# Tartalomjegyzék

# 1. Bevezető

Szakdolgozatom témája a Weben használt leírónyelvek szabványosságának vizsgálata, az eközben feltárt hibák elemzése, megoldási lehetőségek bemutatása. A dolgozat megpróbál rámutatni olyan alapvető problémákra, mellyel nap, mint nap találkozhatunk, és amelyekre már régóta létezik megoldási lehetőség.

A XX. században már szinte minden használati tárgyunk tulajdonságait, alapvető felépítését kisebb-nagyobb mértékben valamilyen szabvány határoz meg, ajánlás ír le. Az adott terméket (legyen az műszaki cikk, vagy élelmiszer) ezeknek a szabványnak a betartásával kell elkészíteni. Természetesen a gyakorlat nem ezt mutatja. Számtalan esetben fordul elő, hogy a gyártók a szabványt nem tartják be, bizonyos dolgokban eltérnek tőle, esetlegesen módosításokkal alkalmazzák. Rosszabb esetben teljesen figyelmen kívül hagyják. Dolgozatomban megpróbálom feltárni a legelterjedtebb hibákat, majd arra megoldási lehetőséget ismertetek, mellyel az oldal számos előnyös tulajdonságra is szert tehet.

Ha jobban belegondolunk, ez több dolgot is eredményezhet: vagy olyan eredményt kapunk, ami a célközönség számára használhatatlan, vagy nem elégíti ki teljes mértékben azokat a funkciókat, amit a felhasználó elvárna, (jó példa erre az érintésvédelmi szabványok be nem tartása) esetlegesen saját testi épségét, életét is veszélyezteti. A szabványok tartalmukban illetve részleteikben országonként eltérhetnek, van azonban egy szervezet, aki világszinten próbálja ezeket összeegyeztetni, kiegészíteni, illetve újakat létrehozni. Ez a Nemzetközi Szabványügyi Bizottság (ISO).

Sok olyan ajánlás létezik azonban, amit bizonyos területre specializálódott szervezetek dolgoztak ki. Ezek közül egyik a World Wide Web Consortium [1]. Feladatuknak tekintik az Internet egyik mellékágával, a Web-bel szorosan összekapcsolódó technológiák összegyűjtését, valamint azok gondozását. Rengeteg ajánlást dolgoztak már ki, némelyik már a negyedik generációján is túl van (HTML 4.01 [1.1]). Ennek oka, hogy az évek során újabb és újabb igények merültek fel, ezeket új szabványok kiadásával elégítették ki. Ezekre, mint stabil pontokra támaszkodhatnak munkájuk során a webfejlesztők.

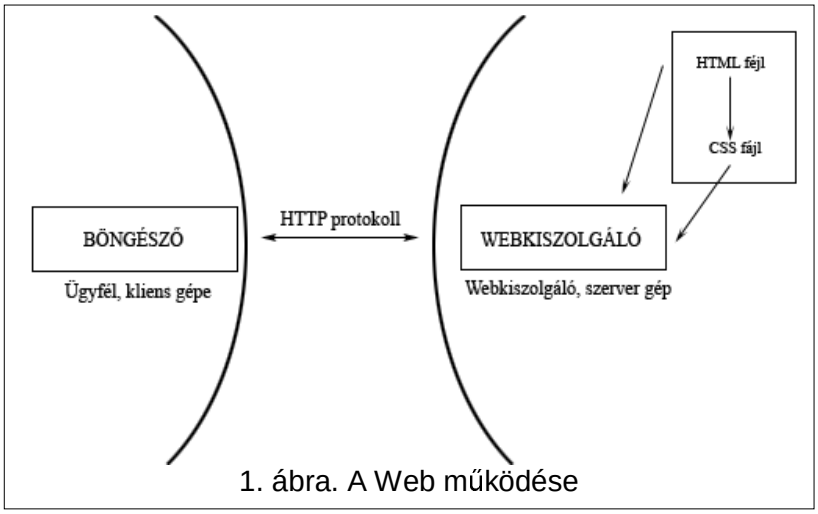
A szabványok be nem tartása egy weblap esetében a következőket vonhatja maga után: előfordulhat, hogy a célközönség egyáltalán nem tudja a dokumentumot hasznosítani. Jobb esetben csak bizonyos külső megjelenési problémák fordulhatnak elő, de az információ lényegében kinyerhető az oldalról.

Mivel azonban a HTML dokumentumokat nem ugyanazzal a programmal szerkesztjük és jelenítjük meg, (a megjelenítés böngészőprogrammal történik) ezért a felhasználó gépén futó megjelenítő alkalmazáson is sok múlik. Ugyanis hiába van egy oldal az ajánlásnak megfelelően elkészítve, ha a böngészőprogram magát a szabványt félreértelmezi, illetve bizonyos részeit nem ismeri. Ennek a problémának az okozói egyértelműen a fejlesztők, akik a szabványok helyett a kényelmet, a lehető legnagyobb hibatűrést próbálják előtérbe helyezni. Példa erre a legelterjedtebb operációs rendszer, melynek böngészője hírhedt arról, hogy a szabványokat nem tartja be. Jelentősen megnehezíti ezzel azoknak a webfejlesztőknek a munkáját, akik olyan oldalakat szeretnének létrehozni, amelyek mindenki számára közel azonos formában érhetőek el.

A kiadás után felmerülő hibákat többségében utólag korrigálni lehet javítócsomag, illetve új verzió kiadásával. A legtöbb böngészőprogram fejlesztő csapat próbálja kijavítani a felismert hibákat, és egyre jobban alkalmazkodnak a szabványokhoz. Szerencsére jó néhány olyan program áll rendelkezésre, melyek a szabványoknak való megfelelést helyezik előtérbe, ráadásul ezek szinte mind ingyenes programok, tehát bárki számára szabadon hozzáférhetők. (Letölthető: *http://firefox.hu*)

## 2. A World Wide Web

A világháló sok szolgáltatása közül a legismertebbek a World Wide Web, valamint az elektronikus levélküldési lehetőség, az e-mail. A www, közkeletűbb nevén a Web, lényegében egy nyílt rendszer, ami az átlagfelhasználó számára főleg képi illetve szöveges információ-hozzáférési lehetőséget jelent. Működése az ***1. ábrán*** tekinthető meg. Az Internet hálózatára kapcsolt úgynevezett Web szervereken, kiszolgálókon tárolják az információt tartalmazó weblapokat, melyek kérésre a HTTP protokoll segítségével küldik az adott weblapot.



1. ábra. A Web működése

Ezekhez a lapokhoz férhet hozzá a felhasználó, mikor a világhálót böngészi. A lapok száma mára már elérte a milliárdos nagyságrendet, folyamatosan kerülnek fel újak,

és esetenként tűnnek el a régiek. Az Internet a kommunikáció és az információ-megosztás egyéb formáinak széles tárházát kínálja – ilyen az elektronikus levelezés, hírcsoportok, stb.

Ma már nem igen tudnánk olyan témakört mondani, mellyel ne foglalkozna egy vagy több oldal, köszönhető ez annak, hogy az emberek általában szívesen megosztják másokkal a rendelkezésükre álló információkat. Manapság ez már gyerekjáték, bárki elhelyezheti dokumentumait, véleményét például egy ingyenes tárhelyet biztosító szerveren. A Web mindig is teret adott a tudományos jellegű adatok cseréjének, ám a kilencvenes évek eleje, közepe óta jelen vannak a kormányok, közhivatalok, és ami a legjellemzőbb, a kereskedelmi célú szervezetek is.

# 3. A HTML (HyperText Markup Language)

A www (World Wide Web) lényegében egy nyílt rendszer, ami az átlagfelhasználók számára főleg képi illetve szöveges információ-hozzáférési lehetőséget jelent. Az Internet hálózatára kapcsolt úgynevezett Web szervereken, kiszolgálókon tárolják az információt tartalmazó weblapokat. Ezekhez a lapokhoz férhet hozzá a felhasználó, mikor a világhálót böngészi. Tulajdonképpen a HTML a Web leírónyelve.

## *3.1. Egy kis történelem*

A Web nem volt mindig ennyire látványos és könnyen kezelhető. Azok az oldalak, melyekkel manapság találkozhatunk, csak a 90-es évek eleje óta léteznek, tehát azóta, hogy létrehozták a hypertext alapú protokollokat. Mire is jó a hypertext? Annak, aki már böngészett az Interneten, talán nem kell elmagyarázni, mekkora segítség, ha az oldalakat, dokumentumokat egy hivatkozáshálózat szövi át, melyek segítségével könnyen juthatunk el egyik oldalról a másikra, vagy adott dokumentumon belül egy másik pozícióra. A HTTP (HyperText Transfer Protocol) és a HTML jóvoltából a kiszolgálón tárolt valamennyi nyilvános adatot egy egyszerű program segítségével is elérhetjük. Az Interneten található oldalak megjelenítésére használt programok a „böngészők”. Természetesen, mint minden egyéb programcsaládból, ezekből is bőséges kínálat áll rendelkezésre, a legtöbb operációs rendszer azonban tartalmazza a saját Web-böngészőjét. Az Internet „ifjúkorában” még csak szöveges megjelenítésű programok álltak rendelkezése, melyet még ma is használnak néhányan, de kijelenthető, hogy a felhasználók döntő többsége grafikus felületű böngészőt használ. Minden böngészőprogram számára a forrásanyagot HTML nyelven kell megírni, ez a nyelv hordozza magában az összes olyan információt, ami a böngészőnek a dokumentum struktúrájának meghatározásához szükséges. Az oldalon megjelenő HTML

lap kódja létrehozható statikusan, illetve feltölthető a tartalom dinamikusan is, különböző scriptek, adatbázisok használatával.

A HTML nyelv őse a 80-as évek elején keletkezett és csakúgy, mint akkoriban a legtöbb szabvány, ez is az IBM cégtől indult el. A fejlesztők arra jöttek rá, hogy a legtöbb szövegnek rengeteg közös része van (címe, alcíme, főrésze, stb.), így ha az egyes részekre vonatkozó formázóutasításokat magába a szövegbe lehetne belefoglalni, akkor viszonylag könnyen lehetne hardver- és szoftver független, formázott szövegeket létrehozni. Az IBM ki is fejlesztett egy ilyen pszeudó programozási nyelvet, amelyet GML-nek (Generalized Markup Language) nevezett el. Ebből a nyelvből alakította ki a Nemzetközi Szabványügyi Hivatal (ISO) az SGML-t (Standard Generalized Markup Language, ISO 8879/1986). A két nyelv között nem volt jelentős különbség. Az SGML nyelv egyik alkalmazása a HTML nyelv, ezt a nyelvet Tim Berners-Lee tervezte meg 1990-ben. Érdekes megfogalmazásban ez tulajdonképpen egy leírónyelveket leíró nyelv. Az elnevezésben a „H” betű a „hyper” szóra utal, vagyis nemcsak felépítést meghatározó információk vannak a fájlban, hanem más oldalakra, fájlokra való hivatkozások is. Az alapötlet végül is az, hogy a szövegben apró jelöléseket (angolul *„tag”*-eket) helyezünk el a megjelenítendő rész elé, amelyek megmondják a futtató programnak, hogy milyen elrendezésben kell a dokumentumot felépítenie.

Ezeket az információkat kisebb és nagyobb relációjelek (<> pl.: <*body*>) közé kell írni, ezeket a program nem fogja megjeleníteni. A HTML nyelv szintaktikája, elemei szabványosítva vannak, ezt a szabványt a World Wide Web Consortium [1] gondozza.

Ezt a szabványt kellene a böngészőprogramoknak minél szigorúbban figyelembe venniük. Többségük ezt meg is próbálja, több-kevesebb sikerrel. A tapasztalat azt mutatja, hogy a böngészőprogram fejlesztők – véletlenül, esetleg szándékosan – nem veszik figyelembe a szabványok egyes részeit, ezért néhány böngészőprogram képes a HTML utasítások mindegyikét értelmezni (pl.: Microsoft Internet Explorer). Ilyenkor az ismeretlen részeket jó esetben figyelmen kívül hagyják, rosszabb esetben, pedig tévesen értelmezik, ezzel nem várt eredményeket okozhatnak. Ennek tudható be az a jelenség, hogy két különböző böngészőprogrammal ugyanazt az oldalt nézve, eltérő eredményt láthatunk. Természetesen a HTML nyelv sem kiforrottan jelent meg, többször javították, egészítették ki a szükségesnek tartott elemekkel.

## 3.2. A HTML különböző verziói

Napjainkban a HTML már a negyedik verzióban jár (egészen pontosan HTML 4.01 [1.1] ). Minden régebbi kiadás részhalmaza az újabb verziónak. Ez azt jelenti, hogy az új verzió csak kibővíti a régebbi változatot, de a korábbi lehetőségeken semmit nem változtat. Előfordulhat azonban, hogy az esetlegesen elavultnak nyilvánított részek használatát nem javasolják. Fontos továbbá megemlíteni, hogy az egyes HTML verziók, és a böngészőprogramok között milyen kapcsolat van. A weblapok tulajdonképpen egyszerű szövegfájlok, akár egy egyszerű szövegszerkesztővel is létrehozhatóak, ezek mindig egy Web-szerveren vannak elhelyezve, hogy bárki, akinek Internet hozzáférése van, elérhesse. Amikor egy ilyen lapot meg szeretnénk a böngészőprogramunkkal tekinteni, a program egy kérést küld a szerver felé, melyben megadja, hogy melyik lapra van szüksége, majd a szerver, ha módjában áll, ezt a lapot küldi válaszként. Ekkor a számítógépünk memóriájába töltődnek az oldal részegységei, a böngészőprogram a megfelelő jelzések értelmezésével az előre megtervezett formátumban fogja megjeleníteni azt. Ezért a böngészőnek ismernie kell az összes – a dokumentumban használt – „*tag*”-et, pontosabban annak szabványban meghatározott jelentését.

### 3.2.1. HTML 0 szint

A HTML 0 az összes böngésző „közös nevezője” – ezt a szintet minden böngészőprogram támogatja. Tulajdonképpen az eredeti SGML alapján épül fel, természetesen lehetővé téve más fájlokra való hivatkozásokat is.

### 3.2.2. HTML 1 szint

Minden olyan jelölést tartalmaz, amit a 0-s szint, kiegészült azonban azzal, hogy lehetővé teszi bizonyos szövegrészek megkülönböztetését, színének, kinézetének megváltoztatásával, valamint lehetővé válik a képek elhelyezése a dokumentumban. Zavaró hiányosságává vált később ennek a szintnek, hogy a szöveggel nem lehet körbefolyatni az ábrát, képet, tehát egy a kép mellet semmilyen szöveges információ nem lehetett. Ezt később a HTML 3-as szint korrigálta.

### 3.2.3. HTML 2 szint

Az előző szintekhez képest annyival bővült, hogy lehetővé teszi az olvasónak az adatbevitelt. Így a felhasználó írhat például szövegmezőbe, vagy választhat egy listából. A Web történetében ez a szint tette először lehetővé az adatok kétirányú áramlását, így már

nemcsak olvasásra szánt anyagok terjeszthetők vele, szert tett egy minimális interaktivitásra is. A szabvány lezárására 1996-ban került sor.

### 3.2.4. HTML 3 szint

Ez volt az első, igazán elterjedt szint, pontosabban a 3.2-es verziója. Tovább bővítve elődjeit, lehetővé vált táblázatok és vezérlőelemek, illetve *applet-ek, script-ek* használata. További érdekesség, hogy itt vált először lehetővé az alsó és felső index használata, ami nagyon fontos segítség a tudományos témájú publikációkhoz. Egy általános statikus HTML lap manapság is nagy részben ehhez a szinthez tartozó jelöléseket használ.

### 3.2.5. HTML 4 szint

A HTML 4.0 verzió 1997 végén (december 18.-án, ISO 8879) jelent meg, mint hivatalos ajánlás, majd némi módosítás után kiadták a végleges verziót (1998 április 24.-én). Ez a következő mechanizmusokkal egészíti ki elődjeit: stíluslapok, *frame-ek* (keretek) használata, továbbfejlesztett szöveg és táblázatirány meghatározás, űrlapok, fogyatékos felhasználók számára elérhetőség, objektumok beágyazása, továbbfejlesztett táblázatok.

Talán napjainkban még mindig az egyik leggyakrabban használt verzió a HTML 4.01. 1999. december 24.-én bocsátották útjára ezt az ajánlást, mint azt a verziószáma is mutatja, nem új szint, hanem elődjének, a 4.0-nak kibővítése.

## 4. XHTML (Extensible HyperText Markup Language)

Az XHTML [1.2] ajánlást 2000. január 26-án bocsátotta ki a W3C [1]. A család dokumentumtípusai XML alapúak, és arra lettek tervezve, hogy együttműködjenek az XML alapú felhasználói alkalmazásokkal. Az XML egy betűszó, jelentése: kiterjeszthető jelölő nyelv (eXtensible Markup Language). Bár szigorított formája az SGML-nek, megőrzi annak erejét, gazdaságosságát, megtartja az összes általánosan használt tulajdonságát. Tulajdonképpen kiterjeszti a HTML 4-et. Az XHTML család első tagja az 1.0-ás verzió volt. Ez a három HTML 4 dokumentumtípus megújítása, XML 1.0 alkalmazásként. Az ilyen típusú dokumentum egyaránt olvasható XML kompatibilis, és néhány egyszerű irányelv betartásával, HTML 4 kompatibilis böngészővel is. Aki e dokumentumtípus használata mellett dönt, az a következő előnyökkel számolhat: az XHTML dokumentumok megfelelnek az XML előírásainak. Könnyedén megtekinthetők, szerkeszthetők, és érvényesíthetők a standard XML eszközökkel. Egy XML dokumentum akkor érvényes (*valid*), ha a DTD fájl által megkövetelt szabályoknak is eleget tesz. Ahogy

az XHTML család fejlődik, az XHTML 1.0 kritériumainak megfelelő dokumentumok egyre jobban együtt tudnak működni különböző XHTML környezetekben.

Az XHTML dokumentumok ugyanolyan jól szerkeszthetőek korábbi, HTML 4-et támogató felhasználói alkalmazásokkal, mint az új, XHTML 1.0-t támogató programokkal. Az XHTML család a következő lépés az Internet fejlődésében. Ezen dokumentummal szemben támasztott követelések némiképp eltérnek a HTML 4-el szemben támasztottaktól: a dokumentumnak „jól formázott”-nak kell lennie. Ez egy új koncepció, amelyet az XML vezetett be. A jól formázottság tulajdonképpen azt jelenti, hogy minden elemnek záró résszel is kell rendelkeznie, vagy speciális formában kell leírni. Bár az úgynevezett átlapolás (hibás egymásba ágyazás) érvénytelen az SGML-ben is, a meglévő böngészők mégis széles körben tolerálják. Az XHTML dokumentumokban minden HTML elemet és attribútum nevet kötelezően kisbetűvel kell írni. Ez a különbség szükséges, mivel az XML érzékeny a kisbetű-nagybetű különbségekre.

Ajánlott továbbá az XHTML laphoz csatolt stíluslap teljes tartalmát is kisbetűvel írni, az esetleges kompatibilitási problémák elkerülése végett. Az SGML alapú HTML 4 néhány elemnél opcionálissá tette a záró elemek használatát: a következő elem kezdete zárta az előzőt. Az XML szintaktika, ezáltal az XHML sem engedi a záró elemek elhagyását. (pl.: eddig a HTML-ben megengedett volt ez a forma: *<li>ez egy listaelem*. Az XHTML-ben ez már szintaktikailag helytelen, a helyes megadás a következő: *<li>ez egy listaelem</li>*) Az elemek tulajdonságainak értékét mindig idézőjelek között kell feltüntetni, még akkor is, ha értékük nem sztring, hanem szám.

# 5. CSS (Cascading Style Sheets)

A HTML nyelvet tulajdonképpen csak leírónyelvnek szánták, bár később bevezettek néhány megjelenést befolyásoló elemet. Eleinte ezek nem is nyújtottak kielégítő megoldást. Az új formázási lehetőségek sok mindent lehetővé tettek az oldal készítője számára, de a nyelv ezzel elvesztette az egyszerűségét, a dokumentumok a formázó tag-ek miatt egyre bonyolultabbá, és összetettebbekké váltak.

Nem volt ritka, hogy a dokumentum tartalmának több mint a felét foglalták el ezek az elemek, jelentősen megnövelve ezzel a HTML fájl méretét, ezen keresztül pedig a letöltési időt. Ezen segített megjelenésével a CSS [1.3] szabvány, melyet a böngészők az utóbbi években egyre egységesebben értelmeznek, s emiatt sokkal rugalmasabbá, szabadabbá vált a HTML oldalak megjelenítése. A CSS egy mozaikszó, jelentése: egymásba ágyazott stíluslapok (Cascading Style Sheets).

Egy egyszerű szintaktikájú nyelvről van szó, mellyel a HTML oldalak megjelenése befolyásolható (elhelyezkedés, szín, méret, stílus, stb.). A CSS tehát egy stíluslap megvalósítás, amely lehetővé teszi, hogy a HTML oldalak szerzői oldalaikhoz egyedi stílust rendeljenek hozzá. A CSS egyik alapvető tulajdonsága a folyamatos stíluslap – HTML lap kapcsolat. A lapok szerzői az általuk alkalmazni kívánt stílust egyszer rögzítik, és hozzákapcsolják minden általuk készített HTML laphoz. A HTML dokumentum, pedig a megjelenítési információkat a hozzá rendelt stíluslapból fogja beszerezni. CSS használatával nemcsak megszépíthető egy oldal, de a készítő munkáját is jelentősen megkönnyítheti. Egyszerűbbé válik például egy későbbi design, arculatváltás. Maga a HTML fájl tehát szinte változatlan marad, csak a szövegre vonatkozó formázási információkat kell módosítani a CSS fájlban. A stílusokat elmenthetjük külső fájlba is (a lehető legnagyobb rugalmasság érdekében), de magába a HTML kódba is belefoglalható, illetve lehetőség van használatukra *inline* módon is, tehát az egyes tag-ek szintjén is. Az egymásba ágyazhatóság arra utal, hogy több stíluslapot, definíciót is megadhatunk egyszerre, illetve egy stílus lehet több elemre is érvényes (egymást felüldefiniálhatják).

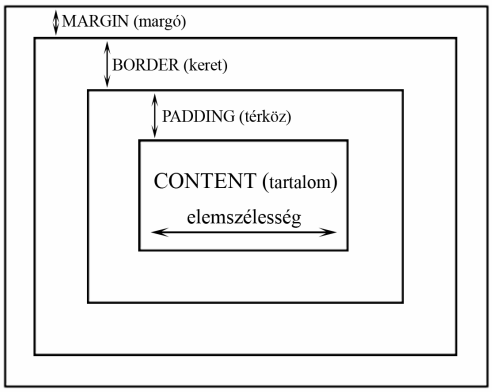
Az oldal hierarchiája szerint a stílusok öröklődnek. Ha legfelső szintű elemre definiálunk egy stílust, akkor az összes többi alsóbb szintű (felül nem definiált) elemre érvényes lesz. Persze a helyzet még mindig nem ideális, s ez egy jó ideig nem is lesz az, köszönhetően a domináns böngésző technológiai elmaradottságának. Maga a CSS1 [1.3] szabvány a W3C által 1996. december 17.-én látott napvilágot. Azután több módosításon is átesett, mígnem 1998. május 12.-én megjelent a CSS2 [1.4] szabvány is, mely hasonlóan a HTML-hez, teljes egészében magába foglalja az előző verziót.

Fontos megemlíteni, hogy a közkedvelt böngészők viszonylag nagy részét támogatják a CSS szabványoknak, de könnyen előfordulhat, hogy valamit félreértelmeznek. Továbbá az sem lehetetlen, hogy egy böngésző már sok mindent ismer a CSS2-ből, de még mindig van olyan CSS1 tulajdonság, amit nem ismer, nem tud értelmezni, megvalósítani. A CSS1 felhasználói alkalmazások képesek olvasni a CSS2 stíluslapokat és elvetik azokat a részeket, amelyeket nem képesek értelmezni. Előfordulhat, hogy a felhasználó olyan régi böngészőt használ, mely még nem támogatja stíluslapok használatát (pl. 3.x verziójú Internet Explorer). A tartalom ekkor is meg fog jelenni a képernyőn, ilyenkor az alapértelmezett formázásokat fogja alkalmazni.

A CSS2 ugyan sok mindennel bővült, de lényegében megtartotta elődje egyszerűségét. Néhány CSS lehetőség a fogyatékkal élő felhasználók számára jobban hozzáférhetővé teszi a Webet: nagy segítség lehet a látáskárosultak számára a hangos

stíluslap (*aural*), mellyel a felolvasó program hangja módosítható, hangerejét, hangszínét, emelkedettségét a dokumentum készítője szabályozhatja a tartalomtól függően (természetesen ehhez elengedhetetlen a felolvasóprogram CSS támogatása).

A betűk formázási lehetőségei sokkal nagyobb teret engednek a fantáziának, ezáltal elkerülhető a képpel megjelenített szövegek használata, mely mindig is nehézségeket okozott a fogyatékos felhasználóknak.




2. ábra. A CSS doboz modellje

A CSS-ben alapvető szerepet játszik a doboz modell. A CSS elsődleges célja ugyanis a vizuális megjelenítés szabályozása, s a doboz modell az elhelyezés alapvető eszköze. CSS-t használva minden elem egy dobozként fogható fel. A doboz modell felépítése ***a 2. ábrán*** látható. Itt még egyszer érdemes megemlíteni, hogy a böngészők (főleg a Microsoft Internet Explorer) nem teljesen egységesen értelmezik ezt a doboz modellt, melyből nem várt eredmények származhatnak.

Lehetőség van eltérő stíluslapok használatára különböző médiumok esetén, tehát más és más beállítások érvényesülhetnek a számítógép képernyőjén, valamint nyomtatásban, illetve speciális eszközök alkalmazásakor. A fontosabb médiatípusok [5] az **1. táblázatban** találhatóak:

A médiatípusok meghatározása 1. táblázat

| **Használt név:** | **Meghatározás:** |
|---|---|
| *screen* | színes számítógép megjelenítő |
| *aural* | beszédszintetizátorokhoz |
| *braille* | Braille érintő képernyőkhöz |
| *print* | lapozható médiához, ez érvényesül nyomtatásnál is |
| *handheld* | kisméretű, monochrome megjelenítőkhöz |

# 6. A Pollack Mihály Műszaki Főiskolai Kar honlapja[1]

## 6.1. Az oldalról röviden

Az oldal 2005. 02. 18.-án került lementésre, a HTML fájl az akkori képet tükrözi. Témáját tekintve az oldal egy főiskolai honlap, portál, amelyről tanárok és diákok egyaránt fontos információkat érhetnek el. Tájékoztatást nyújt az aktuális történésekről, eseményekről, megtalálhatóak a szervezeti egységek hivatkozásai, valamint elérhető az archívum is.

## 6.2. Külsőségek

Elöljáróban a külsőségeket érdemes kielemezni, ezután pedig következhet a megvalósítás technikájának vizsgálata. Az oldal színvilága meglehetősen harmonikus, kellemes perceket okozhat a böngészés ezen az oldalon, felépítése sem túlságosan bonyolult egy átlagos felhasználó számára. A megadott linkek segítségével egyaránt eljuthatunk magasabb, illetve alacsonyabb szintű helyekre is (magasabb szinten a tudományegyetem honlapja értendő).

Az oldal felépítése szerint négy főbb részre bontható, ezek a következők: a fejléc, mely a navigáció szempontjából a legfontosabb hivatkozásokat tartalmazza, a baloldali tartalomoszlop. A jobboldal felső harmadától kezdődik egy érdekességeket tartalmazó terület, majd ez alatt foglal helyet egy újabb navigációs rész. A híreket fontosság és aktualitás szerint blokkokba foglalták, ezeknek más-más színt adott a készítő. A fontos információk figyelemfelkeltőbb, míg az általános tudnivalók a kevésbé harsány kék és bordó színt kapták. Minden blokk háttérszíne a fejlécnek egy világosabb árnyalata, ezáltal a nagy kontrasztú sötét betűk jól olvashatóak. A különböző témájú hivatkozások különböző színben jelennek meg, segítve ezzel a könnyebb navigációt. A blokkokban általában csak egy rövid ismertető, ízelítő található, ha viszont a felhasználó további részletekre is kíváncsi, egy hivatkozás segítségével megtekintheti azt, beljebb jutva ezzel az oldal architektúrájába.

## 6.3. Problémák

A lapok hibátlanságát *Mozilla Firefox* böngésző segítségével a W3C on-line ellenőrzőjével [1], valamint egy ingyenesen hozzáférhető Web-szerkesztő programba

[1] *Az oldal elérhetősége: http://www.pmmf.hu*
*Az eredeti, illetve javított verziók megtalálhatóak a mellékletben.*

(*UltraEdit®-32 v11.00, http://www.ultraedit.com*) épített *Tidy* (*http://tidy.sourceforge.net, 1994-2003 W3C®*) nevű programmal ellenőriztem. Az oldal tulajdonképpen súlyos hibát nem tartalmaz, de az elemzés során 160 figyelmeztetést kaptam. Ezek túlnyomórészt az ajánlás szerint kötelező tulajdonságok elhagyására vonatkozott (pl.: kép objektumok „*alt*" tulajdonságának hiánya). A felmérésben résztvevő magyar felsőoktatási intézmények weblapjai közül egyik sem volt hibátlan, mindegyik tartalmazott több-kevesebb hibát. (A legkevesebb hibaszám 17 volt!)

Hozzáférhetőségi szempontból gondot okozhat, hogy egyik képet sem látták el magyarázó szöveggel (*ALT*, vagy *TITLE* attribútumok), a vakok, gyengén látók nem fogják megtudni, mi is található a képen. Különböző böngészőkkel vizsgálva, az tapasztalható, hogy az oldal mindegyikkel közel azonos megjelenést biztosít. Ez miért is probléma? Az oldaltervezők előszeretettel nyúlnak egy régi, jól bevált megoldáshoz: az oldalt táblázatokból építik fel, melynek szegélyei a felhasználó számára láthatatlanok. Az oldal tartalmát, pedig a táblázatok celláiban helyezik el. Mivel a táblázatok értelmezésében a Web-böngészők szinte teljes mértékben azonosak, kézenfekvő ez a megoldás. Nincs szélesség, illetve magasságbeli eltérés, minden ugyanúgy fog megjelenni a felhasználó képernyőjén, ahogyan a szerkesztő azt megalkotta.

Jogosan merül fel a kérdés ebben az esetben: előfordulhat-e az, hogy a felhasználó és a tervező különböző képernyőt használ? Nos, a felmérések azt mutatják, hogy a felhasználók túlnyomó része egy számítógéphez kapcsolt monitoron jeleníti meg a lapokat. Viszonylag kevés a kézi számítógéppel és a mobiltelefonnal Internetezők száma. Természetesen az ő igényeiket sem szabad szem elől téveszteni, sőt. Ha egy oldal megvalósítása olyan, hogy a vakok számára is élvezhető, akkor egészen biztosan az egészséges felhasználók számára is az. A monitorok jelentős része 1024*768 pixeles felbontásban működik. Amennyiben a felhasználó ennél nagyobb felbontásban tekinti meg a lapot, úgy az nem fog a megváltozott képernyőviszonyokhoz alkalmazkodni, a mérete ugyanakkora marad, mint az előző esetben, ennek oka, hogy a táblázatok méretét pixelben adták meg. A másik véglet, pedig az, amikor a felhasználó kisebbre állítja az ablakméretet, mint a forráskódban megadott táblázat mérete. Ilyenkor ide-oda görgethet az ablakban, ahhoz, hogy az oldal valamely részét megtalálja (vízszintes és függőleges irányba is). Mindenki ismeri ezt az esetet, és elmondható, hogy senki sem örül, ha ezzel kell szembesülnie.

A monitorok felbontását többnyire azért emelik magasabb értékre, hogy többet láthassanak az adott területből. De mivel a lap nem tud alkalmazkodni, a maradék hely üres

marad, pontosabban a háttérkép, vagy háttérszín tölti ki. Tehát belátható, hogy ez a megoldás nem a legoptimálisabb. Sőt, a HTML szabvány a táblázatot adatok rendezett megjelenítésére ajánlja, nem pedig az oldalon lévő elemek pozicionálására, a design kialakítására.

Tanulmányozva a forráskódot (melléklet), az is feltűnő lehet, hogy a kód telis-tele van a táblázat sorait és oszlopait meghatározó *tag*-ekkel, ami jelentős mértékben megnövelheti a lap méretét. A tendenciát jól mutatja, hogy az általam végzett nem reprezentatív felmérésben 20 találomra kiválasztott Web-design cégből 17 ezt a típusú pozícionálási módot használta saját oldalán, a referenciaként feltüntetett oldalaik nagy része is ezzel a megvalósítási móddal készült. Bizonyos értelemben ez érthető, hiszen például a nagy forgalmú portálok, nem engedhetik meg maguknak, hogy weboldaluk elrendezése esetlegesen megváltozzon, köszönhetően a felhasználó böngészőprogramjának. Egy fokkal jobb megoldása a táblázatos formázásnak, ha a táblázat méretét (különös tekintettel a szélességét) százalékos értékben határozzák meg. Ilyenkor az ablak teljes méretének arányában határozódik meg a táblázat szélessége, és így elvileg bizonyos értelemben alkalmazkodó képes lesz a lap.

A vizsgált oldal 800 pixeles oldalszélesség esetében látható teljes mértékben. Amennyiben ennél kisebb az ablak mérete, már hiányozni fog a tartalom egy része. Azonban hiába nyújtjuk az ablakot 1000 pixeles szélességre, a tartalom egyáltalán nem alkalmazkodik, mivel az oldal alapvetően középre igazított, így jobbról és balról is

3. ábra. Az eredeti oldal

4. ábra. A javított oldal

nagyjából 100-100 pixelnyi üres terület van. Ez 1280*960-as felbontáson már 240 pixeles üres területet jelent a két szélen. Ez összegezve 480 pixel! Nem is beszélve az 1600*1200-as felbontásról, ahol összesen már annyi az üres terület (jobb és baloldal összegezve), mint maga az oldal! A ***3. ábrán*** az eredeti oldal látható 1280*1024-as felbontáson, míg a ***4. ábra*** a javított oldalt szemlélteti ugyanazon a felbontáson.

Belátható tehát, hogy ez a megvalósítási technika mennyire nem használja ki a rendelkezésre álló lehetőségeket.

Az oldal jobb oldal alsó részében elhelyezkedő navigációs rész veti fel szabványossági, illetve hozzáférhetőségi szempontból a következő problémát: amint az észrevehető, a képekre kattintva léphetünk tovább bizonyos szervezeti egységek oldalára, tehát itt képet használ az oldal a navigációra. Ez nagyon elegáns, tetszetős megoldás, tulajdonképpen nem is lenne ez olyan rendkívüli probléma, de az oldal tervezője itt is kifelejtette az *„alt”* tulajdonságokat, ami a képen található információkról nyújt magyarázatot látássérült felhasználóknak. Amennyiben a böngészőben a „képek letöltése” funkció nincs engedélyezve, a felhasználó nem fogja tudni, hogy hova is vezet tulajdonképpen az adott link, ami nem megengedhető.

Némi odafigyeléssel azonban elkerülhető ez a probléma, és minimális időráfordítással több ember számára tehetjük elérhetővé az oldal tartalmát. Minden Web-fejlesztőnek célul kellene kitűznie, hogy minél többen láthassák, élvezhessék azt az oldalt, amit létrehozott. Sokakban felmerülhet a kérdés: megéri odafigyelni a fogyatékos felhasználók igényeire? A gyakorlat azt mutatja, hogy az oldaltervezők nagy részének erre a válasza: nem. A magyar felsőoktatási intézmények weboldalainak körében végzett (nem reprezentatív) felmérésem azt mutatja, hogy 10 esetből csupán csak 2-ről mondható el, hogy figyelmet fordítottak a fentebb kifejtett probléma megoldására. Ez az esetek 20 százaléka, ami meglehetősen kevés. Egy olyan oldal szerepelt a listában, ami minden képnél tartalmazta az ALT tulajdonságot, azonban sehol sem volt kitöltve. A szabvány ugyanakkor kötelezően írja elő az *„alt”* tulajdonság feltüntetését.

## 6.4. Forráskód

Minden weblap forráskódjának a dokumentumtípus meghatározásával kell kezdődnie. A forráskód tanulmányozása közben észrevehető, hogy a tervező a HTML 4.0 szabvány előírásait próbálta követni. Természetesen ez csak feltételezhető, hiszen a legtöbb mai szöveges, illetve grafikus HTML szerkesztő program új dokumentum létrehozásakor alapértelmezetten beilleszti valamely dokumentumtípus meghatározását. Általában ez választható, a HTML 4.0, HTML 4.01, vagy XHTML formátumok közül. Így tehát az is elképzelhető, hogy az oldal tervezőjének nem is áll szándékában az ajánlásnak megfelelni. A dokumentum fejléc (*HEAD*) részében több *META* elemet is találhatunk, melyek általánosságban a használni kívánt karakterkészletről (*Charset*), a dokumentum tartalmáról (*Description*), az oldal érvényességének lejárati idejéről (*Expires*), szerzőjének nevéről

(*Author*) szolgáltathatnak információt. Itt adható meg továbbá a keresőoldalak robotjainak az információ, hogy felvegyék-e a találati listába az adott oldalt, az oldal nyelve, valamint, hogy milyen szavak kulcsszavak (*Keywords*) illenek az adott oldal tartalmára (a szerverek nem csak a domain nevek között keresnek, hanem a letárolt index-adatbázisban is, amit az úgynevezett keresőrobotok gyűjtenek össze, a weben elérhető lapok átvizsgálásával). Összességében elmondható, hogy a tervező itt megadhat tulajdonságokat, amikhez értéket rendelhet. Az oldal készítője jelen esetben csupán a HTML (és a hozzá csatolt CSS) dokumentum megjelenítéséhez szükséges karakter-kódalap nevét adta meg. A megfelelő számú *META* elem elhelyezése azonban nagyon fontos dolog. Az ajánlás ugyan nem írja elő kötelezően, de használatuk segíthet például abban, hogy az oldalunk egy keresőszerver találati listájában előkelőbb helyet foglaljon el.
Több helyen észrevehető, hogy pozícionálásra a *tag*-ek *„align”* (igazítás) tulajdonságát használja a készítő, ami ugyancsak a „nem ajánlott” megoldások körébe tartozik.

Az oldal tehát meglehetősen sok hibát, illetve hiányosságot tartalmaz, adott volt hát a feladat, a lehető legtöbb hiba javítása, a rendelkezésre álló technikák segítségével. Az oldal tartalma, megjelenése néhány külső változtatástól eltekintve a régi maradt. Minden megvalósítási módnál törekedtem a lehető legjobb megoldásra, természetesen megvalósítható az oldal más módon is, ez csak egy megoldási lehetőség a sok közül. A formázás, illetve pozicionálás teljes egészében stíluslapok segítségével történt, ez biztosítja, hogy az oldal nagyfokú alkalmazkodó képességgel rendelkezik.

## 6.5. Problémák megoldásai

Talán a legfontosabb problémával érdemes kezdeni, ami a táblázatos formázás, tehát hogy minden az oldalon szereplő kép, szöveg egy láthatatlan táblázat valamely cellájában helyezkedik el. Tehát bármit is választunk ki az oldalról, az egészen biztosan egy téglalap alakú cellában található. A CSS segítségével megvalósított megoldás is tulajdonképpen ezen alapul. Ezzel a technikával elvileg az összes hasonló problémával rendelkező oldal átalakítható. Kisebb-nagyobb négyszögekből építettem fel az oldalt, ezekben helyezkedik el a tartalom. Fontos különbség azonban, hogy minden formázásra vonatkozó információ egy külön fájlban található, ezzel megvalósult a tartalom és a formázási tulajdonságok szétválasztása. A stíluslap egy „css” kiterjesztésű fájl (pl.: *stilus.css*), amit a *„link”* HTML tag segítségével kapcsolok az oldalhoz, megadva a stílusfájl típusát, valamint tartalmának formátumát. Például: *<link rel="stylesheet" type="text/css" href="default.css">*, ezt a dokumentum *HEAD* részben kell elhelyezni. Sokkal egyszerűbbé válik egy esetleges

arculatváltás ezzel a megvalósítási móddal, hiszen nem szükséges a HTML kódot módosítani, elegendő csupán a formázási tulajdonságokat (színeket, méreteket, elhelyezkedéseket) megváltoztatni. Ez egy nagyobb terjedelmű oldalnál jelentős munkamegtakarítást eredményezhet. Nem is beszélve arról, hogy több HTML oldal is használhatja ugyanazt a stíluslapot, tehát még több példányt sem kell tárolni belőle. Megoldásom lényegében hasonló a táblázatos formázáshoz, de annál sokkal többre képes, valamint a szabványoknak, ajánlásoknak is megfelel. Az ergonómiai megvalósítást is bizonyos mértékben figyelembe vettem, de szakdolgozatom témája ezt csak bizonyos mértékben igényli.

A megvalósítás módja a következő: mindenek előtt részletesen szemügyre kell venni az adott oldalt, világosan tudnunk kell, hogy mely részei tartoznak szorosan egybe. Az azonos egységbe szánt részeket egy csoportosító HTML elemmel (*DIV*) egységbe foglaltam, majd a csoportok egy-egy azonosítót kaptak (*CLASS*, illetve *ID*). Ezek között a különbség, hogy a *CLASS* tulajdonság által megadott név több blokkra is használható, az *ID* viszont egy egyedi azonosító, csak egyetlen elem rendelkezhet ezzel a névvel. Alaphelyzetben ezek a jelölőelemek semmilyen speciális jellemzővel nem ruházzák fel az általuk közrezárt tartalmat, csak stíluslapok használata esetén. [4]

Ha például több olyan rész is szerepel az oldalon, melyhez narancssárga 13-as, dőlt, jobbra igazított betűformázás szükséges, akkor létrehozhatunk egy ilyen osztályt a CSS fájlban *narancs* néven, melyre a fentebb felsorolt tulajdonságokat adjuk meg. Ebben az esetben azokra a csoportokra fog érvényesülni a megadott formázás, melyek a HTML dokumentumban *narancs* nevű osztályba tartoznak. Az ***5. ábra*** ismerteti, hogyan lehet a CSS fájlban egy osztályhoz tartozó tulajdonságokat létrehozni, míg a ***6. ábra*** bemutatja, hogyan alkalmazhatjuk ezt a HTML fájlban. Tekintsük a *DIV*-eket szabályos négyszögeknek, amibe azt a tartalmat tölthetjük bele, amire az adott helyen szükségünk van. Az így kialakított blokkokhoz (a szóismétlés elkerülése végett a továbbiakban használni fogom a réteg, tömb, egység, illetve tároló, konténer szavakat is) a stíluslapban tulajdonságokat definiáltam, ezek a tulajdonságok értéket kaptak a megjelenésnek megfelelően. Azon a ponton, ahol tömböt le szeretnénk zárni, egy lezáró

```
.narancs{
   color: #FFA500;
   font-size: 13px;
   font-style: italic;
   text-align: right;

}
```

5. ábra. „narancs” osztály létrehozása

```
<body>
<div class=”narancs”>
  Ez a szöveg narancs
  színű lesz
</div>
<body>
```

6. ábra. HTML kód

*tag*-et kell elhelyezni *( )*. Mivel a *DIV* egy alapvető HTML elem, ezért sok formázási tulajdonsággal rendelkezik: magasság, szélesség, szegély, margó, keret, háttér (lehet kép, vagy csak szín), a tartalom szövegtulajdonsága, láthatóság, átlátszóság, pozícionálás, körülfolyatás.

## 6.6. Megvalósítás

Ezzel a módszerrel például egy hírblokk megvalósítása a következő: létre kell hozni a globális tárolót, azaz azt a négyszöget (négyzet vagy téglalap), ami majd az egész blokkot magába foglalja (*global-kontener*). Mindenek előtt érdemes megadni az újonnan létrehozott egység szélességét, illetve magasságát, ezt a CSS fájlban kell megtennünk *(„width”* tulajdonság). A ***7. ábra*** szemlélteti a globális tároló megvalósítását. Elsődleges a szélesség megadása, mert a magasság meghatározása az egységben található szövegmennyiség függvényében történik. A méretek megadása történhet abszolút, illetve relatív módon. Az abszolút mód használata nem ajánlatos, hiszen ezzel a megoldással a blokk szélességét pixelben kell jelezni, és így elvész az alkalmazkodó képesség előnyének nagy része. Érdemesebb inkább a relatív megadási módot előnyben részesíteni, mert ebben az esetben a szélességet százalékos értékben kell megadnunk. Ez az érték figyelembe veheti az egész ablak, vagy a tartalmazó ős tároló méretét. Mint az később a HTML kódból látszódni fog, jelen esetben nincs a blokknak őse, tehát a szélessége mindig az aktuális ablakméret 25%-a lesz. Ennek a konténernek meghatározunk egy háttérszínt, illetve alapértelmezett betűszínt, majd lezárjuk, így amennyiben létrehozunk egy leszármazott blokkot, az már az ős méreteit fogja figyelembe venni (egyéb tulajdonság megadásának hiányában). Természetesen ezek csak az alapvető tulajdonságok, ha szükséges, akkor ennél sokkal pontosabban meghatározható a blokk külső megjelenése. Ha egy tároló már tartalmaz egy blokkot, akkor az alacsonyabb szintű a leszármazott, a magasabb szintű, pedig az ős vagy szülő.

```
.global-kontener{
   width: 25%;
   background-color: #A9A9A9;
   color: black;
   border: 1px solid black;
}
```

7. ábra. „global-kontener” létrehozása

```
.global-kontener .fejlec{
   background-color: black;
   color: white;
   width: 100%;
   text-transform: uppercase;
   font-weight: bold;
}
```

8. ábra. „fejlec” osztály létrehozása

Amennyiben a blokknak semmilyen magasabb szintű őse nincs, úgy a blokk szélessége a böngészőablak méretének függvényében kerül megjelenítésre. A globális

tárolón belül kell majd elhelyezkednie a blokk fejlécének, illetve a szöveget tartalmazó résznek. A fejléc kialakításánál (***8. ábra***) fontos szempont, hogy az egész fejléc azonos háttérszínnel rendelkezzen. Ezt a már meglévő blokk egy leszármazott tömbjének létrehozásával érhetjük el. Ezután következhet a szöveget tartalmazó rész elkészítése. Újabb tárolót kell létrehozni, melynek háttérszínét a jó olvashatóság érdekében nem érdemes élénk színűre választani, legoptimálisabb a világos háttér - sötét betű megvalósítás, ez a konténer a *„szoveg”* osztálynevet kapta (***9. ábra***). A szöveg elhelyezése után következhetnek a lezárások. Ez a fejléc-blokknál már megtörtént, tehát csak a szöveget tartalmazó, illetve az ős konténer nincs lezárva. A lezárási sorrend azonos típusú *tag*-eknél lényegtelen, mivel nincs különbség közöttük (pl.: * *), a lezárás helye viszont egyáltalán nem mindegy. Ha az egymásba ágyazás műveletébe belezavarodunk, akkor nem a várt eredményt fogjuk kapni. Ezért ez a folyamat kellő körültekintést igényel, mély beágyazás esetén érdemes a nyitó és záró blokkosító *tag*-eket magyarázattal ellátni, a későbbi könnyebb olvashatóság érdekében. A hírblokkot megvalósító HTML kód a ***10. ábrán*** látható. Ha a lezárás megtörtént, akkor a blokk készen van, ha szükséges, akkor a konténerek tartalmának formátumát tovább finomíthatjuk az igényeknek megfelelően. A végeredményt a ***11. ábra*** szemlélteti.

```
.global-kontener .szoveg{
   margin-top: 0.5%;
   padding: 1%;
   background-color: #EFEFEF;
   text-align: justify;
}
```

9. ábra. „szoveg” osztály meghatározása

```
<body>
<div class=”global-kontener”>
   <div class=”fejlec”>
      Hírblokk fejléce
   </div>
   <div class=”szoveg>
      Ide kerülhet a tartalom
   </div>
</div>
</body>
```

10. ábra. A blokk megvalósítása

A blokkok pozícionálását szintén pixelpontosan, illetve százalékosan tudjuk megvalósítani az oldal tetejéhez, illetve bal és jobb széléhez, esetlegesen egy ős konténerhez viszonyítva. Az oldallal kapcsolatos problémák között felsorolásra került a jobb oldal alján található navigációs rész is, mely képek általi navigációt tesz lehetővé (a cél neve a képen található). Ennek kiküszöbölésére terveztem egy szöveges navigációs részt, melynek elhelyezkedése ugyanaz, de az oldal kezelhetősége ez által jelentős mértékben javul. A navigációs blokk színeinek meghatározásánál az oldal


11. ábra. Az elkészült hírblokk

összképét vettem figyelembe. A pozicionálás terén a CSS széles teret enged a fantáziának, vannak beépített igazítások (*float:left;*, *float:right;*), azonban mi magunk is meghatározhatjuk a jobbra-balra igazítás mértékét, használva a *„margin”* tulajdonságokat, melyet minden irányra külön-külön megadhatunk (*margin-left, margin-right, margin-top, margin-bottom*)

A módosított verzióban az összes linket elláttam a *„title”* magyarázó szöveggel, ami nagy segítséget nyújt a hang alapú programokkal böngészők számára, hiszen bővebb információt tudhat meg az adott linkről. Ezzel a megoldással azonban a *„Kattints Ide”* típusú linkek nem fognak értelmet nyerni, fontos a beszédes, találó, és lehetőleg rövid hivatkozásnév. A böngészőprogramokban ez a kiegészítés úgy érzékelhető, ha az adott link fölé visszük az egérkurzort, nemcsak az állapotsorban jelenik meg a céloldal címe, hanem néhány pillanatig a link fölött tartva megjelenik a magyarázó szöveg is.

Ha a fenti tervezési lépéseket követjük, olyan oldalt hozhatunk létre, mely alkalmazkodni fog a böngészőablak méretéhez, a szöveg nem fog kilógni a képből, nem lesz szükség vízszintes irányú görgetésre. A képernyőn megjelenő javított verzió már csaknem a teljes területet felhasználja a megjelenésre, a baloldali rész a 64, míg a jobboldali rész 32 százalékát foglalja el a mindenkori képernyőméretnek. Így a széleken eloszló szabad terület összesen 4%, ami jelentős javulást jelent az előző, táblázatos megjelenítéshez képest. Ez 1600*1200-as felbontáson is csupán csak 64 pixelt jelent, a korábbi 800 pixellel szemben, ami majdnem 48 százalékot tett ki, tehát a javulás mintegy 45-46 százalék. Érdemes foglalkozni a HTML fájlok méretével is. Az eredeti verzió mérete 34 687 bájt, a javított verzió mérete, pedig lecsökkent 15 977 bájtra. Ez az eredeti fájlméret 46 százaléka, tehát a fájlméret a tartalom változatlansága mellet közel a felére csökkent! Mindkét verzió használ stíluslapokat, az eredeti változathoz kapcsolt stíluslap mérete 11 812 bájt, míg a javított verzióhoz 4 637 bájtos css kiterjesztésű fájl csatlakozik, ami 7 175 bájttal kevesebb. A számadatokból is látható tehát, mennyivel hatékonyabb a javított oldal.

A javított oldal letöltési ideje 2. táblázat

| **A kapcsolat sebessége (*Kbps*):** | **A letöltés várható ideje (*s*):** |
|---|---|
| 14.4 | 10.9 |
| 28.8 | 5.4 |
| 50 | 3.1 |
| 64 | 2.5 |
| 128 | 1.2 |
| 1500 | 0.1 |

Az eredeti oldal letöltési ideje 3. táblázat

| **A kapcsolat sebessége (*Kbps*):** | **A letöltés várható ideje (*s*):** |
|---|---|
| 14.4 | 23.9 |
| 28.8 | 11.9 |
| 50 | 6.9 |
| 64 | 5.4 |
| 128 | 2.7 |
| 1500 | 0.2 |

A jelenlegi kódméretek ismeretében meghatározható az oldal letöltési ideje, mely a különböző sebességű Internet-hozzáférések esetében a ***2.*** illetve ***3. táblázatban*** látható.

Ezek az értékek természetesen az oldalt kiszolgáló szerver terheltségének függvényében változhatnak, a feltüntetett értékek az optimális esetre vonatkoznak. Látható, hogy kellően gyors eléréssel nem észlelhető a méretbeli különbség, de a legelterjedtebb modemes kapcsolat (*64Kbps*) esetén a letöltési idő a felére csökken. A szakirodalom szerint nagyon fontos, hogy ne kelljen sokat várni az oldal letöltődésére. Látható, milyen drasztikus mértékben csökkenthető le az oldal mérete a CSS technika használatával. Természetesen egy jó Web fejlesztőnek célul kell kitűznie, hogy minél több információt nyújtson, a lehető legésszerűbb felépítésben, és törekednie kell a minél kisebb fájlméretre.

# 7. A Bánki Donát Gépészmérnöki Főiskolai Kar honlapja[2]

## 7.1. Vizsgálat

Az oldal felépítésére itt is jellemző a táblázatos formázás, ennek javítási lehetőségeit az előzőekben már ismertettem. Ami a szabványosság, illetve használhatósági szempontok szerint egy újabb problémát vet fel, az a keretek (*FRAME-ek*) használata. Az utóbbi 1-2 évben ugyan csökkenni kezdett a kereteket használó oldalak száma, azonban még mindig találkozhat a felhasználó olyan weblapokkal, ahol továbbra sem hagynak fel a *frame*-ek alkalmazásával.

A *frame*-es technológia alapja az, hogy a böngészőprogram képernyőjét két vagy több részre osztjuk fel, ezekben fog megjelenni a tartalom, ami külön HTML fájlokban van letárolva. A keretek alakja csak négyszögletes lehet, és ezek egymásba ágyazhatóak. Jellemző alkalmazási forma, mikor az oldalt ezzel a módszerrel bontják menü- és tartalomrészre. A keretekkel dolgozó oldalak esetében mindig létezik egy leírófájl, amiből az összes többit meghívhatjuk. Ezekben a fájlokban a kötelező HTML elemeken kívül csak a keretekre vonatkozó információk találhatók. Ez körülményesebbé teszi a hibaellenőrzés folyamatát, mert nem a kezdőlapot kell ellenőriztetni, hanem a keretekben lévő oldalakat külön-külön.

A *frame*-ek mérete itt is pixelpontosan, illetve relatív módon adható meg. A tervező azt is meghatározhatja, hogy a keretet a felhasználó átméretezheti-e, illetve ha a tartalom nem fér el az aktuális ablakméreten, legyen-e lehetőség görgetésre. A keretek használata mellett azt az érvet szokták felhozni, hogy függetlenek egymástól, mégis mindegyik kapcsolatban áll a másikkal. Régebben nem egy böngészőnél fordult elő, hogy nem támogatta a keretek megjelenítését, ilyenkor egy tájékoztató szöveget kaphatott a felhasználó. Az oldalakon belül elhelyezhetőek továbbá úgynevezett lebegő *frame*-ek (*IFRAME*), azonban ezek használata végképp nem ajánlott, mert nem mindegyik böngésző támogatja.

Az eredeti Bánki honlap 3 keretből épül, baloldalon a menüoszlop, jobb oldal felső részén a tartalomrész, míg a jobb oldal alján az oldallal kapcsolatos információk kaptak helyet. Egyik sem átméretezhető, és csak a függőleges keretek rendelkeznek szegéllyel, az alsót megpróbálja a készítő elrejteni. A keretek használata helyett érdemesebb itt is áttérni

[2] *Az oldal elérhetősége: http://www.banki.hu*
*Az eredeti, illetve javított verziók megtalálhatóak a mellékletben.*

a CSS segítségével történő megvalósításra, hatékonyabb megoldási forma. A javított verzióban az oldal két nagy konténerre lett felosztva, így baloldalra a navigáció, jobboldalra, pedig a tartalom és az oldalra vonatkozó információk összevonva kerültek.

Ezzel a megoldási módszerrel az oldal egy fájlban van letárolva, melynek mérete 5 912 bájt, ezzel szemben az eredeti példány (ami 3 részből adódik össze) 10 913 bájt nagyságú, tehát a méretcsökkenés ebben az esetben is 50 százalék körüli (pontosan: 54,2 százalék). Az eredeti honlap 4 450 bájt méretű stíluslappal rendelkezik, a javított verziót, pedig úgy sikerült megvalósítani, hogy még ez az érték is lecsökkent 3 327 bájtra, ezzel a stíluslap mérete 25 százalékkal csökkent. A jelenlegi fájlméretek esetében a letöltési idők a

A javított oldal letöltési ideje 4. táblázat

| **A kapcsolat sebessége (*Kbps*):** | **A letöltés várható ideje (*s*):** |
|---|---|
| 14.4 | 4.0 |
| 28.8 | 2.0 |
| 50 | 1.2 |
| 64 | 0.9 |
| 128 | 0.5 |
| 1500 | 0.0 |

A javított oldal letöltési ideje 5. táblázat

| **A kapcsolat sebessége (*Kbps*):** | **A letöltés várható ideje (*s*):** |
|---|---|
| 14.4 | 6.8 |
| 28.8 | 2.7 |
| 50 | 2.0 |
| 64 | 1.6 |
| 128 | 0.8 |
| 1500 | 0.0 |

**4.** és az **5. táblázat** szerint alakulnak.

Az általam felvázolt megoldási forma azzal a kompromisszummal jár, hogy a menüsor is gördülni fog abban az esetben, ha a tartalom részben hosszabb szöveg van. Tehát a jelenlegi verziótól eltérően előfordulhat, hogy az oldal alján lévő tartalom megtekintésekor nem fog látszani a bal oldali menü. Ez a probléma is könnyedén megoldható, ha az oldal tetejére mutató horgonyokat (*anchor*) helyezünk el az oldal alján, nagyon hosszú szöveg esetén a szöveg közé beszúrva akár többet is. Ezek segítségével újratöltés nélkül az oldal tetejére juthatunk.

Hozzáférhetőségi szempontból ugyancsak problémát jelent az eredeti verzióban, hogy az iskola 125 éves évfordulóját hirdető szöveg egy képbe van helyezve, amit nem láttak el magyarázó (*ALT*, vagy *TITLE*) szöveggel. Így az a felhasználó, akinek a böngészője nem tölti le a képeket, nem fog tudomást szerezni az intézmény jubileumáról. Ez az információ a javított verzióban már szöveges információként szerepel. Mivel a képen található szöveg valamilyen grafikus szerkesztőprogrammal készült, ezért a javított verzió egyszerű „szöveges” megoldása közel sem annyira látványos. Általánosságban az egérrel történő kijelöléssel lehet a szöveges, illetve a képpel megjelenített információkat megkülönböztetni.

# 8. Az Indygo apróhirdetési oldal[3]

Végül egy kereskedelmi oldal elemzése következik. Az Indygo fő profilja az apróhirdetés, de foglalkozik továbbá ingyenes szolgáltatásokkal, mint például képeslapok, on-line játékok, chat, valamint fórum.

## *8.1. Vizsgálat*

Az oldalhoz egyetlen stíluslap kapcsolódik, ez azonban kizárólag szövegre, illetve hátterekre vonatkozó utasításokat tartalmaz, pozícionálási információt egyáltalán nem. Ezt teljes egészében táblázatokkal, illetve a HTML-ben található formázó elemekkel valósítják meg. Mint az a javított oldalból is látszik, ebben az esetben is sikeresen használható az előzőleg már ismertetett eljárás. Ezt a megoldást ismertettem az oldal készítőjével is, aki mégis elutasította a CSS-el történő formázásra, pozícionálásra való áttérést. Indoklása egyszerű volt: amíg az összes böngésző nem képes egyformán értelmezni a szabványt, addig ő inkább megmarad a táblázatos megoldásnál. Az oldal dinamikus feltöltésű, tehát az információk feltöltése a lapra adatbázis segítségével történik PHP, illetve CGI *script*-ek segítségével. Ez azonban egyáltalán nem indokolja az ismertetett technika elutasítását, hiszen ezt a megoldást éppúgy be lehet építeni a szkript kódjába, mint a táblázatok esetében. A stíluslapok azonban úgymond böngésző-kompatibilissé tehetőek, akár JavaScript segítségével is. Kiolvasva a böngészőprogram információját, módosításokat lehet végrehajtani a stíluslapon, így az esetlegesen ismert félreértelmezéseket bizonyos mértékben korrigálni lehet.

## *8.2. Forráskód*

Ha valaki csak néhány pillantást is vet a lap forráskódjára, azonnal feltűnhet, hogy borzasztóan bonyolult a felépítése. Nem ritka az 5-6 egymásba ágyazott táblázat sem, ezeknek több sora és oszlopa is van. Abban az esetben, ha valaki módosítást, változtatást szeretne végrehajtani, meglehetősen nehéz dolga lesz, mire megtalálja azt a táblázatot, ami valójában tartalmazza az adott információt (feltéve, ha nem ismeri az oldal pontos felépítését). Megfigyelhető, hogy a kód rengeteg méretező tulajdonságot is tartalmaz, ami szintén hozzájárul a fájlméret megnövekedéséhez, mivel azokat minden egyes elemhez külön-külön meg kellet adnia a készítőnek. A lap *„meta”* elemekkel megfelelő mértékben el van látva, megtalálható az oldal rövid leírása, a keresők számára a kulcsszavak, valamint

[3] *Az oldal elérhetősége: http://www.indygo.hu*
*Az eredeti, illetve javított verziók megtalálhatóak a mellékletben.*

az oldal készítőjének neve, és az oldal tartalmának pontos meghatározása. Az oldal méreteinek meghatározása vegyes képet mutat, néhol relatív, máshol abszolút méretmegadást használt a készítő, de az oldal eltérő felbontások esetében is ugyanúgy néz ki, tehát a legfelső szinten lévő táblázat szélessége abszolút módon lett meghatározva, melynek értéke pontosan 763 pixel. Az oldal tehát 800*600-as felbontásra van optimalizálva. A javított verzióban ez a probléma megoldódott, bármekkora képernyőfelbontás esetén alkalmazkodni fog az oldal.

Aki részletesebben is szemügyre veszi a javított oldal forráskódját, észreveheti, hogy nagyon sok, apró konténerből épül fel egy-egy rész, ez azonban mégsem veti fel az átláthatatlanság problémáját (mint néhány táblázatos megoldásnál), mert az összes blokk osztálynévvel van ellátva, ezért a szemlélődő szinte mindig tudni fogja, hogy melyik résznél tart a forráskódban. A jó átláthatóság érdekében érdemes „beszédes”, ám mégsem túl hosszú azonosító, illetve osztályneveket alkalmazni, jelentősen hozzájárulva ezzel a későbbi újrahasznosíthatósághoz, könnyebbé válik az esetleges kiegészítés, javítás. Az osztálynevek meghatározásánál néhány program (pl.: *Top Style Pro, letölthető: http://www.bradsoft.com*) nem javasolja az alsóvonás („_”) karakter használatát, mert néhány régebbi verziójú böngészőnek gondot okoz ezek felismerése. A stíluslapokban helyette a kötőjelet („-”) alkalmaztam (pl.: *„kontener-fejlec”*).

Összességében látható, hogy a felmerült problémák hasonlóak az előzőekhez, ezért a már ismertetett eljárás ebben az esetben is használható.

# 9. Összegzés

Az Internet megjelenése hatalmas változást hozott az információk megszerzésének módjában. Az emberek gyorsan, könnyen juthatnak olyan mennyiségű információhoz, amely néhány évtizeddel ezelőtt elképzelhetetlen lett volna. Azonban nem elegendő csak azzal foglalkoznunk, hogy meg tudjuk-e szerezni az adott információt, vagy sem. Mindenképpen érdemes figyelmet fordítanunk arra is, hogy ezek az ismeretek milyen környezetben, milyen módon jelennek meg a felhasználó számára. A szakdolgozatban ismertetett megoldási móddal a táblázatos pozícionálást használó oldalak mérete közel a felére csökkenthető, köszönhetően a CSS-el történő elrendezésnek, formázásnak. Mindez úgy tehető, hogy a tartalom változatlan marad, sőt az adott oldal még használhatóbbá válik, képes lesz alkalmazkodni a felhasználó ablakméretre vonatkozó beállításaihoz is.

Megállapítható, hogy nem okolhatóak egyértelműen a webfejlesztők, azért hogy még mindig a régebbi megvalósítási technikát használják. A böngészőprogramok fejlesztői is jelentősen hozzájárultak ahhoz, hogy ez a folyamat elindult, és még napjainkban is tart. De mint az látható volt, némi új ismeret megszerzésével a táblázatos formázás lényegében megszüntethető, helyettesíthető CSS alkalmazásával, és elmondható, hogy ennek használatával minden webfejlesztő saját munkáját könnyíti meg.

# 10. Ábrajegyzék, táblázatjegyzék

# 11. Irodalomjegyzék

**[1] World Wide Web Consortium (W3C)**

(*http://www.w3c.org*)

**[1.1]** HTML 4.01 Specification
*http://www.w3.org/TR/1999/REC-html401-19991224*

**[1.2]** XHTML™ 1.0 The Extensible HyperText Markup Language
*http://www.w3.org/TR/2002/REC-xhtml1-20020801*

**[1.3]** Cascading Style Sheets, level 1
*http://www.w3.org/TR/1999/REC-CSS1-19990111*

**[1.4]** Cascading Style Sheets, level 2
*http://www.w3.org/TR/1998/REC-CSS2-19980512*

**[2] Paczona Zoltán – HTML technikák a gyakorlatban**

Computer Panoráma Kiadó Budapest, 2001 (59-63, 71-98)

**[3] László József - Mindenkinek az INTERNET-ről**

ComputerBooks Kiadó Kft Budapest. 2000, ISBN: 963 618 239 6 (230-237)

**[4] Gál Tibor – Webprogramozás**

Műegyetemi Kiadó Budapest 2004, ISBN 963 420 800 2

**[5] Htmlinfo.Poliyhistor.hu**

http://htmlinfo.polyhistor.hu