő rendszerre épül.

A weblap tartalmának elrendezéséhez táblázatokat vagy kereteket használhatunk. Kereteket csak akkor célszerű alkalmazni, ha a webhelyhez tartozó oldalak több azonos részeket tartalmaznak. Ekkor több keretre fel kell bontani az oldalt és a fix tartalmak egyszeri leírása után külön-külön megírjuk a változó tartalmakat hordozó kereteket. A táblázatok a tartalom formai tagolásán túl az oldalak szélességi méretezésben is fontos szerepet játszanak. A táblázatokat nem feltétlenül kell szegélyezni, ezáltal ez a szerkezet rejtve maradhat. Több táblázat egymásba ágyazására is van lehetőség. Azt figyelembe kell venni, hogy a magasság méretének beállítását csak az Internet Explorer tudja értelmezni. [10]

A képek beszúrása során a célszerűséget kell figyelembe venni. A fölösleges mennyiségben és túlzott méretben beszúrt képek akadályozhatják a böngészés folyamatát, a lényegi információk elérését és zavaró mértékű várakozásra kényszerítheti a felhasználót a letöltésük, ami által a weboldalak elérésének sebessége is romlik.

5.3.2. *Webes tipográfia*

A webes tipográfia számos korlátozást tartalmaz, mivel olyan típusokkal és elrendezéssel kell dolgozzunk, amelyről sejtjük, hogy használható lesz azokon a gépeken is, amelyekeken majd a látogatók megnyitják a lapot. Ilyen korlátozást jelent például a korlátozott betűkészlet, az elválasztás és alávágás lehetőségének hiánya, és akkor még a nem szóltunk a képernyő méretéről, a felbontástól és a környezetről sem, amelyek szintén befolyásolják a

megjelenést. A webes tipográfiát javasolt teljes mértékben a CSS segítségével kialakítani, amely sokkal több és rugalmasabb lehetőséget biztosít, mint a HTML kód.

Ne alkalmazzunk funkció nélküli tipográfiai megoldásokat, mert ezek esetleg pontosan a tartalomtól vonják el a látogató figyelmét. Az alkalmazott színek és a tipográfia funkcionalitása szintén indokolt legyen. A webhely színvilága, tipográfiája és arculata alkosson egységet és igazodjon a témához. [11]

5.3.3. A honlap akadálymentesítése

Az akadálymentes honlap szerkesztése során figyelemmel kell lenni minden internetező igényeire vagyis arra, hogy az információ mindenkinek eljuthasson. Ezt úgy tudjuk biztosítani, hogy a tartalmat több érzékszervvel is érzékelhetővé kell tenni. A legspeciálisabb internetező csoportot a látássérültek alkotják. Az auditív és vizuális (grafikai és képi) tartalmat annak szöveges megfelelőjével is biztosítani kell a látogatók számára. Például egy vezérlő ikon esetén annak funkcióját, egy kép esetén annak címét és rövid leírását külön mezőben (`alt` attribútummal) szövegesen közölni kell. A vakok az oldalak tartalmát speciális felolvasó program segítségével ismerik meg. A videofájlokhoz, grafikonokhoz is ajánlott a szöveges leírás elkészítése. Leghelyesebben akkor járunk el, ha a navigációs szerkezetet, a menüket és linkeket nem képi, hanem szöveges megjelenítésben készítjük el. A tartalom minden elemét elérhetővé kell tennünk a vakok számára is, akik nem tudják használni az egeret. Ezért tehát minden funkciót elérhetővé kell tennünk a billentyűzetten keresztül is. Kikapcsolhatóvá kell tenni a dinamikus oldalakon az interakciók időkorlátját, és leállíthatóvá kell tenni a villogó és mozgó tartalmakat. Kerülni kell az epileptikus rohamokat előidéző hangeffekteket.

A gyengénlátók számára biztosítani kell, hogy az előtérben lévő szavak és képek kiemelkedjenek a háttérből, ezáltal jól olvashatóak legyenek és kerülni kell a bonyolult grafikai megjelenítéseket. A relatív betűméret használata javasolt, vagy biztosítani kell egy olyan kaput az oldalra, ahol nincs méretkorlátozás.

Gondolnunk kell színlátási rendellenességgel rendelkezőkre is, ezért fekete-fehérben is nézzük meg a honlapunkat (a monitoron a színintenzitás leszállításával) és szükség esetén módosítsunk rajta. Lehetőség szerint kerülni kell a flash intrót és a JavaScriptet, viszont ahol csak lehet CSS-sel történő pozicionálást javasolt használni.

A tartalom megfogalmazása során törekedni kell a helyes szóhasználatra, a nyelvtani szabályok betartására. A tartalmat következetesen kell elrendezni az egyes lapokon és egységes felhasználói felületet célszerű biztosítani.

A webhely akadálymentesítettségének fokáról könnyen meggyőződhetünk, ha beírjuk az URL-t a <http://www.paramedia.hu/gyorsteszt.html> oldalon elérhető gyorstesztbe. [13]

5.4. A szerkesztés általános szempontjai

A honlapokkal szemben támasztott legáltalánosabb szempontként a felhasználóközpontúság emelhető ki. Ez tételesen azt jelenti, hogy az vegyük figyelembe a felhasználó igényeit, biztosítsuk a könnyű használhatóságot és fogadjuk a felhasználói értékeléseket. A felhasználói visszajelzéseket felhasználva folyamatosan tökéletesítsük az oldalakat.

A honlap készítése során tekintetbe kell vennünk a szellemi tulajdonnal és az adatvédelemmel kapcsolatos jogszabályi rendelkezéseket, etikai normákat (netikett szabályait) és világosan közölnünk kell a honlapnak és tartalmának a használatára vonatkozó feltételeket.

Biztosítani kell, hogy a használók könnyen megtalálják az igényüknek megfelelő tartalmat és szolgáltatást, egyszerűen az átjárhatóságot. Ez a jól elhelyezett linkeken is múlik. Fontos, hogy a linkek (ugrópont, könyvjelzők) értelmes jelentést hordozzanak, különben az „Erről [itt](#) olvashat bővebben” vagy a „Kattints [ide!](#)” típusúak olyan helyre mutathatnak, amelyek csak hosszas keresgélés után vezetnek el a kívánt helyre. Nem beszélve arról, hogy az oldal nyomtatása esetén semmilyen támpontot nem jelentenek.

Olyan címet kell adni, amely nem csak az adott szöveggörnyezetben értelmezhető (pl. „Története” helyett „A HTML története”), valamint kerülendők a túl hosszú méretezett címek is. Nem keverendő össze a dokumentum címe (amelyet a `<title>` címkével jelölünk a fejlécben) a szövegben szereplő fő- és alcímekkel. A dokumentum címéből csak egy lehet, amely nem része, hanem jellemzője a teljes dokumentumnak, és annak kontextus független azonosítására szolgál. A böngészők többsége ezt az ablak fejlécében jelenítik, és az oldalra mutató könyvjelző is általában ezt a címet kapja.

A weboldal kereső-optimalizációja nagyon fontos, mert a forgalom döntő részét (általában 60-90%-át) a keresőknek köszönhetjük, vagyis általuk találunk az oldalunkra. A keresők mint indexelő rendszerként működnek. Minden oldalhoz rendelünk néhány jellemző kifejezést, amely a tartalomra utal. Ezek közül a legfontosabb kifejezéseket írjuk be HTML

kód esetén a `<title>...</title>` és bővebben a `<meta content="..." />` elemek közé (a kipontozás helyére). Ezek alapján találunk majd ránk a keresők, és ezáltal a felhasználók is.

Kereteket csak indokolt esetekben használjuk, mivel alkalmazásuk esetén az információ csak lépések sorozatával érhető el. Nagy probléma az is, hogy a keretek tartalmára nem lehet egyszerűen hivatkozni, csupán a keretszerkezet nyitóoldalára, és ekkor kérdés, hogy a kívánt tartalmat megtaláljuk-e. Egyéb kellemetlenségeket is okoz, például a képernyőfelület kihasználása romlik, nyomtatásban kifogásolható (hiányos, szétesett) lehet az oldal. Amennyiben elkerülhetetlen a keretek használata, feltétlenül törekedni kell az URL-ek használhatóságának megőrzésére, a célkeret teljes képernyőre történő beállításával (pl. HTML-ben `<a href="egy_aloldal" target="_top">`).

A webhelyünkön csak olyan technikákat szabad alkalmaznunk, amelyek függetlenek a látogató paramétereitől. Ebben az esetben a funkciók elérhetősége nem függ a felhasználó környezeti (hardver és szoftver) feltételeitől. Website-unk nem kötődhet szorosan egy konkrét böngészőhöz, de amennyiben mégis, ezt a mindenki által elérhetővé tett kezdőoldalon jelezni kell. (Ebben az esetben a látogatók egy részéről le kell mondanunk.)

A technikai megvalósítást során olyan stratégiát és szabványokat (ajánlásokat) kell követnünk, amelyek biztosítják a honlapnak és tartalmának tartós, hosszú távú megőrzését.

5.5. Kódolási szempontok

A magyar nyelv sajátos ékezetes karakterei miatt nagyon fontos, hogy a weboldalakban a megfelelő (ISO-8859-2 vagy UTF-8) karakterkódolási szabványt alkalmazzuk. Ezt általában a leírónyelvi kód elején kell megadni.

Arra kell törekednünk, hogy a HTML forráskód legyen áttekinthető, a kódban gyakran használjuk a `<!--` és `-->` jeleket megjegyzések beszúrására annak érdekében, hogy a későbbiekben egy-egy módosítás alkalmával a kódot könnyen vissza tudjuk fejteni.

Az egy könyvtárszerkezetben lévő weblapjaink között javasolt a relatív, kívülre az abszolút linkek használata. Az oldalakon elhelyezett hivatkozásainkat – különösen a külsőket – gyakran szükséges ellenőriznünk.

Mivel a böngészők szabványosságá nem hagy némi kívánnivalót maga után, teszteljük oldalainkat többféle böngészővel és operációs rendszerrel. Amennyiben a hibátlan

megjelenítéshez különleges programokra van szükség, ezt jelezni kell még az előző oldalon annak érdekében, hogy a felhasználó erről időben értesüljön (pl. JavaScript, valamilyen speciális vezérlő stb.), illetve ennek hiányában a tartalom helyett valamilyen üzenet tájékoztassa a felhasználót a problémáról és az így számára rejtve maradt tartalomról. Jelezzük az oldalon, ha speciális böngészőre van szükség a hibátlan megjelenéshez.

Kép beillesztése esetén a HTML kódba írjuk be a képek méretét, ami által a lap szerkezete már akkor a végleges formát kapja, amikor a képek még nem töltődtek le. Kisebb méretű mintaképek létrehozásához ne a magasság (height) és szélesség (width) paramétereket használjuk, mert ekkor az egész képet le kell töltenie a böngészőnek. Ilyen esetben egy kisebb méretű (miniatűr) kép beágyazása javasolt. Ne dolgozzunk kizárólag képekkel, vagyis a szöveges tartalmat szöveg formátumban használjuk és ne képként. A fontos tartalmakat kiegészíthetjük villogó elemekkel, gif és fash animációkkal.

Amennyiben objektum (szöveg, kép, hang, videó stb.) letöltését biztosítjuk az oldalon, fel kell tüntetnünk annak a jellemzőit (pl. helyfoglalási mérete, felbontása, kódolása) is még a művelet megkezdése előtt.

A szöveg formázása során nagyobb üres helyek létrehozásához nem a BR, vagy P elemek többszöri ismétlése javasolt, hanem a PRE elem üres sorai. Számolnunk kell azzal, hogy az előre formázott szövegben esetén a félkövér, vagy dőlt karakterek nem biztos, hogy minden böngészőben ugyanúgy jelennek meg. Kiemelésre ne használjuk az aláhúzást, mert ez konvencionálisan általában linket jelöl.

Néha érdemes kétféleképpen is biztosítani egy adott formázási funkciót, például a középre igazítás: `<center><div align="center">Középre igazítva</div></center>`

A honlap témájához vezető linkgyűjteményt helyezhetünk el webhelyünkön, amennyiben a cégünk, szervezetünk pozícióit erősíti vagy javítja megítélését. Nagyon fontos követelmény, hogy az elhelyezett linkek „éljenek”, és kívánt mértékben aktuálisak legyenek. A törölt linkek rossz hírét kelthetik honlapunknak. Az oldalak karbantartása során ezeket is rendre vizsgálni szükséges.

A látogatók preferálják a stabilan működő webhelyeket, és elkerülik a böngészés során hibaüzeneteket megjelenítő oldalakat. A felhasználók nem szeretik azt sem, ha az oldalon a következő üzeneteket olvashatják: „Feltöltés alatt”, „Elnézést, még nem működik”, „Hamarosan elkészül” stb.

6. WEBLAP-SZERKESZTŐ PROGRAMOK

A webszerkesztő programok két nagy csoportba sorolhatók: léteznek a kódszerkesztők, valamint az úgynevezett vizuális szerkesztők. A **kódszerkesztők** – melyek formázás nélküli ASCII szöveget állítanak elő (text editor-ok) – a *Jegyzettömb* alkalmazásra hasonlítanak, azonban több olyan kényelmi funkcióval is el vannak látva, amely a szerkesztőt segíti munkájában. Ilyen kényelmi szolgáltatások például a műveletek több lépésben történő visszavonása, a kódrészletek eltérő színezésével a kód áttekinthetőségének javítása, a sorok számozása, a kódösszevonás, az automatikus eltolás stb. A kódszerkesztőkben a fejlesztőnek kell kézzel megírnia sorról sorra a weboldalt leíró HTML-kódot. Vannak olyan kódszerkesztő programok is, amelyekben párbeszédablakokon keresztül adhatjuk meg bizonyos HTML elemek, objektumok paramétereit, majd a program maga generálja a kódot, amely később szükség szerint javítható (Macromedia HomeSite 5.0, CuteHTML, Ace HTML, Evrsoft FirstPage 2000). A bonyolult, tartalmas és mutatós weblapok készítésére azonban kényelmetlen a karakteres szövegszerkesztők használata, mert sok munkát és figyelmet igényel a kód pontos, szintaktikailag helyes begépelése.

A **vizuális szerkesztők** a grafikai alkalmazásokra hasonlítanak. A weboldalt vizuális eszközökkel, WYSIWYG módon, mint egy szövegszerkesztőt használva tudjuk létrehozni, míg a program eközben legenerálja az ehhez szükséges forráskódot (Microsoft FrontPage, Macromedia Dreamweaver MX, Netscape Composer stb.). Ezeknek a programoknak a könnyű kezelhetőség az erényük. A profi weblap-szerkesztők nem használják ezeket az alkalmazásokat, mivel a többségük kódgenerálása áttekinthetetlen és problematikus, amely jelentős eltérésekhez vezethet a különböző böngészőkben nézve. A weblapszerkesztésben valamennyire jártas készítőnek ajánlott megoldás: az oldalt vizuális szerkesztővel viszonylag rövid idő alatt össze lehet állítani, majd egy egyszerű text editorral kell elvégezni kódszinten a finomításokat.

Fontos megemlíteni, hogy a nagy böngészőgyártók (Microsoft, Netscape) két irányba fejlesztik alkalmazásaikat. A Netscape a Java felé nyitott, a Microsoft viszont saját megoldásait alkalmazza és fejleszti. Az eredmény pedig a két cég szoftvereinek inkompatibilitása. Az alapvető eltérés a JavaScript és ActiveX elemek támogatása terén jelentkezik. A kompatibilitást szerveren futó alkalmazások telepítésével lehet biztosítani, amely független a böngészőtől, futtatása pedig a webszerveren történik.

Bármilyen szerkesztővel is készítjük a weboldalt, a végeredményt egy általában .html vagy .htm kiterjesztésű, normál szövegfájlba kell menteni. A webhely első betöltődő állománya általában az index állomány (index.html). Erről az oldalról indulhat a böngészés, általában menürendszer segítségével.

A weblapok nagy változásokon mentek keresztül. A kezdeti egyszerű lapokat felváltották a multimédiás képességekkel felruházott weblapok. Ezen az evolúciós időszakon átestek a weblapszerkesztő programok is. A továbbiakban néhány, a használók között elterjedt szerkesztőprogram bemutatása következik.

6.1. Notepad / Jegyzetömb

Egy általános weblap egy egyszerű karakteres szövegszerkesztővel is elkészíthető. Erre a célra a Windows Jegyzetömb alkalmazása is jól használható, amely egyszerűbb dokumentumok készítéséhez használatos alapszintű szövegszerkesztő. Az alkalmazás csak a legalapvetőbb formázási műveleteket teszi lehetővé, ezért a tiszta szöveges formátumot igénylő dokumentumokba nem kerülhet különleges formázás. Ez a tulajdonsága különösen hasznos a weblap céljára készített dokumentumok írásakor. A szerkesztővel készített állományok Unicode, ANSI, UTF-8 és fordított bájtrendű formátumban menthetők.

Weblapszerkesztéskor be kell gépelni a megfelelő szerkezetben és szintaktikával a tageket, és html formátumban mentve az oldal egy böngészővel akár azonnal megjeleníthető. Komoly felkészültség és gyakorlat kell ahhoz, hogy egy bonyolultabb oldalt ilyen módon létrehozzunk. Ehhez a munkához ismerni kell az összes HTML vezérlőelemet és az oldal felépítése is sokkal több időt fog igénybe venni, mintha valamelyik webszerkesztésre kifejlesztett szoftvert használnánk.

6.2. NVU

Open Source fejlesztés, amely a leírása szerint konkurens lehet a Dreamweavernek és a FrontPage-nek. Elnevezése az „n-view” vagy „new view” kifejezéshez kötődik, magyarul új nézetet, új nézőpontot jelent. Az NVU történetében több verziót élt meg első megjelenése óta.

Alkalmazása mellett számos előnyös tulajdonsága szól. Az NVU ingyenesen letölthető és felhasználható, többplatformos alkalmazás, melynek számos nyelvi lokalizációja (a magyar is) és forráskódja is elérhető. Egyszerű a kezelőfelülete, jól használható súgóval rendelkezik és nem igényli a HTML tudást. Jelölőnyelv tisztítót és beépített forráskód ellenőrzőt tartalmaz, és természetesen W3C szerint érvényes (valid) kódot generál. CSS szerkesztője is van és támogatja a PHP kódok beszúrását is.

6.3. Microsoft FrontPage

A vizuális szerkesztők csoportjába tartozik, vagyis amit szerkesztés közben a képernyőn látunk, az jelenik meg majd a böngészés során is. Ez persze csak akkor igaz maradéktalanul, ha az Internet Explorer legújabb változatát használjuk, köszönhetően annak, hogy mindkét szoftver Microsoft termék. Emiatt óvatosan kell bánni a böngésző specifikus elemek használatával. HTML és DHTML állományok készítésére alkalmas. Jelenleg legfrissebb változata 2007-ben jelent meg, és ez az új verzió az MS Expression Web 2007 nevet kapta. Az Expression Web egy szabványokra épülő webhelyek létrehozásához kifejlesztett professzionális tervezőeszköz.

A FrontPage a többi szerkesztőprogram szolgáltatásaihoz viszonyítva annyival nyújt többet, hogy lapok szerkesztésén túl segítségével a webhelyet is felügyelhetjük, vagyis például ellenőrizhetjük a hiperhivatkozásokat, a közzétételt szabályozhatjuk. A programmal lehetővé válik a keretek és stíluslapok létrehozása és kezelése is. Felülete a már jól megszokott MS Office kezelőfelület, így aki ismeri az MS Word szövegszerkesztőt, az a FrontPage szerkesztővel is könnyen megbarátkozik. A grafikus szerkesztőfelület mellett a generált kód is megtekinthető és a HTML nézetben szerkeszthető.

Továbbfejlesztett tervezési lehetőségeket nyújt az igényesebb megjelenésű webhelyek létrehozásához. A webhely elrendezésének gyors, könnyű és pontos vezérlésében segítségünkre van. A legjobb eredmény eléréséhez az általunk kívánt grafikai eszközökkel való tervezést támogatja. Olyan eszközök használatát is engedélyez, melyek a webhelyet elérhetőbbé teszik, és biztosítják, hogy a látogatók megtekinthessék munkánkat. A FrontPage 2003 szerkesztővel lehetőségünk van a HTML hatékonyságának növelésére, az interaktív oldalak beépített parancseszközökkel való létrehozására és a gyorsabb, szabványosabb és pontosabb kódolásra a professzionális kódoló eszközök használatával. A szoftver lehetőséget

nyújt számos közzétételi eszköz közül választásra és a webhely hatékonyságának növelésére, mivel új lehetőségeket kínál a kapcsolatok és az információszerzés területén.

A szerkesztett oldalakat és a megnyitott webhelyet hatféle nézetben jeleníthetjük meg, amely nézetek a szerkesztő munkáját könnyítik. Menüvel szabályozható, hogy a nézetváltást segítő sáv látható legyen-e a képernyő szélén. A nézetek a következők lehetnek:

1. *Lap nézet*: a weblapok létrehozására, szerkesztésére és megtekintésére szolgáló nézet. Az adott lap háromféleképpen jeleníthető meg: *Normál* (WYSIWYG szerkesztő), *HTML* (a kódot mutatja), *Minta* (böngésző program jeleníti meg).
2. *Mappák*: a webhely-tartalom rendszerezését (létrehozását, másolását, áthelyezését, törlését) teszi lehetővé.
3. *Jelentések*: a webhely tartalmának elemzését (fájlok csoportosítását, kapcsolati feltérképezését stb.) teszi lehetővé.
4. *Navigálás*: a webhely navigációs szerkezetének létrehozására, megjelenítésére, kinyomtatására és módosítására szolgál.
5. *Hivatkozások*: a webhely hivatkozásainak (belső és külső) állapotát jeleníti meg és grafikusan jelzi az ellenőrzött és a megszakadt hivatkozásokat.
6. *Feladatok*: a webhely elkészítéséhez vagy karbantartásához szükséges feladatok listája.

Megjeleníthető a képernyő jobb szélén a munkaablak sáv, ahol például új weblapot, vagy webhelyet hozhatunk létre, a vágólapot kezelhetjük stb.

Összefoglalva megállapítható, hogy a FrontPage a konkurens szerkesztőprogramokhoz képest sokkal több szerkesztőeszközt biztosít, ugyanakkor a böngésző-független kompatibilitást veszélyezteti mindez, már nemcsak a böngésző kliens, hanem a webszerver oldalán is. [14]

6.4. Macromedia Dreamweaver

Az **Adobe Dreamweaver**, vagy egyszerűen **Dreamweaver** egy html szerkesztő és webfejlesztő program, eredetileg a Macromedia gondozásában (de az **Adobe Systems** tulajdonában). Rendelkezik WYSIWYG szerkesztővel, így biztosítja egyszerre a kódolási környezetet és a vizuális felületen megvalósuló oldaltervezést.

Támogatja az FTP-n keresztül történő feltöltést. Számos bővítése (extension) létezik, amellyel még egyszerűbbé, kényelmesebbé tehető a használata, valamint kódrészletekkel

tudja segíteni a szerkesztő munkáját. A vele készített forráskód tiszta, átlátható és mentes mindenféle fölösleges sallangoktól. Néhány figyelemre méltó tulajdonsága (AJAX framework, Browser kompatibilitási ellenőrzés, XML támogatás, LV támogatás stb.) miatt egyre többen használják. Véleményem szerint a Macromedia Dreamweaver a weblapszerkesztő szoftverek fejlesztésének csúcsán áll.

A program ismeri az újabb HTML szabványokat is, így használhatjuk a modern vezérlőelemeket is. Könnyedén elhelyezhetünk a dokumentumunkban ActiveX objektumokat, vagy akár Java applet-eket is. Támogatja az ImageMap funkciót, tehát egy kép bizonyos részeihez könnyedén rendelhetünk hozzá egy hiperhivatkozást. Bonyolultabb menüszerkezeteket lehet vele készíteni pillanatok alatt és a frame-ek kezelése is egyszerű. Egy hátrányát viszont megemlíteném: nincs magyar verziója, bár ez egy weblapfejlesztő számára szakmai angol nyelvismerete birtokában nem akadály.

A Dreamweaver támogatja a HTML 4.0 szabványt is. Ennek megfelelően a layerek használatával többretegű oldalt készíthetünk, sőt akár réteganimációkat is. A layerek használata a képek és a szövegek elhelyezését is nagymértékben megkönnyíti, ugyanis a layer az oldalon bárhova (pl. akár egy tized milliméterrel feljebb vagy lejjebb is) elhelyezhető. Más weblapszerkesztő programok nem minden esetben teszik lehetővé a képek pontos beállítását. A layerek egymásra is húzhatóak, le is takarhatják egymást. Újdonságot jelent az is, hogy leellenőrizhetjük, hogy az általunk használt HTML elemeket ismerik-e a korábbi 2.0/3.0-s böngészőprogramok. Megtehetjük, hogy oldalainkat átkonvertáljuk 3.0-s böngészőkkel kompatibilis formátumba, de ekkor elveszíthetjük a layerek adta előnyöket. A Dreamweaver segítségével még az oldalszövegben elhelyezett hiperhivatkozások helyes működése is ellenőrizhető.

Más WYSIWYG szerkesztőknél gyakran előfordult, hogy az oldalszövegben elhelyezett scripteket, mint például egy számláló-szkript történő hivatkozás, egész egyszerűen nem voltak hajlandóak elmenteni. Ez azzal járt, hogy a szkript beszúrása után már csak az oldal forrását volt célszerű módosítani, sok bosszúságot okozva a készítőnek. A Dreamweaver ebben is nagyon barátságos, egyáltalán nem nyúl az ilyen típusú hivatkozásokhoz. Nagy segítséget nyújt a program abban is hogy az egyes objektumokhoz viselkedési módokat társítsunk. Nincs szükség csak pár egérgattintásra, és máris telerakhatjuk az oldalt dinamikusan változó gombokkal. Az eseménykezelő metódusok a böngészők különböző

verziói szerint vannak csoportosítva, tehát nyugodtan tervezhetünk csak 3-as, vagy csak 4-es verzióra.

Munka közben egy ablakban megtekinthető az oldal forrása is. A forráson végzett változtatások azonnal érvényesülnek a dokumentumban is. Összességében tehát egy könnyen kezelhető, sokoldalú HTML szerkesztővel van dolgunk, amely a weblapok felépítését, kezelését, karbantartását nagyban megkönnyíti. [14]

7. WEBOLDALAK SZABVÁNYOSSÁGA, VALIDÁLÁS

A weboldalak felépítését a jelölőnyelvi kód biztosítja. A weboldal jelölőnyelvi kódja akkor szabványos (érvényes), ha az megfelel valamelyik érvényben lévő jelölőnyelvi szabványnak (ajánlásnak). A szabványosság a weboldalak hozzáférhetőségének egyik alapfeltétele. A jelölőnyelvi kód érvényességének vizsgálata (validálása) során az ellenőrizzük, hogy az adott weboldalak jelölőnyelvi kódja érvényes-e, azaz megfelel-e valamelyik érvényben lévő jelölőnyelvi szabványnak. A DOCTYPE-pal deklarálható, hogy a kód melyik jelölőnyelvi szabványt követve készült. A szabványok szerint a DOCTYPE deklarációnak szerepelnie kell a weboldal kódjának legelején. Ennek hiányában a weboldal kódja nem tekinthető érvényesnek (szabványosnak). A HTML vagy XHTML érvényességi vizsgálatot legegyszerűbben a W3C által biztosított – a 4.4.4. részben már említett – automatikus érvényesség-ellenőrző programmal végezhetjük. Validálást végezhetünk egyéb eszközökkel is, pl. az MSXML Parseren keresztül, az Opera validátorával stb. Megjegyzendő, hogy a validálás jól használható a kódolásban előforduló elírási hibák kiszűrésére is.

Az SGML/XML dokumentumot akkor nevezzük érvényesnek, ha maradéktalanul megfelel a hozzá tartozó DTD-nek. XML esetében a DTD elhagyható, illetve helyettesítheti az XML Schema.

A mai HTML böngészők a szintaktikai hibákat átlépik, illetve megpróbálják „kitalálni” a dokumentum létrehozójának szándékát, amely nem mindig hozza a kívánt eredményt. Nehéz lenne eldönteni, hogy vajon mi volt előbb: a szabványokat be nem tartó felhasználók, vagy a nem szabványos dokumentumokat is megjelenítő böngészők? Az eredmény szempontjából mindegy, a kódok sokszor átláthatatlanok vagy feldolgozhatatlanok a rosszul strukturált elemek miatt.

Az előzőekkel szemben az XML alapú böngészők ha nem jólformázott dokumentummal találkoznak, nem tudják feldolgozni azt. Ez persze kellemetlen a kódszerkesztőknek, de a bosszúságok mellett előnye is származik belőle. Kevésbé robosztus, ezáltal gyorsabb böngészőprogramokkal tudunk dolgozni.

A webszabványok előírják, hogy a tartalmat a jelölőnyelvi kódban, a megjelenését a CSS kódban kell megadni. A jelölőnyelvi kód elsődleges feladata a tartalom és a szemantikai struktúra biztosítása. Az így megírt oldalak a robotok által indexelhetők, áttekinthető tartalmúak és a böngészők többségével megjeleníthetők lesznek. A modern webfejlesztési

gyakorlat szerint kifogásolható, ha a jelölőnyelvi kódban megjelenésre vonatkozó elemek, attribútumok szerepelne. A tartalom és a megjelenés szétválasztása alapvetően hozzájárul az oldal akadálymentesítéséhez is. A CSS kód érvényességét egy – szintén a W3C által készített – CSS érvényesség-ellenőrző programmal¹¹ vizsgálhatjuk.

A honlap tulajdonosok és készítőik a szabványos weboldalaikra a legtöbb esetben büszkék, és ezt matricákkal jelzik is.



7. ábra: W3C validator matricák

Az egyes honlapokat beindexelő szoftverek, a Keresőrobotok is saját HTML értelmező motorral rendelkeznek. Azért is fontos szabványos weboldalakat írni, mert ekkor nagyobb esély van arra, hogy robotok értelmező motorja képes lesz feltárni a webhely tartalmát.

¹¹ <http://jigsaw.w3.org/css-validator/> - CSS Validation Service - Check Cascading Style Sheets (CSS) and (X)HTML documents with style sheets

8. BROWSEREK

A browser, magyarul böngésző olyan program, amely lehetővé teszi a webes csatlakozást és képes a webszervereken vagy saját gépen tárolt HTML vagy XHTML dokumentumok megjelenítésére, valamint azok rendezésére a felhasználó számítógépén. Ha a fájl nem teljesen felel meg egy szabványnak, vagy a fájl nem tudja megjeleníteni a szabványnak megfelelően a böngésző, akkor esetleg másképp jeleníti meg. Ismerünk szabványkövető és nem szabványkövető böngészőket. A mai böngészők többsége az első csoportba sorolható.

Egy weblap elérhető a webcímnek a böngésző címsorába történő begépelésével, vagy rákattintva egy másik weboldalnak egy hiperlinkjére, vagy a böngésző „*Kedvencek*” vagy „*Könyvjelzők*” funkciója segítségével.

A legelterjedtebb böngésző program az Internet Explorer, a Netscape Navigator (illetve utódainak tekinthető Mozilla és FireFox) és az Opera. Az Internet Explorer megvásárolható, a többi ingyenesen letölthető a fejlesztők honlapjáról. Ezeken túl számos más – többnyire ingyenes – böngésző létezik, beleértve a Linux alatt futókat is. A böngészők különböző beépülő elemekkel bővíthetők, melyek mindegyike a különleges szolgáltatások elérését biztosítja (multimédiás tartalmak és animációk lejátszása, vektoros képek mozgatása és nagyítása stb.).

Az **Internet Explorer**-t elsősorban Windows operációs rendszerekre fejlesztették ki, annak alapvető részét képezik, emiatt elterjedtségük ezzel is magyarázható. A látogatók túlnyomó többsége a 6.0-os esetleg a 7-es verziót használja, de kis százalékuk még mindig 5.5-ös és 5.0-ás verziókkal böngészik. Az említett négyfajta verziónak négy, némileg különböző motorja van, ami máshogy tér el a szabványoktól, más hibái vannak, emiatt olykor igen nehéz az összes verzióra való fejlesztés. A legújabb, nagymértékben átalakított kezelőfelületű és bővített funkciójú Internet Explorer 7 a régebbi, sokak által használt operációs rendszereken – mint pl. Windows 2000, Windows 98 – nem elérhető. Ráadásul egy Windows rendszerre egyszerre csak egy Internet Explorer telepíthető, ezért például a biztonsági frissítések módjára terjesztett IE 7-es böngészőt külön el kell távolítani, ha az egyelőre az elterjedtebb IE 6-tal szeretnénk megnézni oldalunk. [14]

Az internet hőskorának nagy klasszikus böngészői voltak a **Netscape** 4.x-es sorozat különböző verziói, mára már csak elenyésző számú felhasználó böngészik vele, akár csak az ezt követő Netscape verziókkal (7.02, 8.0). Az eredetileg a Netscape által fejlesztett böngészőmotor (gecko) nyílt forráskódúvá tételével jöttek létre a Mozilla, illetve a Firefox

böngészők (Netscape 5.x). Mind a régi Netscape, mind a különböző Mozilla verziók gyakorlatilag majdnem minden létező operációs rendszerre elérhetőek, a felhasználóknak mégis csak mintegy 10%-a használja, mivel nem alapértelmezett böngészője egyetlen operációs rendszernek sem. A gecko motort használó böngészők közül jelenleg a Firefox a legnépszerűbb, az eredeti Mozilla projekt pedig Seamonkey néven él tovább.

A **Mozilla Firefox** technikai értelemben véve napjaink egyik legjobb böngészője, mely a többi böngészőhöz képest számos előnyt nyújt (eltekintve a jól működő szabad forráskódú projektekre általában jellemző technológiai innovációtól és a biztonságosabb, stabilabb végterméktől), mint például:

- Az oldalon belüli keresést a leghatékonyabban oldja meg, mivel nem felugró ablakokba kell beírni a keresett kifejezést, ráadásul képes szöveges űrlapokban való keresésre.
- Sok Firefox kiegészítő segítségével nemcsak az alapvető böngésző-funkciókat lehet bővíteni, hanem új funkciókkal is fel lehet ruházni a programot, mint például RSS olvasó, fájlcsereklő kliens. [14]

Az **Opera** böngésző egy norvég cég által hosszú évek óta fejlesztett zárt forráskódú, több operációs rendszeren futó program. Persze nem minden operációs rendszerre érhető el mindig a legújabb verzió, hasonlóképpen a letölthető nyelvi verziók köre sem azonos, ezért lehet, hogy magyar nyelven csak egy régebbi verziót jelent még meg, vagy egyáltalán nem elérhető magyar nyelven az adott operációs rendszerre. Előnyei között említhető a gyorsasága, számos hasznos kiegészítő funkciója és könnyű használhatósága. Felmérések szerint a látogatóknak alig több, mint egy százaléka használja ezt a böngészőt, a program pusztán fennmaradása nagyrészt innovatív funkcióinak és gyorsaságának köszönhető. A böngésző funkció mellett még mail-kliens is található a programban. Az Opera for Mobile pedig az okostelefonok, PDA-k és egyéb mobiltelefonok széles választékára fejlesztett böngésző.

A **Konqueror** a Unix/BSD/Linux operációs rendszerekre fejlesztett KDE grafikus környezet alapértelmezett böngészője. Az általa használt, KHTML-nek nevezett motor felhasználásával írták meg a Mac OS X alapértelmezett böngészőjét, a **Safari**-t. Gyakorlatilag elenyésző számú látogató használja ezeket a verziókat még úgyis, hogy a Safari 2007 óta Windowsra is elérhető.

A 6. sz. *mellékletben* egy a böngészők használati arányait bemutató 2005. évben készült nemzetközi statisztika olvasható, a 7. sz. *melléklet* pedig egy saját fejlesztésű weboldal böngészési statisztikáját mutatja be.

ÖSSZEGZÉS

Az internet a technika és az alkalmazások fejlődésének következtében egyre nagyobb szerephez jut mindennapi életünkben, és szakmai munkánkban is. A internet és a web mint technológia és mint információhalmaz az 1990-es megszületése óta egyre jobban teret hódít a mindennapokban, és egyre látványosabb megjelenést tesz lehetővé, egyre több funkcióval bővülnek lehetőségei. Az információk áramlása soha nem látott méreteket ölt. Hatására egyszerre megváltoznak a tájékoztatási rendszerek, a kereskedelem, a jogrendszer és a szociális kapcsolatok. Azok, akik alábecsülik a jelentőségét és nem használják ki a lehetőségeit, lemaradnak. A web segítségével bárki kiléphet a nyilvánosság elé. Kiválóan alkalmas bemutatkozásra, termékek, szolgáltatások reklámozására, vásárlásra (webáruházban), információk kérdőíves összegyűjtésére, banki szolgáltatások igénybevételére stb.

A szöveges tartalmak leírására alkalmas kezdetleges leíró jelölő nyelvek még a számítógépek megjelenése előtt kidolgozásra kerültek. A jelölő nyelv olyan mesterséges nyelv, melynek segítségével az információ tartalma és szerkezete jelölhető és leírható. A hatvanas évektől kezdve pedig a dokumentumok számítógépes formázott tárolására és megjelenítésére már használtak jelölő nyelveket. A webdokumentumok nyelvének első nemzetközi szabványa az SGML, az IBM által kifejlesztett GML nyelv alapján a nyolcvanas évek elejétől került kidolgozása. Az SGML és a később ennek alapján kidolgozott XML az általános jelölő nyelvek csoportjába tartoznak. Ezeknek csak a jelölésre használható elemei és a szintaktikájuk meghatározott, szemantikájuk viszont csak a konkrét alkalmazásaikban meghatározott.

A HTML hosszú ideig kiváló nyelv volt a weboldalak leírásához és az általános tartalmak kódolására ma is jól megfelel. Az utóbbi években azonban az elektronikus kereskedelem mind nagyobb térhódítása következtében jelentkező speciális igények kielégítésére a HTML nyelv már nem alkalmas. Hogy miért is? Mert elavult a technikája és biztonságtechnikai szempontból nagyon sebezhető. Arról nincs szó, hogy a HTML lapok nem fejlődtek, hiszen folyamatosan újabb módszereket alkalmaztak a tökéletesítésére. Ez a folyamat újabb nyelveket is eredményezett, melyek alapelemeikben és módszereikben nem jelentősen térnek el, viszont célirányultságukban jelentős különbséget mutatnak.

Az XML megjelenésével újabb korszak jött el. Az XML nyelvet nagyobb mennyiségű adat elektronikus továbbítására fejlesztették ki. Ez egy olyan leíró nyelv, melyben a jelölésre

használt címkék szabadon definiálhatók és szemantikai értelmet adnak a dokumentumban foglaltaknak. Itt magunk definiálhatunk szinte mindent. A XML jelentős változást jelent, mivel túllép a jelölő nyelvek korábbi problémáin: az SGML túlzott komplexitásán és a HTML megjelenítés orientáltságán.

Ezek után a HTML nyelvet „összegyúrták” az XML nyelvvel, ebből alakult ki az XHTML, amely bővíthető hyperszöveg leíró nyelv. Ezt a fájlt könnyebb feldolgozni, és a böngésző program csak akkor jeleníti meg lapunkat, ha az a fájl hibátlan, azaz megfelel a nyelvet leíró szabványnak. Ellenkező esetben hibaüzenet jelenik meg képernyőnkön. Ebből következik, hogy a szerkesztőnek nagyon oda kell figyelni a szintaxisra és a lap felépítésére. Minden esetben érdemes validálni.

Az interaktivitás érdekében a nyelvi eszközök fejlesztése a dinamikus oldalakra fektette a hangsúlyt. További jelölő nyelveket alkalmaztak a webprogramozók, mint a DHTML-t és benne a Javascriptet. Mellettük egyre többen kezdték el alkalmazni a HTML-hez kidolgozott stíluslapokat a megjelenés leírására és a lapok designjának emelésére. Fontos szempontot kellett itt figyelembe venni: a látogató elkápráztatása. A dinamikus és funkcionálisan teljes weblap manapság egy sikeres vállalkozás alapja. Marketing szempontból is figyelembe kell vennünk ezeket a nyelveket, mivel az a célokat is teljesítő honlap, amely könnyen elérhető és sok látogatója van. E szempontokat fontos egy webszerkesztőnek figyelembe vennie. Az aprólékos tervezés, a finomítások stb. mind-mind kiemelt feladat, de legfontosabb tényező a folyamatos frissítés lehetőségének biztosítása.

Az oldal tervezése során elsőként a célokat kell megfogalmazni és hozzárendelni a közlendő tartalmakat és a funkciókat, amelyeket biztosítani szeretnénk a látogató számára. A tartalom rendezése után alakítható ki az oldalak rendszere és a navigációs struktúra. Meg kell tervezni a formai elemeket, és az interaktivitást biztosító funkciókat is. Figyelemmel kell lenni az akadálymentesítési követelményekre. A kódolás során a szabványosságra kell törekedni a böngészőfüggetlen megjelenítés érdekében.

A mai szerkesztőprogramok sok támogatást adnak a készítő számára. Nem kell már ezernyi kódot megjegyezni, hogy egy oldalt létrehozzunk. A mai tervezőprogramok az „alkotás” lehetőségét nyújtják számunkra egyszerű módszerekkel. Persze ez nem jelenti azt, hogy nem kell tudnunk a generált kód jelentését. Kiemelt szempont, hogy ismerjük a nyelvek leírását, szintaxisát. A szerkesztőprogram csak segít a munkában, de az sem tudhat mindent. Fontos, hogy tisztában legyünk a tagekkel, szabályokkal, különben a megjelenítő

programunk, a browser nem nyújt majd olyan élményt a látogató számára, mint amit szerettünk volna. Egy rosszul felépített, nem a célnak megfelelő honlap kevesebb hasznot hoz, alkalmanként elriasztó lehet, sőt, többet árt, mint használ. Az a jó honlap, amely képes kielégíteni a megcélzott látogatói kör tartalmi, formai és kezelési igényeit, és képes megjeleníteni bármely browser, vagy legalábbis azok, amelyeket a legtöbb látogató használ.

FOGALOMTÁR

formok	A HTML formjai speciális jelöléskészleten alapulnak, amelyek képessé teszik a böngészőket arra, hogy adatot gyűjtsenek a felhasználóktól, majd továbbítsák azokat a webszerverekre feltelepített input-feldolgozó programokhoz. A felhasználók a formokon keresztül lépnek kapcsolatba a webszerverekkel.
hipertext	Szövegek, képek és egyéb számítógépes adatok rendszerezésének olyan módszere, amelyben az egyedi adat egy másik adathoz vezet. Az adatok rendszerezésének egyik nonlinearis rendszere.
hipertext link	A HTML rendszerében egy speciális dokumentumelem, amelynek kiválasztásával a felhasználó egy adott dokumentumról egy másikra kerül.
horgony	Egy kóddal ellátott szöveges vagy képi elem, amely belső vagy külső linkként működik egy adott HTML dokumentumban, vagy egy másik dokumentumból induló horgony végpontja.
Java	A Sun cég által kifejlesztett gép- és operációs rendszer független programozási nyelv. Virtuális számítógépre (Java Virtual Machine - JVM) tervezték, ezért a Java programok módosítás nélkül futtathatók bármely gépen, amely képes a JVM szimulálására. Az Interneten a Web-alapú szolgáltatásoknál (pl. Web-áruházak) van fontos szerepe.
keresőgép	Speciális webprogram, amely az elérhető weboldalak adatbázisának tartalmát kutatja a felhasználó által megadott témák, kulcsszavak alapján.
kliens program	Az ügyfél - kiszolgáló (kliens/szerver) üzemmódban működő hálózati szolgáltatásoknál a felhasználó saját gépén futó szoftver, amely a felhasználóval való kapcsolattartás és a letöltött információk megjelenítése feladatát átveszi a távoli gépen működő szerver

programtól. A kapcsolat felépítését mindig a kliens kezdeményezi, és általában minden további tranzakciót is.

Script	script nyelven írott utasítássor, program.
script nyelv	Közvetlenül a forráskódú változatban, futásidőben értelmezett és gépi kódú utasításokká fordított nyelv.
tag	Címke, jellemző, HTML nyelven írt szövegek alapköveinek elnevezése.
webhosing	Weblapok üzemszerű tárolását jelentő szolgáltatás ügyfelek részére.
webmail	Webes felületen folytatott levelezés.
Webmaster	Hálózati, internetre kötött gépet (HTTP szerver vagy Website-ot) menedzselő személy, aki a szolgáltatás tartalmi és/vagy technikai oldaláért felelős.
webpage	Weboldal, honlap.
webring	Az azonos témakörű Web-területek láncba fűzött címlapjai alkotják.
website	Sok belső oldallal rendelkező, komplex tartalmat nyújtó hálózati hely, mely sok webpage-et tartalmaz.

RÖVIDÍTÉSJEGYZÉK

CGI	<i>Common Gateway Interface</i> , a webes böngészők és szerverek egymás közötti kommunikációjának és szolgáltatásainak folyamatát szabályozó leírás. Meghatározza a HTML-formok és egyéb dokumentum alapú kérdőívek adatainak továbbítási formátumát és szintaxisát.
DTD	<i>Document Type Definition</i> , magyar jelentése: dokumentum típus rögzítése. Egy elemdeklarációkból és jellemződeklarációkból felépített szerkezet, amely egy dokumentum értelmezési szabályait tartalmazza. A dokumentum szabványos SGML leírása, amely rögzíti azokat a szerkezeti elemeket és megjelenítési jelöléseket, amiket a dokumentumok szerkesztésekor kell felhasználni.
FTP	<i>File Transfer Protocol</i> , internetes fájlátviteli szolgáltatás, amely a TCP/IP protokollokon alapul. Segítségével lehetséges fájlokat másolni a hálózaton lévő más FTP szerverekre illetve szerverekről.
GUI	<i>Graphical User Interface</i> , magyar jelentése: grafikus felhasználói felület. Ez teszi lehetővé a szimbólumok alapján történő webes böngészést. A felhasználó egy vizuális jelekre alapozott felületet használ.
HTTP	<i>HyperText Transfer Protocol</i> , magyar jelentése: hiperszöveg átviteli protokoll. Internet-protokoll, amely a web felhasználói és a szerverek közötti kapcsolat fenntartásáért felelős.
HTTPS	<i>HyperText Transfer Protocol Secure</i> , magyar fordítása: biztonságos hiperszöveg átviteli protokoll. A HTTP protokollnak a titkosított, biztonságos hitelesítést lehetővé tevő változata. Gyakran találkozhatunk webáruházak, banki rendszerek, illetve internetes levelezőszerverek esetén, vagyis ahol fontos, hogy az adatforgalmat illetéktelenek ne csapolhassanak meg.

IP	<i>Internet Protocol</i> : speciális hálózati protokoll (adatátviteli szabályrendszer), amelyet az interneten lévő számítógépek összekötésénél használunk.
ISP	<i>Internet Service Provider</i> : olyan szervezetet jelöl, amely szolgáltatásként a számítógép használóknak kínál kapcsolódási lehetőséget az internetre díjfizetés ellenében. Magyarul internet-szolgáltatónak nevezzük.
PDA	<i>Personal Digital Assistant</i> : személyi digitális asszisztenst jelent, amely egy kézi számítógépnek az elnevezése.
SVG	<i>Scalable Vector Graphics</i> : méretezhető vektorgrafika, amely az XML leíró nyelvcsalád tagja és tiszta szöveges formátumot eredményez.
TEI	<i>Text Encoding Initiative</i> : szöveggódoló kezdeményezés, nemzetközi együttműködés a számítógépen tárolt szövegek formátumának szabványosítására (az SGML és XML alkalmazásával).
URI	<i>Uniform Resource Identifier</i> , magyarul egységes erőforrás azonosító. Karakter sorozat, amelyet egy webes erőforrás azonosítására használunk. Az erőforrást kétféleképp azonosíthatja: hely szerint (URL) vagy név szerint (URN), tehát az URI = URL + URN
URL	<i>Uniform Resource Locator</i> , azaz egységes erőforrás helymeghatározó. Ennek segítségével hivatkozhatunk az interneten található különböző weboldalakra, e-mail címekre, letölthető fájlokra stb. Tartalmazza az eléréshez szükséges kommunikációs protokoll nevét (pl. HTTP, HTTPS, FTP, MAILTO stb.), a számítógép, vagy tartomány nevét (vagy IP címét), egy hálózati port számot, amelyen az adott szolgáltatás elérhető, és azt az elérési utat, amelyen az erőforrás megtalálható. Felépítése tehát: protokoll://tartománynév:portszám/elérési_útvonal A protokoll elhagyása esetén a http szerint keres.

URN	<i>Uniform Resource Name</i> , magyarul egységes erőforrás név. Az absztrakt erőforrásoknak (könyvek, folyóiratok, dokumentációk stb.) az azonosítója. Az URN – ellentétben az URL-lel – állandó. Az URN-ek kiadása szabályozott módon, meghatározott intézményeken keresztül történik. Egy könyvnek az ISBN adata alapján rögzített azonosítója a következő lehet: urn:isbn:0-520-02356-0 .
XFRML	<i>XML Financial and Reporting Markup Language</i> : XML-en alapuló pénzügyi és beszámoló jelölő nyelv
WAP	<i>Wireless Application Protocol</i> : vezeték nélküli alkalmazási (adatátviteli) protokoll. Nyílt nemzetközi szabvány, amelyet hordozható eszközökön (mobiltelefonok, PDA-k) történő webböngészés lehetővé tételére fejlesztettek ki a.
WML	<i>Wireless Markup Language</i> : vezeték nélküli jelölő nyelv, XML alapú leírónyelv, a WAP specifikációt támogató eszközökhöz tervezték.
WYSIWYG	angol mozaikszó, mely a <i>What You See Is What You Get</i> - magyarul „Amit látsz, azt kapod” - szavakból származik. Magyar megfelelője az ALAKHŰ , az <i>azt látod (a képernyőn)</i> , <i>amit kapsz, hűen (a nyomtatón)</i> szavakból. A végeredmény már látható a dokumentum létrehozása közben. A weblapszerkesztőket használva a felhasználó vizuálisan elkészíti a weblapot, a program pedig „megírja” hozzá a látványt előállító kódot.

FELHASZNÁLT IRODALOM

- [1] *Bíró Szabolcs*: Szövegfeldolgozás XML alapokon. Neumann Kht., 2005
<http://www.hik.hu/tankonyvtar/site/books/b10104/word.doc#d0e585>
A letöltés időpontja 2008. október 13.
- [2] *Boda István*: Weblap-szerkesztés és leíró jelölő nyelvek
<http://www.inf.unideb.hu/~bodai/pascal2/weblapszerkesztes.html> 2008. augusztus 27.
A letöltés időpontja 2008. szeptember 5.
- [3] *Bradley, Neil*: Az XML-kézikönyv.
SZAK Kiadó Kft. 2005.
- [4] *Debolt, Virginia*: HTML és CSS – Webszerkesztés stílusosan.
Kiskapu Kiadó, 2005.
- [5] *Drótos László*: Hálózati értelmező szótár, NIIF, Budapest, 1999.
http://www.iif.hu/dokumentumok/niif_fuzetek/szotar/
A letöltés időpontja 2008. szeptember 6.
- [6] *Kupás Péter*: Weblapszerkesztés. Oktatási segédlet.
Károly Róbert Főiskola, 2003.
- [7] *Livingston, Dan*: CSS & DHTML webfejlesztőknek.
Kossuth Kiadó, Budapest, 2003.
- [8] *Salgáné Medveczki Marianna*: Az XML: új perspektívák könyvtár-informatikában.
Tudományos és Műszaki Tájékoztatás. 51. évfolyam (2004) 2. szám
- [9] *Sikos László*: XHTML – A HTML megújulása XML alapokon.
BBS-INFO Kiadó, Budapest, 2004.

- [10] *Spányik Balázs*: Weboldalkészítő sulí. PCWORLD, 2008. március - szeptember
<http://www.pcworld.hu/story.php?sid=10793> A letöltés időpontja 2008. október 17.
- [11] *Szabó Zénó László*: Tipográfia a weben. Egyetemi jegyzet. ELTE 2002
<http://edutech.elte.hu/jegyzet/tipografia/> A letöltés időpontja 2008. október 8.
- [12] *Tittel, E. – Pitts, N. – Valentine C.*: Dummies könyvek - HTML 4.
Kossuth Kiadó, Budapest, 2001.
- [13] Ajánlások az Elérhető Médiaért. *Látó-tér Alapítvány, Vakok Iskoláinak Módszertani Központja, W3C magyarországi irodája*
<http://www.paramedia.hu/ajanlasaink.html> A letöltés időpontja 2008. november 3.
- [14] *Felhasználói kézikönyvek, kezelési leírások*: Microsoft Office FrontPage 2003, Mozilla Firefox, Opera, Internet Explorer 7.0, Macromedia Dreamweaver MX 2004

ISO 8879:1986 szabvány szerinti Latin 2 entity-k

```
<!-- (C) International Organization for Standardization 1986
      Permission to copy in any form is granted for use with
      conforming SGML systems and applications as defined in
      ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
      <!ENTITY % ISolat2 PUBLIC
          "ISO 8879:1986//ENTITIES Added Latin 2//EN">
      %ISolat2;
-->
<!ENTITY abreve SDATA "[abreve]"--=small a, breve-->
<!ENTITY Abreve SDATA "[Abreve]"--=capital A, breve-->
<!ENTITY amacr SDATA "[amacr]"--=small a, macron-->
<!ENTITY Amacr SDATA "[Amacr]"--=capital A, macron-->
<!ENTITY aogon SDATA "[aogon]"--=small a, ogonek-->
<!ENTITY Aogon SDATA "[Aogon]"--=capital A, ogonek-->
<!ENTITY cacute SDATA "[cacute]"--=small c, acute accent-->
<!ENTITY Cacute SDATA "[Cacute]"--=capital C, acute accent-->
<!ENTITY ccaron SDATA "[ccaron]"--=small c, caron-->
<!ENTITY Ccaron SDATA "[Ccaron]"--=capital C, caron-->
<!ENTITY ccirc SDATA "[ccirc]"--=small c, circumflex accent-->
<!ENTITY Ccirc SDATA "[Ccirc]"--=capital C, circumflex accent-->
<!ENTITY cdot SDATA "[cdot]"--=small c, dot above-->
<!ENTITY Cdot SDATA "[Cdot]"--=capital C, dot above-->
<!ENTITY dcaron SDATA "[dcaron]"--=small d, caron-->
<!ENTITY Dcaron SDATA "[Dcaron]"--=capital D, caron-->
<!ENTITY dstrok SDATA "[dstrok]"--=small d, stroke-->
<!ENTITY Dstrok SDATA "[Dstrok]"--=capital D, stroke-->
<!ENTITY ecaron SDATA "[ecaron]"--=small e, caron-->
<!ENTITY Ecaron SDATA "[Ecaron]"--=capital E, caron-->
<!ENTITY edot SDATA "[edot]"--=small e, dot above-->
<!ENTITY Edot SDATA "[Edot]"--=capital E, dot above-->
<!ENTITY emacr SDATA "[emacr]"--=small e, macron-->
<!ENTITY Emacr SDATA "[Emacr]"--=capital E, macron-->
<!ENTITY eogon SDATA "[eogon]"--=small e, ogonek-->
<!ENTITY Eogon SDATA "[Eogon]"--=capital E, ogonek-->
<!ENTITY gacute SDATA "[gacute]"--=small g, acute accent-->
<!ENTITY gbreve SDATA "[gbreve]"--=small g, breve-->
<!ENTITY Gbreve SDATA "[Gbreve]"--=capital G, breve-->
<!ENTITY Gcedil SDATA "[Gcedil]"--=capital G, cedilla-->
<!ENTITY gcirc SDATA "[gcirc]"--=small g, circumflex accent-->
<!ENTITY Gcirc SDATA "[Gcirc]"--=capital G, circumflex accent-->
<!ENTITY gdot SDATA "[gdot]"--=small g, dot above-->
<!ENTITY Gdot SDATA "[Gdot]"--=capital G, dot above-->
<!ENTITY hcirc SDATA "[hcirc]"--=small h, circumflex accent-->
<!ENTITY Hcirc SDATA "[Hcirc]"--=capital H, circumflex accent-->
<!ENTITY hstrok SDATA "[hstrok]"--=small h, stroke-->
<!ENTITY Hstrok SDATA "[Hstrok]"--=capital H, stroke-->
<!ENTITY Idot SDATA "[Idot]"--=capital I, dot above-->
<!ENTITY Imacr SDATA "[Imacr]"--=capital I, macron-->
<!ENTITY imacr SDATA "[imacr]"--=small i, macron-->
<!ENTITY ijlig SDATA "[ijlig]"--=small ij ligature-->
```

<!ENTITY IJlig SDATA "[IJlig]"---=capital IJ ligature-->
 <!ENTITY inodot SDATA "[inodot]"---=small i without dot-->
 <!ENTITY iogon SDATA "[iogon]"---=small i, ogonek-->
 <!ENTITY Iogon SDATA "[Iogon]"---=capital I, ogonek-->
 <!ENTITY itilde SDATA "[itilde]"---=small i, tilde-->
 <!ENTITY Itilde SDATA "[Itilde]"---=capital I, tilde-->
 <!ENTITY jcirc SDATA "[jcirc]"---=small j, circumflex accent-->
 <!ENTITY Jcirc SDATA "[Jcirc]"---=capital J, circumflex accent-->
 <!ENTITY kcedil SDATA "[kcedil]"---=small k, cedilla-->
 <!ENTITY Kcedil SDATA "[Kcedil]"---=capital K, cedilla-->
 <!ENTITY kgreen SDATA "[kgreen]"---=small k, Greenlandic-->
 <!ENTITY lacute SDATA "[lacute]"---=small l, acute accent-->
 <!ENTITY Lacute SDATA "[Lacute]"---=capital L, acute accent-->
 <!ENTITY lcaron SDATA "[lcaron]"---=small l, caron-->
 <!ENTITY Lcaron SDATA "[Lcaron]"---=capital L, caron-->
 <!ENTITY lcedil SDATA "[lcedil]"---=small l, cedilla-->
 <!ENTITY Lcedil SDATA "[Lcedil]"---=capital L, cedilla-->
 <!ENTITY lmidot SDATA "[lmidot]"---=small l, middle dot-->
 <!ENTITY Lmidot SDATA "[Lmidot]"---=capital L, middle dot-->
 <!ENTITY lstrok SDATA "[lstrok]"---=small l, stroke-->
 <!ENTITY Lstrok SDATA "[Lstrok]"---=capital L, stroke-->
 <!ENTITY nacute SDATA "[nacute]"---=small n, acute accent-->
 <!ENTITY Nacute SDATA "[Nacute]"---=capital N, acute accent-->
 <!ENTITY eng SDATA "[eng]"---=small eng, Lapp-->
 <!ENTITY ENG SDATA "[ENG]"---=capital ENG, Lapp-->
 <!ENTITY napos SDATA "[napos]"---=small n, apostrophe-->
 <!ENTITY ncaron SDATA "[ncaron]"---=small n, caron-->
 <!ENTITY Ncaron SDATA "[Ncaron]"---=capital N, caron-->
 <!ENTITY ncedil SDATA "[ncedil]"---=small n, cedilla-->
 <!ENTITY Ncedil SDATA "[Ncedil]"---=capital N, cedilla-->
 <!ENTITY odblac SDATA "[odblac]"---=small o, double acute accent-->
 <!ENTITY Odblac SDATA "[Odblac]"---=capital O, double acute accent-->
 <!ENTITY Omacr SDATA "[Omacr]"---=capital O, macron-->
 <!ENTITY omacr SDATA "[omacr]"---=small o, macron-->
 <!ENTITY oelig SDATA "[oelig]"---=small oe ligature-->
 <!ENTITY OElig SDATA "[OElig]"---=capital OE ligature-->
 <!ENTITY racute SDATA "[racute]"---=small r, acute accent-->
 <!ENTITY Racute SDATA "[Racute]"---=capital R, acute accent-->
 <!ENTITY rcaron SDATA "[rcaron]"---=small r, caron-->
 <!ENTITY Rcaron SDATA "[Rcaron]"---=capital R, caron-->
 <!ENTITY rcedil SDATA "[rcedil]"---=small r, cedilla-->
 <!ENTITY Rcedil SDATA "[Rcedil]"---=capital R, cedilla-->
 <!ENTITY sacute SDATA "[sacute]"---=small s, acute accent-->
 <!ENTITY Sacute SDATA "[Sacute]"---=capital S, acute accent-->
 <!ENTITY scaron SDATA "[scaron]"---=small s, caron-->
 <!ENTITY Scaron SDATA "[Scaron]"---=capital S, caron-->
 <!ENTITY scedil SDATA "[scedil]"---=small s, cedilla-->
 <!ENTITY Scedil SDATA "[Scedil]"---=capital S, cedilla-->
 <!ENTITY scirc SDATA "[scirc]"---=small s, circumflex accent-->
 <!ENTITY Scirc SDATA "[Scirc]"---=capital S, circumflex accent-->
 <!ENTITY tcaron SDATA "[tcaron]"---=small t, caron-->
 <!ENTITY Tcaron SDATA "[Tcaron]"---=capital T, caron-->
 <!ENTITY tcedil SDATA "[tcedil]"---=small t, cedilla-->
 <!ENTITY Tcedil SDATA "[Tcedil]"---=capital T, cedilla-->
 <!ENTITY tstrok SDATA "[tstrok]"---=small t, stroke-->
 <!ENTITY Tstrok SDATA "[Tstrok]"---=capital T, stroke-->
 <!ENTITY ubreve SDATA "[ubreve]"---=small u, breve-->
 <!ENTITY Ubreve SDATA "[Ubreve]"---=capital U, breve-->
 <!ENTITY udblac SDATA "[udblac]"---=small u, double acute accent-->

```

<!ENTITY Udblac SDATA "[Udblac]"--=capital U, double acute accent-->
<!ENTITY umacr SDATA "[umacr]"--=small u, macron-->
<!ENTITY Umacr SDATA "[Umacr]"--=capital U, macron-->
<!ENTITY uogon SDATA "[uogon]"--=small u, ogonek-->
<!ENTITY Uogon SDATA "[Uogon]"--=capital U, ogonek-->
<!ENTITY uring SDATA "[uring]"--=small u, ring-->
<!ENTITY Uring SDATA "[Uring]"--=capital U, ring-->
<!ENTITY utilde SDATA "[utilde]"--=small u, tilde-->
<!ENTITY Utilde SDATA "[Utilde]"--=capital U, tilde-->
<!ENTITY wcirc SDATA "[wcirc]"--=small w, circumflex accent-->
<!ENTITY Wcirc SDATA "[Wcirc]"--=capital W, circumflex accent-->
<!ENTITY ycirc SDATA "[ycirc]"--=small y, circumflex accent-->
<!ENTITY Ycirc SDATA "[Ycirc]"--=capital Y, circumflex accent-->
<!ENTITY Yuml SDATA "[Yuml]"--=capital Y, dieresis or umlaut mark-->
<!ENTITY zacute SDATA "[zacute]"--=small z, acute accent-->
<!ENTITY Zacute SDATA "[Zacute]"--=capital Z, acute accent-->
<!ENTITY zcaron SDATA "[zcaron]"--=small z, caron-->
<!ENTITY Zcaron SDATA "[Zcaron]"--=capital Z, caron-->
<!ENTITY zdot SDATA "[zdot]"--=small z, dot above-->
<!ENTITY Zdot SDATA "[Zdot]"--=capital Z, dot above-->

```

Kulcsszavak SGML/XML-ben

SGML	XML	Értelmezés
CDATA	CDATA	Az attribútum csak karakteres adat lehet. Ezt az adattípust az értelmező nem dolgozza fel, változatlan formában átengedi az ellenőrzésnél.
ENTITY	ENTITY	Az attribútumban egy entitásra vonatkozó hivatkozás található.
ENTITIES	ENTITIES	Több entitást is meg lehet adni referenciaként. Az egyedeket egy listában, egymástól térköz karakterekkel elválasztva kell megadni.
ID	ID	Az attribútum egyedi azonosító, mely a dokumentum egy meghatározott pontját adja meg.
IDREF	IDREF	Az attribútum referenciát tartalmaz egy ID-re, mely a DTD egy más pontján van deklarálva. (Az ID-ket a dokumentum tartalmának hiperhivatkozásokkal való jelölésére használhatjuk.)
IDREFS	IDREFS	Olyan, mint az IDREF, de itt az attribútum ID-k egy listáját tartalmazza.
NAME	NMTOKEN	Az attribútum értéke lehet bármilyen szó vagy lexikális elem – szám, betű, írásjelek (kötőjel, kettőspont, aláhúzásjel).
NAMES	NMTOKENS	Egymástól szóközzel elválasztott NMTOKEN értékek listája.
NMTOKEN	NMTOKEN	
NMTOKENS	NMTOKENS	
NUMBER	NMTOKEN	
NUMBERS	NMTOKENS	
NUTOKEN	NMTOKEN	
NUTOKENS	NMTOKENS	
NOTATION	NOTATION	Az attribútum értéke egy NOTATION adattípus, melynek deklarációja a DTD egy másik pontján helyezkedik el.
Felsorolás	Felsorolás	A lehetséges értékek egy listáját tartalmazza. A listát zárójelek közt kell elhelyezni, ahol az adatok függőleges vonallal vannak elválasztva egymástól. A konkrét érték mindig a felsorolás valamelyik eleme kell, hogy legyen.

Forrás: Bíró Szabolcs [1]

A HTML-ben minimalizálható attribútumok és az XHTML alakjuk

HTML	XHTML
checked	checked="checked"
compact	compact="compact"
declare	declare="declare"
defer	defer="defer"
disabled	disabled="disabled"
ismap	ismap="ismap"
multiple	multiple="multiple"
nohref	nohref="nohref"
noresize	noresize="noresize"
noshade	noshade="noshade"
nowrap	nowrap="nowrap"
readonly	readonly="readonly"
selected	selected="selected"

Különleges karakterek leírói és kódjai

Meg-jelenés	HTML leíró	Unicod (UTF-8)	Meg-jelenés	HTML leíró	Unicod (UTF-8)	Meg-jelenés	HTML leíró	Unicod (UTF-8)
	 	 	À	À	À	à	à	à
í	¡	¡	Á	Á	Á	á	á	á
¢	¢	¢	Â	Â	Â	â	â	â
£	£	£	Ã	Ã	Ã	ã	ã	ã
¤	¤	¤	Ä	Ä	Ä	ä	ä	ä
¥	¥	¥	Å	Å	Å	å	å	å
¦	¦	¦	Æ	Æ	Æ	æ	æ	æ
§	§	§	Ç	Ç	Ç	ç	ç	ç
¨	¨	¨	È	È	È	è	è	è
©	©	©	É	É	É	é	é	é
ª	ª	ª	Ê	Ê	Ê	ê	ê	ê
«	«	«	Ë	Ë	Ë	ë	ë	ë
¬	¬	¬	Ì	Ì	Ì	ì	ì	ì
	­	­	Í	Í	Í	í	í	í
®	®	®	Î	Î	Î	î	î	î
¯	¯	¯	Ï	Ï	Ï	ï	ï	ï
°	°	°	Ð	Ð	Ð	ð	ð	ð
±	±	±	Ñ	Ñ	Ñ	ñ	ñ	ñ
²	²	²	Ò	Ò	Ò	ò	ò	ò
³	³	³	Ó	Ó	Ó	ó	ó	ó
´	´	´	Ô	Ô	Ô	ô	ô	ô
µ	µ	µ	Õ	Õ	Õ	õ	õ	õ
¶	¶	¶	Ö	Ö	Ö	ö	ö	ö
·	·	·	×	×	×	÷	÷	÷
¸	¸	¸	Ø	Ø	Ø	ø	ø	ø
¹	¹	¹	Ù	Ù	Ù	ù	ù	ù
º	º	º	Ú	Ú	Ú	ú	ú	ú
»	»	»	Û	Û	Û	û	û	û
¼	¼	¼	Ü	Ü	Ü	ü	ü	ü
½	½	½	Ý	Ý	Ý	ý	ý	ý
¾	¾	¾	Þ	Þ	Þ	þ	þ	þ
¿	¿	¿	ß	ß	ß	ÿ	ÿ	ÿ

A JavaScript vezérlési szerkezetek

blokkosítás	
<pre>{ utasítás; ... utasítás; }</pre>	A kapcsos zárójelek közé írt utasítások egy utasításnak számítanak.
elágazás	
<pre>if (feltétel) utasítás; [else utasítás;]</pre>	Ha a feltétel igaz, akkor az első utasítás hajtódik végre. Ha nem, és adott az else ág, akkor az.
többszörös elágazás	
<pre>switch (kifejezés) { case címke: utasítás; break; case címke: utasítás; break; ... default: utasítás; }</pre>	Ez csak a JavaScript 1.2-től érvényes utasítás! Kiértékelődik kifejezés, majd sorban összehasonlítódik a címkékkel. Ha valamelyikkel megegyezik, akkor az ahhoz tartozó ág végrehajtódik. Ha egyik címkével sem egyezik, és adott a default címke, akkor az. Ha nem adunk meg break utasításokat, akkor a vezérlés átfolyik a következő ágba.
ciklus	
<pre>for ([kezdő kifejezés]; [ciklusfeltétel]; [léptető kifejezés]) utasítás;</pre>	Kiértékelődik a kezdő kifejezés, majd amíg a feltétel igaz, végrehajtódik az utasítás és a léptetés.
hátultesztelő ciklus	
<pre>do utasítás; while (feltétel);</pre>	Végrehajtódik utasítás mindaddig, amíg a feltétel hamis nem lesz.
előltesztelő ciklus	
<pre>while (feltétel) utasítás;</pre>	Amíg a feltétel igaz, végrehajtódik az utasítás.
címkézett utasítások	
<pre>címke: utasítás; break [címke]; continue [címke]</pre>	A címkézetlen break kilép a legbelső ciklusból. Címkével a címkézett utasítást fejezi be. A continue rátér a ciklus következő iterációjára. Címkézve az adott ciklust folytatja.
lépkedés a mezőkön	
<pre>for (változó in objektum) utasítás;</pre>	A változó sorra felveszi az objektum összes mezőjét.

érvényességi tartomány váltás	
with (objektum) utasítás;	Az utasítás minden hivatkozása először az objektum mezőire való hivatkozásként lesznek kezelve. Ha nincs megfelelő mező, akkor külső változóként.
megjegyzések	
// /* ... */	A // után a sor végéig megjegyzés következik. A /* és */ között többsoros megjegyzés is szerepelhet.

Forrás: Javascript referencia <http://www.mediacenter.hu/help/javascript/>

2005. évi havi bontású nemzetközi statisztika
a böngészők használati megoszlásáról

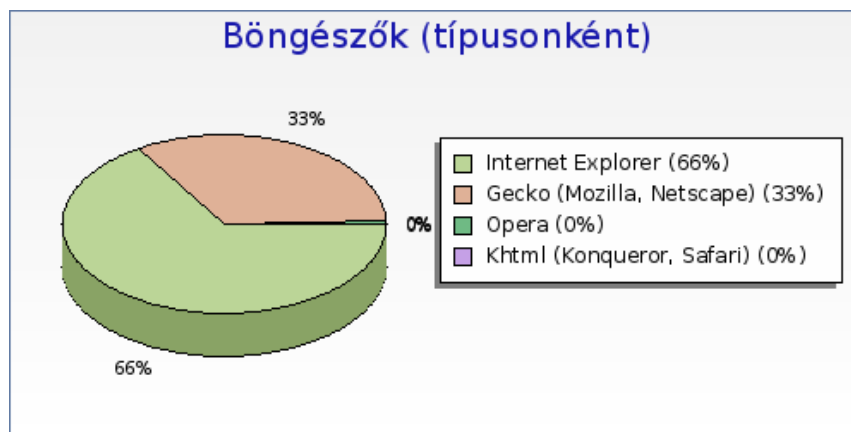
Rendering Engine	Nov 2005	Oct 2005	Sep 2005	Aug 2005	Jul 2005	Jun 2005	May 2005	Apr 2005	Mar 2005
MSHTML-Modern (Internet Explorer 6)	82.77% ▲	82.51% ▼	83.33% ▲	83.26% ▲	82.81% ▲	82.19% ▲	81.04% ▲	80.49% ▲	79.99%
MSHTML-Legacy (Internet Explorer 4/5/5.5)	3.76% ▼	3.94% ▼	4.34% ▼	4.49% ▼	4.88% ▼	6.51% ▼	7.77% ▼	8.76% ▼	9.23%
Gecko (Moz Suite, Firefox, Netscape 7, etc)	9.92% ▼	9.97% ▲	9.02% ▲	9.00% ▼	9.05% ▲	8.31% ▲	8.22% ▲	7.84% ▲	7.78%
KHTML (Safari/Konqueror)	1.63% ▲	1.62% ▲	1.30% ▲	1.23% ▲	1.19% ▲	1.01% =	1.01% ▲	0.96% ▲	0.95%
Opera (Opera - all versions)	0.83% =	0.83% ▲	0.72% =	0.72% ▼	0.73% ▲	0.68% ▲	0.67% ▲	0.65% ▼	0.66%
Netscape (Netscape 3/4)	0.07% =	0.07% ▼	0.08% ▲	0.07% ▼	0.08% ▼	0.10% ▲	0.09% =	0.09% =	0.09%
Other/Unknown (Unknowns, spiders, etc)	1.02% ▼	1.06% ▼	1.22% ▲	1.23% ▼	1.26% ▲	1.20% ▼	1.21% =	1.21% ▼	1.31%

Forrás: Browser Statistics By Rendering Engine

http://johnhaller.com/jh/useful_stuff/browser_statistics/

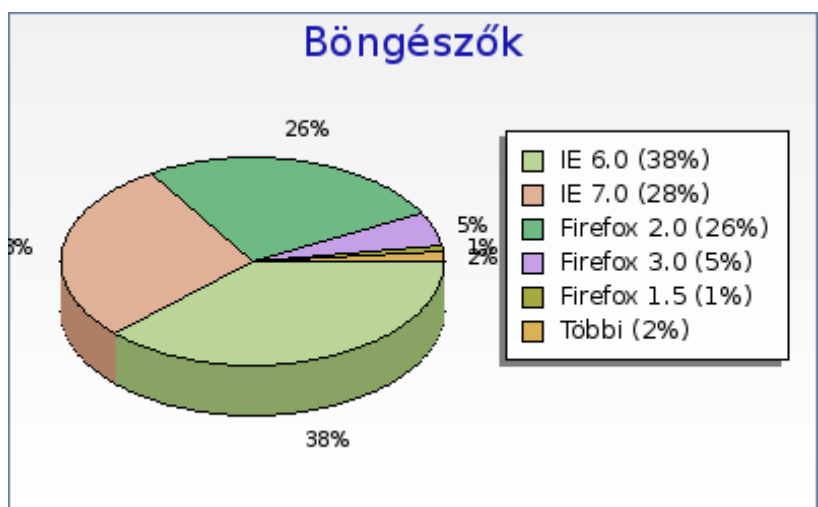
A sátoraljaújhelyi Kazinczy Ferenc Általános Iskola honlapjának<http://kazinczyfaltisk.extra.hu/>**látogatottságának statisztikái a 2008. január 1. és november 17. közötti időszakban**

Böngészők (típusonként)	Szám
Internet Explorer	1993 (66.3 %)
Gecko (Mozilla, Netscape)	996 (33.1 %)
Opera	15 (0.5 %)
Khtml (Konqueror, Safari)	2 (0.1 %)



Böngészők	Szám
Internet Explorer 6.0	1143 (38.0 %)
Internet Explorer 7.0	848 (28.2 %)
Firefox 2.0	789 (26.2 %)
Firefox 3.0	152 (5.1 %)
Firefox 1.5	26 (0.9 %)
Firefox 1.0	18 (0.6 %)
Opera 9.5	13 (0.4 %)
Mozilla 1.7	6 (0.2 %)
Firefox 0.1	3 (0.1 %)
Opera 9.6	2 (0.1 %)

Böngészők	Szám ▲▼	Oldalak látogatónként ▲▼	Oldalak fontos látogatónként ▲▼	1 oldal látogatási aránya ▲▼	Átlagos látogatási időtartam ▲▼
Internet Explorer 6.0	1143	4.8	6.2	26 %	00:06:24
Internet Explorer 7.0	848	4.1	6.1	39 %	00:04:40
Firefox 2.0	789	3.8	5.4	37 %	00:04:59
Firefox 3.0	152	4.0	5.6	34 %	00:03:04



Forrás: www.extra.hu