Debreceni Egyetem
Informatika Kar

E-learning, digitális tananyagok készítése: módszertan és megvalósítás

Témavezető:
Dr. Bujdosó Gyöngyi
egyetemi adjunktus

Készítette:
Csizmadia Balázs
informatika tanár

Debrecen
2008

1. Bevezetés **5**
2. Az e-learning **7**
2.1. Az e-learning története **8**
2.2. A számítógéppel támogatott oktatás/e-learning **9**
2.3 Az e-learning képzési formái **10**
2.3.1 Tanuló által irányított e-learning **11**
2.3.2 Elősegített (facilitated) e-learning **11**
2.3.3. Oktató által irányított e-learning **11**
2.3.4 Beágyazott e-learning **11**
2.3.5 Telementoring és e-coaching **12**
2.4 Az e-learning tanulmányi keretrendszere és tartalomkezelő rendszere **12**
2.5 Az e-learning tartalom felépítése **14**
2.6 Az e-learning előnyei és hátrányai **14**
2.7 Az e-learning piacot meghatározó szereplők **15**
2.7.1 A közoktatás **15**
2.7.2 A felnőttképzés **16**
2.7.3 Vállalati képzések **16**
3. Az e-learning szabványosítása **17**
3.1 Az IEEE e-learning szabványa, a LOM **17**
3.2 Az ADL SCORM szabványrendszere **18**
4. Az e-learning módszertana **20**
4.1 A tananyag tartalmának összeállítása **20**
4.2 A tananyag szövegének megformálása **21**
4.3 A tananyag hipertexes tagolása, kép- és hangelemeinek kiválasztása **21**
5. HTML, CSS, JavaScript **22**
5.1 HTML (HyperText Markup Language) **22**
5.1.1 A HTML dokumentum felépítése **22**
5.1.2 A HTML dokumentum szövegtestének felépítése **24**
5.1.3 A HTML dokumentum címszintjeinek felépítése **25**
5.1.4 Bekezdések a HTML dokumentumban **25**
5.1.5 Kereszthivatkozások HTML dokumentumok között **26**

5.1.6 Karakterformátumok a HTML dokumentumban 27
5.1.7 Képek elhelyezése a HTML dokumentumban 28
5.1.8 Listák a HTML dokumentumban 28
5.1.9 Táblázatok 29
5.1.10 Keretek 31
5.1.11 A HTML és a multimédia 32
5.2 Stíluslapok, CSS 33
5.2.1 A stíluslapok kialakulásának története 33
5.2.2 Beágyazott stíluslap 35
5.2.3 Külső stíluslap 35
5.2.4 Elemhez rendelt stílus 36
5.2.5 Importált stíluslap 36
5.2.6 Stílusok formátuma 36
5.2.7 Kiválasztók 37
5.2.8 Elem kiválasztás 37
5.2.9 Osztály alapú kiválasztás 37
5.2.10 Azonosító alapú kiválasztás 38
5.2.11 Helyzetérzékeny kiválasztás 39
5.2.12 Szülő-gyermek kiválasztás 39
5.3 JavaScript 39
5.3.1 JavaScript és HTML 40
5.3.2 Adatok, változók és kifejezések 41
5.3.3 Utasítások 42
5.3.3 Objektumok 43
5.3.6 Beépített objektumok 45
5.3.7 Eseménykezelés 46
5.3.8 Állapotsor kezelés 48
6. A weboldal bemutatása 50
6.1. A weboldal bemutatása az e-learning oldaláról 50
6.2. A weboldal bemutatása az informatika oldaláról 51
6.3. Az oldal használata 55
6.4. Tesztelés, üzembe helyezés 56

7. Összefoglalás **57**
Irodalomjegyzék **58**

1. Bevezetés

Az oktatás történetében rendkívül fontos szerepe van a számítógép és az Internet megjelenésének. Az első számítógépeket hatalmas - egy teljes termet elfoglaló - méretük, és bonyolult kezelésük miatt, csak a felsőoktatásban használták. Ahogy az információs és kommunikációs technika fejlődött, úgy vették használatba a számítógépeket az általános- és középiskolák, valamint a felnőttoktatással foglalkozó intézmények.

Az Internet általános használata csak a 1990-es évektől, a WWW (World Wide Web) megjelenése után terjedt el. Ma már ott tartunk, hogy az Internet segítségével naprakész információkat szerezhetünk, és számtalan szolgáltatását használva, mint például az e-mail, chat, videokonferencia, blog a tőlünk távolabb élő emberekkel is kapcsolatot tudunk tartani. Létrejött egy új fogalom, az információs társadalom.

Az információs társadalom nevezhető tudás-alapú társadalomnak is, ahol a tudás birtokosa egyben a munkaerőpiac legkeresettebb embere. Az információ a gazdaság mozgatórugója is. Az egyén tudása, készsége, ismeretanyaga az idő múlásával elévül, szinten tartása végett újból tanulnia kell, akár munka mellett. Az információs társadalom egyik alapvető tétele az élethosszig tartó tanulás, ami jelentős időráfordítást is igényel.

Ebből egyenesen következik, hogy az oktatásnak változnia kell, követnie, támogatnia szükséges az élethosszig tartó tanulást. Az iskolának nemcsak a tananyagot kell megtanítani a tanulóival, hanem azt a képességet is, hogy önállóan tudjanak tanulni. Ebben a folyamatban megváltozik a tanár szerepe is.

Az 1980-as évektől kezdve majdnem minden országban megtalálható a távoktatás. A távoktatás lehetőségeit kihasználva a munkaerő-piaci keresletnek megfelelően folyhat a felnőttek képzése, vagy a szükség szerinti átképzése. A vállalatok számára azonban ez az oktatási módszer nem költségkímélő, mivel akár munkakiesést is okozhat egy-egy tanfolyam elvégzése. Az információs és kommunikációs technika eszközeit a távoktatás beleépítette a módszertanába, ezzel is segítve az otthontanulás lehetőségét. Az Internet széleskörű elterjedésekor megszületik egy új oktatási forma, az e-learning.

Az e-learning, vagy magyarul elektronikus tanulás, egyesíti a távoktatást a hálózati technológiával. Míg a távoktatásnál a hallgatók nyomtatott tananyagot kapnak kézhez, és távolról küldik be dolgozataikat, az e-learning tanfolyamot végző hallgató egy másik eszközrendszerrel: a számítógép, illetve hálózatok segítségével oldja meg mindezt, megszabadulva a merev időkorláttól és a lineáris leckefelépítéstől. Bárhol, bármikor elérheti a tananyagot, csupán egy számítógépre és egy Internet kapcsolatra van szüksége.

Az e-learning tanfolyamokat csak abban az országban lehet sikeresen szervezni, ahol megfelelő sávszélességű Internet, és számítógép ellátottság van. A szélessávú Internet-kapcsolat előfizetése, az Internet-szolgáltatók piaci versengése miatt elfogadható árra csökkent Magyarországon, valamint a számítógép ellátottság is növekvő tendenciát mutat az utóbbi években. Az e-learning folyamatos fejlődése nehezíti a szabványosítást. Hivatalos szabvány nem létezik, bár számos szervezet foglalkozik a fejlesztésükkel. Ahhoz, hogy egy intézmény e-learning kurzust indítson, szüksége van egy keretrendszerre és egy e-learning módszertanú tananyagra. Számos keretrendszer közül választhatunk, de a tapasztalat azt mutatja, hogy egy jó tananyag vásárlása, netán elkészítése, sokkal komolyabb problémát okoz. A pedagógusok képzésében, illetve átképzésében még nem mindenhol van jelen az e-learning módszertanának oktatása.[1]

Én a Hajdú-Bihar Megyei Egészségbiztosítási Pénztárnál dolgozom, mint szakmai rendszergazda. A munkaköröm nagy részét a Pénztárnál használt DigiNet elnevezésű iktató, irat- és ügyfolyamat-kezelő rendszer rendszergazdai feladatainak az ellátása teszi ki, amibe beletartozik az ügyintézők oktatása is. A programhoz tartozik egy papír alapú felhasználói utasítás, ami kb. 80 oldal. Minden ügyintézőnek ezt kinyomtatni elég költséges volna, ezért próbáltam egy költséghatékonyabb módszert kitalálni, ami segítség lehet a dolgozóknak. Meglátásom és tapasztalataim szerint a közszférában csak a hagyományos oktatási modell érvényesül. Ez főleg abból adódik, hogy erre van igény és felkészültség. Ezt támasztotta alá, amikor más közigazgatási szerveknél érdeklődtem a munkahelyi oktatásról. Ezért a dolgozatom részeként egy webes felületű használati utasítást készítettem, ami egy tanár-centrikus tanulásra épít. Ez lényegében egy tudásátadást jelent.

[1] Kovács Ilma: Új út az oktatásban?, Budapesti Közgazdaságtudományi Egyetem és a Professzorok Háza Felsőoktatási Koordinációs Iroda, Budapest, 1997

Szakdolgozatom következő fejezeteiben az e-learning-gel fogalakozom, valamint bemutatom azokat a nyelveket amelyet felhasználtam a weboldal elkészítéséhez és magát az oldalt is ismertetem.

2. Az e-learning

Az e-learning (elektronikus tanulás) fogalmát többféle értelemben használják, magát az angol szót is többféleképpen írják (e-learning, elearning, eLearning). Legtágabb értelemben technológiával támogatott tanulás (technology supported learning) számítógép segítségével történő tanulás, digitális tananyag segítségével történő tanulás. Ilyen értelemben jelentheti a CD-ROM segítségével egyénileg történő tanulást is multimédiás számítógéppel. Mások elsősorban vagy kizárólag az internet vagy intranet alapú tanulást a hálózaton szervezett és folytatott oktatást értik rajta. Így az informatikai technológiát hasznosító távoktatást is jelöli. Az Internet alapú képzés (Internet-based training), az online tanulás (online learning) kifejezések használata mellett a kilencvenes évek második felétől, de különösen az ezredfordulótól, mindinkább az e-learning kifejezés vált általánossá, felölelve az iskolarendszerű oktatás megújítását, a felsőoktatási alkalmazást, a szakmai továbbképzések és az egyes egyén önképzésének új lehetőségeit.

Miközben mind többféle e-learning termék és szolgáltatás jelent meg elsősorban az Egyesült Államokban, majd világszerte, és valóságos e-learning ipar van kialakulóban az Európai Unió e-learning kezdeményezése nyomán, 2000 óta fokozatosan az oktatás és képzés új koncepcióját hordozza ez a kifejezés. Ez pedig az új alapkészségeket kifejlesztő és azokra építő, mindenki számára elérhetővé tett, egész életen át tartó tanulás programja és lehetősége.

Az információs társadalom projekteknek és az e-learning kezdeményezéseknek szükségképpen szerves részei a tanárképzés és tanártovábbképzés megújítására irányuló programok.[2]

[2] Magyar Gallup Intézet http://ip.gallup.hu/elearning/index.htm

2.1. Az e-learning története

Az e-learning története több szakaszra bontható aszerint, hogy milyen tömegkommunikációs eszközök alkalmazásával helyettesítették a hagyományos, tantermi körülmények között zajló, közvetlen tanár-diák viszonyra építő oktatási formákat.

Az első szakasz az 1840-es évekre tehető, ekkor terjedt el a bélyegek használata és ezáltal a levelező oktatás. A Nagy-Britanniában élő Isaac Pitman neve fémjelzi ezt a korszakot. Ő volt aki a gyorsírást oktatta ily módon. 1917-ig más országokban is megjelent a levelező oktatás: Németország 1856; Franciaország 1877; Svédország 1898, Amerikában 1891, illetve Ausztráliában 1914.

A második szakaszban a bélyeg helyett az éter kapott főszerepet. A tananyagot rádión keresztül jutatták el a diákokhoz. Ez a kísérlet is Nagy-Britanniából indult ki, de korántsem akkora sikerrel, mint a levelező oktatás. Sokkal inkább a hagyományos képzési módszerekkel párhuzamosan, azok kiegészítésére használták. 1927-ben volt az első kísérlet, a BBC iskolarádiós műsorok sugárzását tűzte műsorára. Később ezt a módszert is átvették más országok: Franciaország 1937; Amerikai Egyesült Államok 1939; Ausztrália 1942; Szovjetunió 1920-as évek, egyéb volt szocialista országok 1950-es évek, de sehol sem tett szert vezető szerepre az alternatív oktatási módszerek között.

A harmadik szakasz az 1980-as évekre tehető, amikor a fejlett országokban megjelenik a számítógéppel támogatott oktatás (Magyarország 1990-es évek). Ez a fajta oktatási mód új lehetőségeket nyit meg az oktatás, képzés területén, azáltal, hogy lehetővé teszi az eddig csak elszigetelten alkalmazható eljárások immár együttesen történő felhasználását. A levelező oktatás csupán vizuális úton juttatta el a diákokhoz a tananyagot. A rádió útján történő képzés ezzel szemben éppen, hogy vizuális élményt nem nyújtott, csak a hallásunkra „összpontosított”. A számítógép alkalmazásával lehetővé vált a többi érzékszervre történő együttes hatásgyakorlás anélkül, hogy ehhez különböző eszközöket kelljen felhasználnunk. A számítógép ugyanis lehetővé teszi az audiovizuális eszközök egyszerre történő alkalmazását.[3]

[3] Kovács Ilma: Új út az oktatásban?, Budapesti Közgazdaságtudományi Egyetem és a Professzorok Háza Felsőoktatási Koordinációs Iroda, Budapest, 1997

2.2. A számítógéppel támogatott oktatás/e-learning

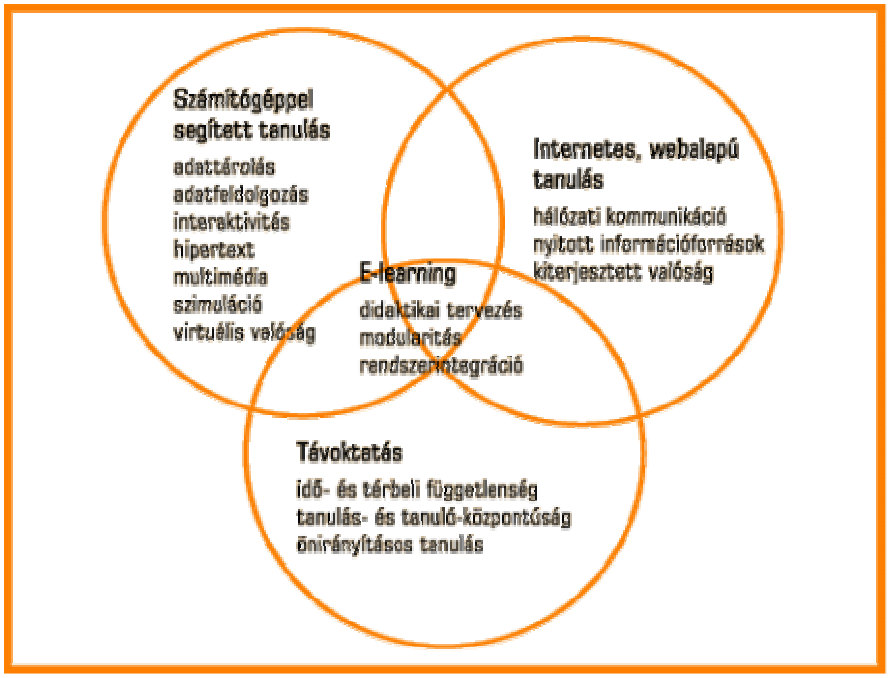
Az előző fejezetben tárgyalt történeti rész harmadik szakaszát tovább boncolgatva juthatunk el a korábban már definiált e-leraning-hez.

Az e-learning korai formája a CBT (Computer Based Training), azaz számítógéppel segített tanulás, ahol az oktatóanyag valamilyen digitális adathordozón található (például: floppy, CD stb.). Ezáltal a tananyag hordozhatóvá vált, a hallgató bármikor és akárhányszor lejátszhatta a tanulás során. A hely és idő korlátjai a hallgató igényeihez mérten kibővülnek. Hátránya, hogy a tanár és a tanuló között semmilyen kapcsolat nem épült ki, és csak az adathordozón található tartalmat közvetítette.

A technológiák fejlődésével lehetőség nyílik rá, hogy az elektronikus tanulás valódi képzésmenedzsmenttel társuljon. Ezt az oktatástípust WBT-nek (Web Based Training), azaz számítógépes hálózaton keresztül megvalósuló tanulásnak, on-line learning-nek nevezzük. Az oktatás egy új elemmel bővül, megjelenik a hálózati kommunikáció, így a hallgató a tanárral e-mail, chat, fórum, videokonferencia formájában tartja a kapcsolatot.

E-learning-ről csak 1990-es évek elejétől beszélhetünk, ekkor jelent meg az Egyesült Államokban, mint képzési forma. Európában az 1990-es évek végén honosodott meg az e-learning. Az oktatás egy informatikai hálózaton - ami lehet LAN, WAN, Internet vagy intranet – történik, aszinkron, illetve szinkron formában. A kommunikáció ezen a számítógépes hálózaton keresztül történik, a hallgató és a tanár között. Az oktatási-tanulási folyamatok tervezése, szervezése, kivitelezése és értékelése is a számítógépes hálózat segítségével történik.

Az e-learning fogalma és története áttekintése után az 1. ábra mutatja meg, hogy milyen oktatási és tanulási formákból tevődik össze az e-learning.



1. ábra
Az e-learning összetevői[4]

2.3 Az e-learning képzési formái

Az e-learningnek két fajtája alakult ki: a szinkron és az aszinkronképzés. A szinkron képzésnél a tanár és a tanuló egyidőben van jelen az oktatási folyamatban, a hallgató bármikor kérdést tehet fel az oktatónak, amire azonnal választ kap. Hátránya ennek a képzésnek, az időbeli kötöttség. Az aszinkron képzésnél a tanár és a tanuló időbeli és térbeli teljes elkülönülését tételezi fel, azaz a hallgató önállóan és egyedül veszi át a leckét.

Az e-learningnek az alábbi típusait definiálhatjuk:

- tanuló által irányított,
- elősegített (facilitated),
- oktató által irányított,
- beágyazott e-learning,
- telementoring és e-coaching.

[4] Komenczi Bertalan: Didaktika elektromagna? Az e-learning virtuális valóságai
http://www.oki.hu/oldal.php?tipus=cikk&kod=2004-11-ta-Komenczi-Didaktika#top

2.3.1 Tanuló által irányított e-learning

Ennél a típusnál a tanuló nincs kapcsolatban az oktatóval és más diákokkal, csak a tananyaggal. Az oktatóanyag tartalmazza az összes útmutatást, magyarázatot. Nincs mód az ellenőrzésre, nem lehet mérni, hogy a diák mennyit tanult. A felhasználó böngészőn keresztül használja az e-learning lehetőséget.

2.3.2 Elősegített (facilitated) e-learning

A hallgató tud kommunikálni más tanulókkal vagy a facilitátorral. A facilitátor nem tanít, csak segít a problémák megoldásában. Jogosult osztályozni, kiértékelhet feladatokat és dolgozatokat. A kommunikáció színtere egy fórum, de a facilitátor minden hallgatóval külön fel tudja venni a kapcsolatot.

2.3.3. Oktató által irányított e-learning

A hagyományos távoktatási módszerek kiegészítve a webtechnológiával. Ebben az oktatási formában valós idejű kommunikáció folyik: video- és audiokonferencia, chat, képernyő- vagy alkalmazásmegosztás, telefonbeszélgetés. Itt is használnak fórumot, ahol a hallgatók egymással, illetve az oktatóval megbeszélhetik a problémákat, feladatokat. Ennél az e-learning típusnál hátrány lehet az alacsony sávszélesség, mivel egy videokonferenciához szélessávú Internet szükséges.

2.3.4 Beágyazott e-learning

A beágyazott e-learning esetén beépített oktatás vagy segítségnyújtás található, amit a felhasználó azonnal használhat, ha segítségre van szüksége a probléma megoldásánál. Ez a beépített oktatás egy program, amit általában a tanuló gépére kell telepíteni. Itt nincs közvetlen kapcsolat a hallgató és az oktató között. Kommunikáció csak a diáktársakkal lehetséges fórum segítségével.

2.3.5 Telementoring és e-coaching

A mentori kapcsolat általában hosszú idejű. A mentor és a hallgató között videokonferencia, internettelefon és más együttműködési eszközök biztosítják a kapcsolatot. Hagyományos értelemben vett oktatás nincs, mivel a mentor inkább tudást, tapasztalatot ad át. On-line coaching esetében rövid és jól definiált probléma megoldása történik. Itt az on-line coach konzulens szerepet tölt be.[5]

2.4 Az e-learning tanulmányi keretrendszere és tartalomkezelő rendszere

Az e-learning tanulmányi keretrendszere az LMS (Learning Management System), amely az alapvető oktatási szervezési feladatokat, tananyagok megjelenítését, valamint különböző tanulmányi funkciókat lát el. Lényegében egy virtuális osztályteremként is felfoghatjuk, melyhez különböző tananyagokat tudunk hozzárendelni egy adott szabvány szerint. Az LMS rendszerek struktúrája nagyon változó, de egy közös pontjuk van: mindegyik rendszer a hallgatókról és kurzusokról információkat tárol.

Egy LMS kiválasztásakor fontos szempont:

- a hallgatói jelentkezés automatikus kezelése,
- előre elkészített tananyagok importálása,
- nem az LMS-ben lévő tananyagok kezelése,
- más rendszerekhez való integrálhatóság,
- automatikus számlázási lehetőség,
- automatikus hallgatói esemény követése,
- távoli adminisztráció lehetősége.

Mivel az LMS-eket intézményeknek megfelelően hozzák létre, nem volt biztosított a hallgatók számára az átjárás lehetősége. Ez a szemlélet a Web fejlődéstörténet 1.0-ás szakaszában jellemző. A Web fejődése kihat az e-learning fejlődésére. LMS+tananyag → e-learning 1.0.

[5] Digitális taneszközök használata a szakképzésben
http://www.sulinet.hu/ikt/17_szakmacsoport/rendeszet/rendeszet_06.html

A Web 2.0 az „olvasott web”-ről az „írott-olvasott web” irányába változik. Ez elsősorban nem technológiai, hanem szemléletbeli változást jelent. Az új irány a tanulóközpontú szemlélet. Erre épülő személyes tanulmányi környezetet hoztak létre, aminek a neve PLE (Personal Leraning Environment). Ezek a rendszerek már tartalmazzák a blog írás lehetőségét, egy helyen több tucat weboldal tartalmának figyelését (RSS, Rich Site Summary), kapcsolatok ápolását (FOAF). A hallgató létrehozza saját, egyéni tanulmányi környezetét, aminek segítségével rákapcsolódik a kívánt oktatói keretrendszerekre. A nyílt, rugalmas PLE-k tanulóközpontú szemlélete komoly kíhívás a zárt, hagyományos intézménycentrikus oktatási módszertanra épülő LMS rendszereknek. Tanulóközpontúság+web alapú tanulás → e-learning 2.0.[6]

Az LMS és PLE összehasonlítását az alábbi táblázat szemlélteti.[7]

LMS	PLS
rugalmatlan	rugalmas
intézmény-központú	tanuló-központú
formális	kötetlen
zárt rendszer	nyitott rendszer
alkalmazás	platform

Az LCMS (Learning Content Management System) rendszerekhez olyan szoftverek tartoznak, amelyeket elektronikus kurzusok fejlesztéséhez használnak. Az LCMS rendszer nagy segítség lehet a szakértők, oktatók, média-készítők, programozók számára. A hallgatók egyszerűen ki tudják választani a számukra szükséges anyagokat.

A LCMS biztosítja:

- a didaktikus feldolgozást,
- a tartalomfejlesztést,
- megosztást,
- terjesztést,
- a tartalom változtatásának lehetőségét.

[6] Hazai e-learning fejlődéstörténet http://www.crescendo.hu/2007/06/19/hazai-e-learning-fejlodestortenet
[7] Egyéni tanulmányi környezet (PLE) cikk alapján http://www.crescendo.hu/node/5

2.5 Az e-learning tartalom felépítése

Tanulási tartalom (learning content): a digitalizált tananyagot jelenti, fő jellemzője a hierarchikus felépítés, ez a logikai és fizikai felépítésre egyaránt igaz.

Tananyagelem (asset): a tananyag legkisebb egysége, ez tulajdonképpen egy fájlt takar, ami tartalmazhat szöveget, képet, hangot, videót stb.

Megosztható tartalomobjektum (sharable content object): egy vagy több tananyagelem, az LCMS képes önállóan kezelni.

Lecke (lesson): egy vagy több tartalomobjektum, legkönnyebben egy hagyományos tankönyv egy leckéjének vagy fejezetének képzelhetjük el. Ha több leckéből áll egy tananyagrész, akkor már modulról beszélünk.

Kurzus (course): egy vagy több modulból áll. Egy kurzus lehet egy tankönyv, egy tantárgy, vagy egy tanfolyam.

Képzési program (curriculum): több, összetartozó kurzusból áll össze a képzési program, amely egy adott képesítéshez tartozó tanfolyamok összerendelését végzi.

2.6 Az e-learning előnyei és hátrányai

Az előnyöket és hátrányokat sok szemszögből vizsgálhatjuk. Nézhetjük a kurzust szervező, az oktató, valamint a hallgató szemszögéből.

A kurzust szervező számára az e-learning előnye a képzések rugalmassága és átláthatósága, a képzés költéshatékonysága, valamint az oktatási anyag könnyen bevihető és frissíthető. Hátrányként említhető a bonyolult szervezési feladatok, valamint a kezdeti magas költség és az ezzel járó befektetési kockázat.

Az oktatók számára előny a tanítási folyamat nyomon követhetősége, az oktató csak a tanításra koncentrál, és nem elhanyagolható az intézményi támogatás sem. Hátrány az új típusú pedagógia feladat és az intézmény részéről nagyobb ellenőrzésre nyílik lehetőség.

A hallgatóknak előnyt jelent az oktatáshoz kapcsolódó járulékos költségek nagymértékű csökkenése, a tanuló saját ütemében haladhat az anyaggal a hely és idő függősége nélkül, a tananyag tárolható, és előhívható, rendszeres kommunikáció lehetséges a mentor és a hallgató társak között, az új technikai eszközök, szoftverek használatának elsajátítása. A hallgatóknál hátrány lehet az oktatás személytelenné válása, az interaktivitás csökkenése, az önálló tanulás nehézségei, az azonnali visszajelzés és értékelés, a megfelelő infrastruktúra hiánya.[8]

2.7 Az e-learning piacot meghatározó szereplők

2.7.1 A közoktatás

A közoktatás egyik legnagyobb problémája, hogy legtöbbször a hallgatók nem versenyképes ismeretekkel kerülnek ki az iskolából. Mára az oktatás fő célja a képességek, kézségek elsajátítása lett. Ezek birtokában a saját egyedi élethelyzetnek megfelelő tudást lehet megszerezni. A diák fő feladata az élethosszig tartó tanulás képességének elsajátítása. Ezt a képességet csak az írás és olvasás készsége, egy idegen nyelv ismerete és az infokommunikációs technológia együttes ismeretében birtokolhatjuk.

Néhány szempont, ami a digitális tananyag, illetve az e-learning mellett szól a közoktatásban:

A hagyományos pedagógia tartalma és eszköztára széttördelt, ez egyes tudáselemek elkülönülnek, és nem helyezhetők át. A digitális pedagógia részei kiemelhetők, áthelyezhetők, szabadon bővíthetők.

A papír alapú tananyagot a diszciplináris tudás határozza meg, míg a digitális tananyagot az interdiszciplináris tananyagépítés. Ma már nem elemi fokú tudósokra, hanem ismereteiket integrálni képes, sokoldalúan művelt dolgozókra van szükség A digitális tananyag jobban

[8] Digitális taneszközök használata a szakképzésben
http://www.sulinet.hu/ikt/17_szakmacsoport/rendeszet/rendeszet_06.html

felkészíti a hallgatót a valós életre, mivel több tudományágat, szakterületet érint és a gondolkodásfejlesztésben is hatékonyabb ez az elrendezés.

A lineáris tananyagszervezés leíró szerkezetű: a tananyag a tudást ismeretek formájában tartalmazza. Ezzel szemben, a digitális tananyag-szervezési modell logikai struktúrát követ: az információ a hálózati kapcsolódásain keresztül élő tantárgyi kapcsolatok alakíthatók ki.

2.7.2 A felnőttképzés

A felnőttképzés célja az élethosszig tartó tanulás biztosítása. Ismereteinket aktualizálni, frissíteni kell, szükség esetén át kell képeznünk magunkat, ha a munkaerőpiac számára értékessé akarunk válni. Az Európai Unió az egész életen át történő tanulást támogatja és ösztönzi. A mai világban már nem az a jellemző, hogy bekerülünk egy munkahelyre és onnan megyünk nyugdíjba. Egy állampolgárnak akár több munkahelye is lehet élete során. A cél az, hogy jobb, jövedelmezőbb helyet találjon. Az e-learning megfelelő oktatási forma, egy felnőtt, dolgozó, családos ember számára.

A felnőttképzéssel foglalkozó intézmények száma egyre nő, de az informatikai eszközökkel való ellátottságuk még hagy némi kívánni valót maga után. Jelenleg a pedagógusképzés sincs egészen felkészülve az e-learning módszereinek oktatására. Gyakori probléma a megfelelő tanár hiánya, vagy hogy a meglévő digitális tananyag nem megfelelő egy e-learning-es oktatásra. A tanulni vágyó felnőttek nem mindegyike rendelkezik a szükséges eszközökkel.

2.7.3 Vállalati képzések

A legtöbb vállalatnál az alkalmazottak részt vesznek 4-5 napos képzésen, továbbképzésen. Ez idő alatt az adott munkavállalót vagy helyettesítik, vagy a munkájával marad el. Mind a kettő plusz költséggel jár. Nagyobb vállalatok egységei között földrajzi távolság is lehet, mégis képzéseit egy helyen végzi. Pénzt és időt takaríthatnak meg, ha az e-learning oktatási módot használják.

Az e-learning előnye a hagyományos oktatással szemben:

- a munkakieséssel járó költségek csökkenek,

- a járulékos költségek csökkenek (utazás, szállás, étkezés),
- alacsonyabb képzési költségek (kivétel a beüzemeltetés, ami nagyobb beruházást igényel),
- az oktatás helytől és időtől független,
- a tanult ismeretek azonnal hasznosíthatóak.

3. Az e-learning szabványosítása

A Web-alapú oktatási technológiák jelen állapotára alapvetően jellemző az intézményi, tartalmi, felhasználásbeli és technológiai sokszínűség. Nagy mennyiségű oktatási anyagok kerültek digitalizálásra, és sok oktatási tartalom már alapvetően webes felhasználásra jött létre. A digitális tananyagoknak az újrafelhasználhatóságát, és más rendszerekkel való elérhetőségét biztosítani kell, ami szabványosítás nélkül nem kivitelezhető. Jelenleg nem létezik hivatalos e-learning szabvány, csak ajánlások vannak.

Szabványok gyakorlati jelentősége:[9]
- Tananyagok „újrafelhasználhatóak” és környezet-függetlenek.
- Alkalmazhatóak egységes minőségi értékelési szempontok.
- Általánosan használható keretrendszerek és tananyagszerkesztők.
- Lehetségessé vált „dobozos” tananyagok készítése.
- Fejlesztési és bevezetési költségek könnyen és pontosan meghatározhatók.

Nagyon sok szabályrendszer létezik a felsorolásuk és ismertetésük egy külön dolgozat témája lehetne. Két szabványt szeretnék bemutatni ebben a fejezetben az egyik a LOM (Learning Object Metadata) a másik a SCROM (Sharable Content Object Reference Model).

3.1 Az IEEE e-learning szabványa, a LOM

LOM (Learning Object Metadata) az első globális, általános, elfogadott szabvány, amely az online képzési tartalmak leírását, metaadatokkal való kiegészítését támogatja. A szabványt 2002-ben fogadták el. A szabványt az IEEE (Institute of Electrical and Elektronics Engineers) szervezet deklarálta.

[9] Gál András, E-learning, Szakdolgozat, Debreceni Egyetem, 2007

Tanegységek (LO, Learning Object) kereséséhez, beszerzéséhez, cseréjéhez alkalmazzák. Maga a szabvány az LO méretére nem ad meghatározást, de ez a legkisebb értelmes tanegység, amelyekből egy adott tananyag összeállítható. A tanulási egységet jellemző leírást nevezzük metaadatoknak. A metatadat segítségével egy LO minden lényeges tulajdonsága leírható.

A szabvány meghatározza az LO szintaktikai és szemantikai jellemzőit. A LOM célja, hogy meghatározza azon minimális szükséges jellemzőket, amelyekkel a tanulási egységek könnyen kezelhetők, minősíthetők és visszakereshetők.

A LOM tulajdonképpen a tanegységekből és velük kapcsolatos erőforrásokból nyert adategyüttes.

Egy tanegységet leíró adatelemek különböző kategóriákba sorolhatók. A LOM kilenc kategóriát különböztet meg:
Általános: a tananyagelem általános leírására szolgál
Életciklus: az erőforrások életciklusával kapcsolatos tulajdonságok
Meta-metaadat: a metaadatról ad információt
Technikai: az erőforrások technikai jellemzői
Oktatási: oktatási és pedagógiai tulajdonságok
Tulajdonjogok: szellemi tulajdonjogok és felhasználói jogok feltételei
Kapcsolat: más tanegységekhez való kapcsolódás jellemzői
Megjegyzés: megjegyzések a szolgáltatások oktatási használatával kapcsolatban
Besorolás: a tanegységek kapcsolata egy adott másik besorolási rendszerhez

3.2 Az ADL SCORM szabványrendszere

Az Egyesült Államok Honvédelmi Minisztériuma (Department of Defense, DoD) 1997-ben hozta létre az ADL (Advanced Distributed Learning) kezdeményezést, hogy a minisztérium hatáskörében egységes stratégiát dolgozzon ki az oktatásra és információs technológiára. Célja az oktatás és képzés korszerűsítése, a kormány együttműködésének segítése az

egyetemekkel és az üzleti szektorral, valamint az e-learning szabványosítása. Az ADL kezdeményezés magas szintű elvárásokat fogalmazott meg az oktatási tartalommal szemben: újrafelhasználhatóságot, hozzáférhetőséget, tartósságot és keresztfelhasználhatóságot.

E dokumentum a Megosztható Tartalom Objektum Hivatkozási Modelljét (SCORM) fogalmazza meg, amely megfelel e magas szintű elvárásoknak.

A SCORM szabályozza a tananyagcsomag szerkezetét, a tananyagot futtató környezetet és a tananyag metaadatokat.

A SCORM két fontos funkciót lát el: tartalomszervezést és kommunikációt a tananyag és a keretrendszer között.

A tartalomszervezés lényege, hogy nagy mennyiségű dokumentáció redundanciamentes és megjelenésfüggetlen előállítás és tárolás. Az elv tömören az, hogy egy egységes adatbázisban, megjelenéstől függetlenül, és redundanciamentesen tároljuk az adatainkat, és ebből hozzuk létre az adott dokumentációt. Az így kapott dokumentum strukturált szövegként áll össze, mely médiumtól függően lehet nyomtatott, webes, vagy akár e-learning tananyag is.

A SCORM kommunikációs szerepe a következő. A tananyagot ha beimportáljuk egy keretrendszerbe, akkor az LMS felépíti a kívánt tananyagstruktúrát.

A felületre bejelentkezett tanuló megnyitja az adott tananyagot, elkezdi nézegetni a leíró részeket, majd megoldja a hozzá tartozó teszteket. Az oktatási környezet szerepe, hogy regisztrálja ezeket az eseményeket, a leíró tartalmaknál *megnézte* / *nem nyitotta meg* , a teszteknél pedig ennél bővebben: *átment* , *megbukott* , *nem fejezte be* állapotokat tartja nyilván.

A kommunikációs szerep a diákok interakcióinak a nyomon követésére szolgál.[10]

[10] Advanced Distributed Learning (Fejlett Elosztott Tanulás) Megosztható Tartalom Objektum Hivatkozási Modell (SCORM) 1.2 verzió SCORM áttekintés http://elearning.sztaki.hu/repository/18.pdf

4. Az e-learning módszertana[11]

Az e-learning tananyagnál is az írott szöveg a meghatározó, de a szöveg funkciója megváltozott a hagyományos tankönyvek szövegétől. Az e-learning tananyag nem papíron, hanem képernyőn jelenik meg. Ahhoz hogy megfelelő tananyag készüljön, ismerni kell az új módszertani és technikai lehetőségeket. Egy pedagógia és pszichológiai ismeretekkel rendelkezi oktató nem biztos, hogy jó tananyagot tud készíteni. Ezt külön oktatni kell, illetve kellene, mert Magyarországon még kevés intézmény foglalkozik a pedagógusok e-learning módszertani képzésével.

4.1 A tananyag tartalmának összeállítása

Az e-learninges tananyagok tagolását kétféle módon kell elvégezni. Didaktikai és a technikai tagolásról beszélhetünk.

Didaktikai tagolás az, amikor a tanulási folyamat optimalizálásához a tananyag logikai összefüggésrendszerét vesszük alapul. Technikai tagolásnál a tananyag strukturálása a lényeg, ami egyrészt a programozást, másrészt a tananyag érdekessé tételét jelenti.
Didaktikai tagolásánál a tananyag e-learning megfelelője a kurzus. Egy kurzus létezhet önállóan, vagy kapcsolódhat másik kurzushoz.

A kurzus felépítése a következő:

- keretrendszer;
- modulok.

A keretrendszer bevezető oldalból, a kurzus végén található összefoglalóból és tesztekből állhat. A modulok bevezető oldalból, leckékből, a modul végén található összegző oldalból, modulzáró tesztekből állnak. A leckék a modul elemi egységei. A modulok tetszőleges mélységig ágyazhatók egymásba. A lecke tartalmazza az elsajátítandó téma leírását, a tanítási-tanulási tevékenységeket.

[11] Lengyel Zsuzsanna Mária: E-learning: tanulás a világhálón keresztül című szakdolgozata 6. fejezete alapján

Technikai tagolásnál a leckéket felépítő tananyagelemek segítségével tartalmi-fogalmi tagolás adható a tananyagnak, illetve vizuális megkülönböztetés is lehetővé válik. Ezért az elemeket a tananyag szövegén belül meg kell különböztetni. Ilyen elemek:

- a szöveges elemek,
- az adattábla-elemek,
- a képi elemek,
- az akusztikai elemek, és
- párbeszédes elemek.

4.2 A tananyag szövegének megformálása

E-learning tananyag szövege lehet, problémaorientált vagy rendszerorientált.

A rendszerorientált szöveg kész tudásrendszert közvetít. Logikusan felépített, a tanulónak csupán meg kell tanulnia.

A problémaorientált szöveg esetében azt kell feltételezni, hogy a tanulás a probléma megoldását jelenti. Fő cél a problémamegoldó képesség fejlesztése problémahelyzetek bemutatásával.

Természetesen mindkét megközelítés együttesen is alkalmazható egy adott tananyagban. Egy jól felépített szövegnek, rendszer- vagy problémaorientáltnak egyaránt a következő feltételeknek kell megfelelniük:

- figyelembe kell venni a célcsoport előfeltételeit, és
- a szöveg legyen érthető, segítse a tanulót a tanulásban.

4.3 A tananyag hipertexes tagolása, kép- és hangelemeinek kiválasztása

Egy e-learning tananyagban szerepelhet belső, és külső link. A belső link segítségével a tananyag bejárható. Eljuttathatja a hallgatót régebbi leckék fontosabb részeire, vagy akár egy lecke korábbi pontján megadott példára. A külső link kivezet a tananyagból, általa a tananyaghoz kapcsolódó információhoz juthatunk.

A képhasználat célja, hogy a tanuló képet alkothasson egy tárgyról, jelenségről. Egy képet be lehet illeszteni a lecke szövegébe, vagy külső, illetve belső link segítségével el lehet jutni hozzá.

Auditív elemek alkalmazásánál a következő lehetőségek léteznek:

- írott szöveg megszólaltatása
- hangos tananyag
- a szöveges illetve vizuális részeket kiegészítő auditív elemek
- zenei betétek, speciális hangeffektusok használata
- didaktikai funkciójú tanári közlemény.

5. HTML, CSS, JavaScript

A dolgozatom részét képező honlap alapja a HTML leíró nyelv, melybe JavaScript és CSS ágyazódik. A honlap HTML dokumentumok és beillesztett képek sokaságából áll össze. A megjelenített oldalak (címsorok, bekezdések, listák) stílusát külső stíluslap határozza meg, a menüsor könnyen kezelhetőségét és küllemét a JavaScript biztosítja. A továbbiakban a már említett nyelveket és stíluslapokat fogom bemutatni.

5.1 HTML (HyperText Markup Language)

5.1.1 A HTML dokumentum felépítése

A HTML dokumentum egy olyan szövegfájl, amely a szövegen kívül tartalmaz ún. "HTML-tag"-eket - formázóutasításokat -, valamint megjelenítendő objektumokra történő hivatkozásokat is. Ezek a HTML formázóutasítások (más szóval: parancsok, elemek) befolyásolják a dokumentum megjelenítését, kapcsolatait. Ezeket az utasításokat a böngészőprogram értelmezi és végrehajtja. Ezen okból a formázóutasítás mindig megelőzi azt a részét a dokumentumnak, amelyre vonatkozik.

A HTML utasításokat a szövegben < és > jelek közé kell zárni. Egy-egy utasítás - HTML parancs, HTML elem - hatását általában a záró utasításpárja szünteti meg, amely megegyezik a nyitó utasítással, csak a / jel vezeti be (természetesen a < és a > jelek között). Az utasítások

nagy része opcionális elemeket is tartalmazhat, melyek csak a nyitóutasításban szerepelhetnek, a záróban nem. Az opciók értékadásánál az idézőjel nem mindig kötelező, csak ajánlott. A HTML utasítás kulcsszavaiban nem különböztetjük meg kisbetűket és nagybetűket.

A HTML formátumú szövegfájlok a <HTML> utasítással kezdődnek és </HTML> utasítással végződnek.

A HTML dokumentum két fő része: *fejléc* és *törzs*. A dokumentumot a fejlécelemek vezetik be, melyek kezdetét és végét a <HEAD>…</HEAD> utasítások jelzik. A fejlécelemek között szokás a dokumentumcímet megadni, mely címet a <TITLE> és a </TITLE> utasítások, közé kell zárni. Ezt a részét a dokumentumnak általában az ablak címsorában jelenítik meg a böngészőprogramok.[12]

A <BASE HREF="protokoll://gépnév/elérési_út"> utasításban szereplő URL határozza meg a báziscímet, melyből a relatív címeket értelmezni kell. Az intelligens kiszolgálók korában nem kötelező megadni.

A <BASEFONT SIZE="szám"> utasítással jelölhető ki a dokumentumban az alapértelmezett betűméret.

Az <ISINDEX> utasítás jelzi a keresőrendszerek számára, hogy index-dokumentumról van szó.

Az <LINK> utasításban szereplő opciók jelzik a dokumentum kapcsolatait más dokumentumokkal, stíluslappal, címszalaggal, stb.
Az <META NAME="mező" CONTENT="érték"> utasítás jelezheti a keresőrendszerek számára a dokumentum-adatbázisba kerülő adatokat, pl. a dokumentum alkotóját, a létrehozó programot, rövid tartalmat, stb.

Nézzünk egy példa dokumentumot:

[12] Almási Pál: A HTML dokumentumokról http://www.szabilinux.hu/htmlhelp/p13.htm

```
<HTML>
<HEAD>
<TITLE>A dokumentum neve/címe</TITLE>
Fejléc elemek ...
<META NAME="Author" CONTENT="XY">
<BASEFONT SIZE=8>
</HEAD>
<BODY>Dokumentumtörzs
<H1>Alcím</H1>
<P>Normál bekezdés
</BODY>
</HTML>
```

5.1.2 A HTML dokumentum szövegtestének felépítése

Minden HTML formátumú szövegfájl a <BODY> és a </BODY> utasításokkal közrezárt részében tartalmazza a megjelenítendő részét. (A dokumentum-kereteket kivéve.) Ezen elemek között kell elhelyezni mindent: a szöveget, hivatkozásokat, képeket, stb.
A <BODY BACKGROUND="fájlnév.kit" BGCOLOR="színkód" TEXT="színkód" LINK="színkód" VLINK="színkód" ALINK="színkód"> utasításban a dokumentumtörzsre vonatkozó fenti előírások is szerepelhetnek opcióként.

A BACKGROUND="elérési_út/fájlnév.kit" opcióval a dokumentum hátteréül szolgáló fájlt jelölhetjük ki. Háttérszínt a BGCOLOR="színkód" opcióval kiegészített utasítással definiálhatunk.

A dokumentumban a szöveg színét a TEXT="színkód" opcióval jelölhetjük ki. A LINK="színkód" opció a hivatkozások megjelenési színét határozza meg. A VLINK="színkód" pedig a már bejárt hivatkozásokat jelölő színt határozza meg.

5.1.3 A HTML dokumentum címszintjeinek felépítése

A HTML formátumú szövegfájlban definiálhatunk címeket, ill. alcímeket hat szint mélységig. A legfelső szintű címet a <H1 ALIGN="hely"> és a </H1> utasításpárral kell közrezárni. A második szintet a <H2 ALIGN="hely"> és a </H2> utasítások határolják stb.

Minden szint más-más betűformátumban jelenik meg a dokumentumban, a böngészőprogram beállításától függően. A címek igazítását szabályozza az ALIGN opció, melynek lehetséges értékei: *left, center, right*. Amennyiben túl hosszú a cím, de egy sorosnak kell maradnia, akkor a NOWRAP opció megakadályozza a cím betördelését több sorba.

A címek csak a szemlélő számára keltik a tagoltság érzetét, a valóságban nem tagolják fizikailag szakaszokra a dokumentumot. Ezt a tagolást a <DIV CLASS="osztály" ALIGN="hely">, </DIV> utasításokkal lehet meghatározni, ahol a CLASS opció sorolja a megfelelő SGML osztályba a szakaszt, az ALIGN pedig a szakasz igazítási formátumát írja elő. Az automatikus tördelést itt is megakadályozza a NOWRAP opció, ez esetben a szakasz tördelését a <P> vagy a <BR> utasításokkal lehet szabályozni.

5.1.4 Bekezdések a HTML dokumentumban

A HTML formátum bekezdésekre tagolódik. A HTML fájlban a bekezdések *kezdetét* a <P> utasítás jelzi. A bekezdés a következő bekezdés kezdetét jelző utasításig tart. Tehát nincs záró utasításpárja. A legtöbb böngészőprogram két bekezdés között egy üres sort szúr be megjelenítéskor. A bekezdés elem hordozhat magában a bekezdés stílusát meghatározó opciókat. A bekezdés igazítását a <P ALIGN="hely"> formájú utasítással szabályozhatjuk. Az automatikus tördelést a NOWRAP opció tiltja meg a böngészőprogram számára. Amennyiben tördelhetetlen szóközt igényel a szöveg, akkor az egyszerű szóköz helyett alkalmazzuk a   különleges karaktert. Amennyiben egy bekezdésen belül mindenképpen új sort szeretnénk kezdeni, akkor a <BR> utasítást kell használni. (Nincs zárópárja.)

5.1.5 Kereszthivatkozások HTML dokumentumok között

A HTML formátum lényegét az egymásra és egymás tartalmára való hivatkozások jelentik (vagyis a hypertext lehetőség). A dokumentum bármely részéhez hivatkozást (linket) helyezhetünk el, amelyet aktivizálva, a hivatkozottal összefüggésben lévő szöveghez jutunk el. A hivatkozó utasítások megjelenési formája sokféle lehet, a célobjektumtól függően.

A legegyszerűbb esetben a hivatkozás az adott fájl egy távolabbi részére mozdítja a böngészőablakot. A hivatkozás kezdetét a <A HREF="#jelző"> utasításnak a dokumentumban való elhelyezése jelzi. A hivatkozást a </A> utasítás zárja le. Ez az elempár közrezárhat szövegrészt, képet, stb. A közrezárt részt a böngészőprogram a dokumentum többi részétől eltérően jeleníti meg (pl. aláhúzással, kerettel, ...), az egérkurzorral fölé érve a mutató alakja megváltozik. Azt a részt (praktikusan: könyvjelzőt), ahová a hivatkozás mutat a <A NAME="jelző"> és a </A> utasítások kell, hogy határolják.

A legtöbb esetben a egy hivatkozás egy másik fájlra/dokumentumra mutat. A hivatkozás kezdetét ekkor a <A HREF="protokoll://elérési_út/fájlnév.kit"> utasítás jelzi, a hivatkozást ekkor is a </A> utasításelem zárja le. Mind a protokoll, mind az elérési út elhagyható, amennyiben azonos URL-en van a kiindulási dokumentum és a hivatkozott. A hivatkozott fájlnak e példában nincs külön névvel (könyvjelzővel) jelölt része. Működés szempontjából a fentebb leírtak vonatkoznak erre a hivatkozási formára is.

A legbonyolultabb esetben a hivatkozás egy másik fájl valamely pontosan meghatározott részére mutat. A hivatkozás kezdetét a <A HREF="protokoll://elérési_út/fájlnév.kit#jelző">
utasítás jelzi, és a hivatkozást szintén a </A> elem zárja le. Ebben az esetben a hivatkozott fájl kell, hogy tartalmazzon egy olyan részt (könyvjelzőt), ahová a hivatkozás mutat. Ezt a részt a <A NAME="jelző"> és a </A> utasítások határolják.

Ha az <A HREF=...>, </A> utasításpár képet fog közre, akkor a kép szegéllyel jelenik meg, amely szegély letiltható az <IMG SRC=...> utasításban elhelyezett BORDER=0 opció

alkalmazásával. A képekkel kapcsolatos egyéb hivatkozási lehetőségeket a képek elhelyezése a HTML dokumentumban alcím alatt tárgyalom.[13]

5.1.6 Karakterformátumok a HTML dokumentumban

A HTML formátumú szövegfájlban is használhatjuk a szövegszerkesztőkben megszokott karakterformátumokat. Az alábbi táblázat a formázás kezdő és záróutasítása között a mintát is tartalmazza.

Kezdő elem	Ilyen betűformátumot eredményez	Záró elem
<B>	**Félkövér betűformátumot eredményez**	</B>
<I>	*Dőltbetűs formátumot eredményez*	</I>
<U>	Aláhúzott formátumot eredményez	</U>
<S>	~~Áthúzott formátumot eredményez~~	</S>
<TT>	Fixpontos betűket eredményez	</TT>
<EM>	*Kiemeli a szöveget*	</EM>
<CITE>	*Idézetekre használható*	</CITE>
<VAR>	*Változónevet jelöl*	</VAR>
<STRONG>	Ez is egy kiemelési lehetőség	</STRONG>
<CODE>	Kódoknál használjuk	</CODE>
<SAMPLE>	Minták jelzésére használjuk	</SAMPLE>
<KBD>	Billentyűfelirat jelzése	</KBD>
<BQ>	Idézet megjelenítése	</BQ>
<BIG>	Nagyméretű betűformátumot eredményez	</BIG>
<SMALL>	Kisméretű betűformátumot eredményez	</SMALL>
_{	Alsóindexet eredményez	}
^{	Felsőindexet eredményez	}
<BLINK>	Villogó szöveget eredményez	</BLINK>
<FONT...>	A részleteket lásd lentebb	</FONT>

[13] http://redqueen.uw.hu/modules.php?name=Sections&op=viewarticle&artid=1

A <FONT FACE="név" COLOR="színkód" SIZE="szám">, </FONT> utasításpárral direkt módon előírhatók a megjelenő szöveg betűinek a jellemzői.

5.1.7 Képek elhelyezése a HTML dokumentumban

A HTML formátumú dokumentumban képeket - grafikákat - is megjeleníthetünk. Az <IMG SRC="elérési_út/fájlnév.kit"> utasítás a szöveg aktuális pozíciójába helyezi a megadott képet.

Elemezzünk egy példát:
<IMG ALIGN="hely" SRC="elérési_út/fájlnév.kit" HSPACE="vszám" VSPACE="fszám"
WIDTH="szméret" HEIGHT="mmeret" UNITS="egység" USEMAP="#jelző"
ISMAP ALT="szöveg">.

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

A HSPACE a kép melletti vízszintes térközt, a VSPACE pedig a függőleges térközt (margókat) határozza meg.

A WIDTH a szélességét, a HEIGHT pedig a magasságát adja a képnek, az UNITS által meghatározott egységben (*pixel* vagy *en*).

Az ALT azt a szöveget adja meg, amelyet nem grafikus böngészők használata esetén meg fog jelenni a kép helyett.

5.1.8 Listák a HTML dokumentumban

A *számozott listát* az <OL> és az </OL> utasítások közé kell zárni, a *számozatlan listát* pedig az <UL>, </UL> utasításpár közé. Mindkét listában használhatjuk a listafejlécet, melyet az <LH> utasítás vezet be - az </LH> pedig zár. Mindkét listatípusban a listák sorai az <LI> utasítással kezdődnek és <LI> a lezáró párjuk.

A fentiekből következik, hogy a listasorok egy <LI> utasítástól a következő <LI> utasításig, ill. a lista záróeleméig tartanak.

Számozott lista esetében a kezdő sorszám közvetlenül megadható az <OL SEQNUM="szám"> formájú kezdőutasítással. Másik lehetőség, hogy egy előzőleg definiált lista számozása folytatható az <OL CONTINUE> kezdőutasítás használatával. Egyébként az <OL> utasítás 1-től kezdi a listatagok számozását.

A számozatlan listák kezdőeleme is hordozhat formázóinformációkat. Az <UL SRC="fájlnév.kit"> formájú utasítás például a listasort megelőző bajuszként a megadott képfájlt használja. Az <UL DINGBAT="karakter"> a megadott bajuszkaraktert alkalmazza. Az <UL WRAP="irány"> pedig többoszlopos listák esetén az igazítás formáját határozza meg. (A WRAP opció a *horiz* és a *vert* értékeket veheti fel.)

A számozatlan listák speciális - külön HTML utasításokkal létrehozható - fajtái a könyvtárlista és a menülista. A könyvtárlista típus a <DIR> utasítással kezdődik és a </DIR> utasításra végződik. A menülista pedig <MENU> és a </MENU> utasításokkal határolt. Ezek a listaformák a normál számozatlan listáktól mindössze annyiban különböznek, hogy a könyvtárlista tagjai 20 karakteresnél, a menülista tagjai, pedig egy sorosnál nem lehetnek hosszabbak és nincs "bajuszuk".

A leíró listát a <DL> és a </DL> utasítások közé kell zárni. A lista fejléc megadása azonos az többi listatípusnál látottal. A listák egyes alkotóelemeinek kezdetét a <DT> utasítás jelzi, az ehhez tartozó leírás kezdetét pedig a <DD> utasítás határozza meg. Nincs egyik utasításnak sem záró párja, ezért a lista tag a <DT> elemtől a <DD>-ig, a hozzá tartozó leírás, pedig a <DD> elemtől a következő <DT>-ig tart.

5.1.9 Táblázatok

Ez az utasításcsoport képes a legváltozatosabb szöveg-, és képmegjelenítési formák előállítására. A <TABLE> és a </TABLE> utasítások közé zárt részt tekintjük egy táblázatnak.

A táblázatnak a címét a <CAPTION> és a </CAPTION> utasítások között kell megadni. (Az így megadott cím nem a táblázatban, hanem előtte fog megjelenni.) A cím <CAPTION ALIGN="hely"> formájú megadással igazítható.

A táblázat minden sora a <TR> utasítással kezdődik és a következő <TR>-ig, ill. a táblázat végéig tart. Egy sor tartalmazhat oszlopfejléceket és adatokat. Az oszlopfejléceket a <TH> utasítás vezeti be és választja el egymástól. A táblázat adatcellái, pedig a <TR>-rel megkezdett sorban egy <TD> utasítással kezdődnek és minden cella a következő <TD>-ig - ill. a következő sor elejét jelző elemig - tart, ahol értelemszerűen új cella kezdődik. Az oszlopfejléceknek és az adatcelláknak csak a kezdőutasítása használatos - habár van lezáró utasításuk is (</TH>, </TD>) -, mert a záróutasításuk elhagyható.

A táblázat nyitóutasítása tartalmazhat a teljes táblázatra vonatkozó beállításokat:
<TABLE BORDER="szám" ALIGN="hely" COLSPEC="oszlopjellemzők" UNITS="egység" NOWRAP CELLPADDING="pszám" CELLSPACING="kszám" BGCOLOR="színkód">

Ahol a BORDER opció a rácsozat szélességét határozza meg. (*0* esetén nincs rácsozat.) Az ALIGN a teljes tábla elhelyezkedését határozza meg (*left, right, center* lehet). A COLSPEC egy oszlop igazítását és szélességét adja meg. Egy oszlopra vonatkozóan egy betű és szám egybeírva (pl.: *L12 C24 R10*), melytől a következő oszlop értékeit egy köz választja el. Az UNITS a számokhoz tartozó mértékegységet jelöli ki (*en, relative* - oszlopszélességhez -, *pixel*). A NOWRAP opció a cellák szövegének tördelését tiltja le. Végül a BGCOLOR a táblázat háttérszínét határozza meg.

A táblázat oszlopfejlécei nem csak a legfelső oszlopban szerepelhetnek, hanem a táblázatban bárhol (pl. sorok címeként is).

Mind az oszlopfejlécekben, mind az adatcellákban használhatók a következő formázásra való opciók:
COLSPAN="szám":

Egyesít több egymással szomszédos cellát - vízszintesen.
ROWSPAN="szám":
Egyesít több egymás alatti cellát - függőlegesen.
ALIGN="hely":
Igazítja a cellák tartalmát - vízszintesen. Lehetséges értékei: *left, center, right, justify, decimal*
VALIGN="hely":
Igazítja a cellák tartalmát - függőlegesen. Lehetséges értékei: *top, middle, bottom, baseline*

5.1.10 Keretek

Egyetlen böngészőablakban több HTML dokumentum is megjeleníthető a <FRAMESET> és a </FRAMESET> utasításpár, valamint a szorosan kapcsolódó <FRAME> utasítás együttes használatával.

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

A fenti módon definiált területekre a <FRAME SRC="objektum"> utasítás tölti be a megadott objektumot, mely objektum lehet egy teljes HTML fájl, annak egy maghatározott része, ill. egy kép. Az így kitöltendő keretek viselkedését szabályozza az utasítás <FRAME NAME="név" SRC="objektum" SCROLLING="érték" MARGINWIDTH="szám" MARGINHEIGHT="szám"> alakú megadása.

Az adott keretnek nevet ad a NAME opció, a szkrollozást letilthatja SCROLLING="no" kiegészítés (ezenkívül a *yes* és az *auto* értékeket veheti fel a SCROLLING opció), a

MARGINWIDTH és a MARGINHEIGHT pedig a kereten belüli margók szélességét szabályozza.

Például a fejlécben megadott <BASE TARGET="név"> utasítás a NAME="név" opcióval elnevezett keretbe irányítja a hivatkozásokat. Egyébként az <A HREF=...> utasítás is ismeri a TARGET="név" opciót. (A TARGET="_top" opcióval az egész böngészőablakot elfoglalja a hivatkozott dokumentum, tehát feloldja az ablak keretekre osztását.) Ha ezek egyike sem szerepel, akkor a hivatkozás a hivatkozó objektum keretében jelenik meg, felülírva azt.

A <FRAMESET>, <FRAMESET> utasításpárral határolt területnek meg kell előznie a <BODY> utasítással kijelölt dokumentumtörzset. Sőt a egy <NOFRAMES> utasítással kell jelezni a dokumentum azon részének kezdetét, amelyet akkor kell a böngészőnek megjelenítenie csak, ha nem ismeri a keretutasításokat. És csak ez a <NOFRAMES>-mel kezdődő rész tartalmazhatja a <BODY> és a </BODY> utasításpárt.

5.1.11 A HTML és a multimédia

A HTML formátumú szövegfájlban szerepelhetnek multimédia jellegű fájlokra történő hivatkozások is.

A <BGSOUND SRC="hangfájl.wav" LOOP="szám"> utasítás a letöltés közben lejátszandó hangeffektust határozza meg, mely a LOOP-ban megadott alkalommal ismétlődik.

A <SOUND SRC="hangfájl.wav"> utasítás hatására a böngésző megjelenít egy hivatkozást a lejátszandó hangeffektusra, amelyet aktiválva az adott hangfájlt lejátssza.

Mozgóképet tartalmazó fájl is elhelyezhető a HTML dokumentumban az <IMG DYNSRC="fájlnév.avi" LOOP="szám"> utasítással, amely az <IMG SRC=...> utasítással is kombinálható. (Ebben az esetben a mozgó képsorok lejátszása után a megadott kép jelenik meg a dokumentumban, ill. akkor is, ha a böngésző nem tudja a mozgóképet lejátszani.) Az opciók megegyeznek a hang- és képfájloknál alkalmazhatókkal.

A HTML-ben külön nem támogatott típusú fájlok (pl. Flash) beágyazhatók a dokumentumba az <EMBED SRC="fájlnév.kit" WIDTH="vszám" HEIGHT="fszám"> utasítással. Az így beágyazott objektumok saját megjelenítő programot igényelnek, amelyet a böngésző feladata meghívni.

5.2 Stíluslapok, CSS

A CSS jelentése Cascading Style Sheets, azaz egymásba ágyazott stíluslapok. A HTML oldalaink megjelenését befolyásoló egyszerű nyelvről van szó, mely segítségével meghatározhatjuk, hogy hogyan (és hol) jelenjenek meg az egyes HTML elemek (paragrafusok, címsorok, stb.) Befolyásolhatjuk a színüket, méretüket, elhelyezkedésüket, margóikat, stb. Az egymásba ágyazhatóság (kaszkádolás) arra utal, hogy több stíluslapot, meghatározást is megadhatunk egyszerre, illetve egy stílus lehet több elemre is érvényes, amit egy másik stílussal felüldefiniálhatunk. A stílusok öröklődnek az oldal hierarchiája szerint. Ha például a gyökér elemre definiálunk egy stílust, akkor az többnyire az oldal összes elemére érvényes (a tulajdonságok örökölhetőségétől függően).[14]

5.2.1 A stíluslapok kialakulásának története

A HTML nyelvet fejlesztői a tartalom egyszerű közlését megoldó leírónyelvnek tervezték. A nyelv megjelenése egyre több ember számára tette lehetővé, hogy publikálhasson a weben, és az internet terjedése is egyre több és több új felhasználót hozott. Az évek során felmerült annak az igénye, hogy lehessen a nyomdai megjelenéshez hasonlóan befolyásolni a weboldalak kinézetét. Így a HTML kiegészült bizonyos formázási lehetőségekkel.

Az új formázási lehetőségek sok mindent lehetővé tettek, de a nyelv ezzel elvesztette az egyszerűségét, megnőtt a dokumentumok mérete és komplexitása. A különböző böngészők sajátosságaival is szembe kellett nézni.

Ezen segített megjelenésével a CSS szabvány, melynek segítségével szabadabban, rugalmasabban tudjuk befolyásolni HTML oldalaink megjelenését. A CSS szabvány leírása

[14] Bártházi András: CSS alapjai I-VII. http://weblabor.hu/cikkek/cssalapjai1

1996. december 17-n látott napvilágot a W3C honlapján. A szabvány azóta több kiadást ért meg, illetve 1998. május 12-n napvilágot látott a CSS 2 szabvány leírása is (a CSS 2.1 és CSS 3 kidolgozása, pedig folyamatban van).[15]

A CSS nagyban megkönnyíti egy oldal formázási lehetőségeit. Az alábbi példa egy olyan oldalt mutat be, ahol több címsor és bekezdés található.
Az eredményt a következő képen láthatjuk:

```
1. <html>
2. <head>
3.   <title>CSS példa</title>
4.   <style type="text/css"><!--
5.   h1 { font-size: 20px; color: #a00000; }
6.   p { font-size: 12px; color: #00a000; }
7.   --></style>
8. </head>
9. <body>
10. <h1>Bevezetés</h1>
11. <p>Bejelentkezés </p>
12. <h1>Iktatás </h1>
13. <p>Lekérdezés </p>
14. <h1>Archiválás </h1>
15. <p>Nyomtatás</p>
16. </body>
17. </html>
```

A *style* elemeken belül meghatározzuk, hogy a HTML állományban levő összes címsor (h1) elem és bekezdés (p) elem a fentiekben meghatározott módon jelenjen meg. Csak egyszer kell ezt megtenni, a kódunk sokkal átláthatóbb lett. A karakterek pontos méretet is meghatározhattuk, ezúttal pixelben megadva azt, de további mértékegységek is rendelkezésünkre állnak. Amennyiben valamiért utólag módosítanunk kell a megjelenést, egyből csak egy helyen kell ezt megtennünk, és mindenhol meg is változik. Az egyszerűbb

[15] Meyer, Eric A.: CSS zsebkönyv : webtartalmak formázása stíluslapokkal, Kiskapu, 2006

szerkeszthetőségen és karbantarthatóságon kívül az így elkészített dokumentum sokkal kisebb méretű is lesz, gyorsabban letöltődik, kisebb adatforgalmat generál, de a megjelenített tartalom és forma ugyanaz marad. Ha egy külső állományban helyezzük el a stílusdefiníciókat, akkor több dokumentum megjelenését is befolyásolni tudjuk egyszerre.

5.2.2 Beágyazott stíluslap

Ezt láthattuk a fenti példában. A stílusleírást a HTML oldal fejlécében kell elhelyezni, *style* elembe ágyazva. A fenti példában HTML megjegyzésbe is tettük (<!-- ... -->), ezzel azt érjük el, hogy az olyan böngészők, amik nem ismerik a stíluslapokat, ne jelenítsék meg a stílus definíciónkat. Az így meghatározott stílus az egész dokumentumra érvényes lesz.

5.2.3 Külső stíluslap

A maximális rugalmasság elérése érdekében ma szinte minden honlapon alkalmaznak külső stíluslapo(ka)t. Előnyük, hogy anélkül módosíthatjuk a honlap számos oldalának megjelenését, hogy az egyes dokumentumok forráskódjába bele kellene nyúlni. Így ha meggondoljuk magunkat pl. az aktív linkek színével kapcsolatban, csak egy helyen kell azt kijavítanunk és ez minden dokumentumra érvényes lesz.[16]

A külső stíluslapokra az oldal fejlécében tudunk hivatkozni, egy link elem segítségével:

```
1. <head>
2. <link rel="stylesheet" href="kulso.css" type="text/css">
3. </head>
```

A stíluslapot tartalmazó állományban (a példában kulso.css a neve) csak a stílusaink definícióját kell elhelyeznünk, pl:

```
1. h1 { font-size: 20px; color: #a00000; }
2. p { font-size: 12px; color: #00a000; }
```

A beágyazott stíluslapokhoz hasonlóan az így meghatározott stílus az egész dokumentumra érvényes.

[16] Síkos László: Stíluslapok a weben: CSS kézikönyv, BBS-INFO Könyk. és Inform. Kft, 2005

5.2.4 Elemhez rendelt stílus

Bár kevés alkalommal van rá szükség, de akkor jól jön, hogy az egyes HTML elemekhez helyben is tudunk stílust meghatározni. Ez nagyon hasonlít ahhoz, mintha font elemekkel határoznánk meg az adott elem megjelenését, bár nagyságrendekkel több formázási lehetőségünk van. Egy elem stílusának a meghatározásához egy style attribútumot kell hozzá felvennünk:

1. <h1 style="font-size: 20px; color: #a00000;">Bevezetés</h1>

Az így definiált stílus csak az adott elemre, illetve az adott elemen belül lesz érvényes.

5.2.5 Importált stíluslap

További lehetőségünk, hogy egy stílus meghatározáson belül egy másik, külső stílus meghatározásra hivatkozzunk (erre értelem szerint a beágyazott és a külső stíluslap meghatározásoknál van módunk). Ez a következőképpen történhet:

1. <style type="text/css"><!--
2. @import url(http://www.honlapunk.hu/stilusok/masik.css);
3. --></style>

A cím megadásakor hasonlóképpen járhatunk el, mint a linkeknél általában, azaz akár *url(masik.css)* formában is írhatjuk, amennyiben ugyanabban a könyvtárban található, mint a HTML állományunk. A külső stílus-hivatkozásnak meg kell előznie minden más definíciót, amennyiben már szerepel előtte más, figyelmen kívül lesz hagyva.

5.2.6 Stílusok formátuma

Egy önálló stílus definíciós állomány, vagy egy beágyazott stílus több meghatározást tartalmazhat. Egy-egy meghatározás két részből áll, egy kiválasztó és egy tulajdonság részből. A kiválasztó rész azt határozza meg, hogy mely HTML elemekre vonatkozzon a definíció, míg a tulajdonság rész a megjelenést befolyásolja. (Amikor elemhez rendelünk stílust, akkor csak a tulajdonság részt kell meghatároznunk.)

Egy stílusdefiníció felépítése:

1. kiválasztó { tulajdonság }
2. kiválasztó { tulajdonság }
3. kiválasztó { tulajdonság }

Az egyes definícióknak, de még a kiválasztónak és a tulajdonságnak sem szükséges új sorban lenniük, gyakorlatilag szabadon ránk van bízva, hogy a fenti tartalmat milyen elrendezésben valósítjuk meg.

Az előző példákban láthattuk, hogy egy-egy kiválasztóhoz több tulajdonságot is megadhatunk, ekkor a tulajdonságokat pontosvesszővel elválasztva kell felsorolnunk. Több kiválasztót is meghatározhatunk, a kiválasztókat vesszővel kell elválasztani:

1. kiválasztó, kiválasztó { tulajdonság; tulajdonság; }

5.2.7 Kiválasztók

A kiválasztók nagyon rugalmasan használhatóak, egyszerű és mégis összetett módon tudjuk meghatározni, hogy mely elemekre vonatkozzanak a tulajdonságaink.

5.2.8 Elem kiválasztás

A legegyszerűbb kiválasztási lehetőséget nyújtja a számunkra, egy HTML elemet határozhatunk meg vele, amihez hozzá szeretnénk rendelni a különböző tulajdonságokat. Az eddigiekben ilyenre láthattunk példát:

1. h1, h2, h3 { color: #a00000; }

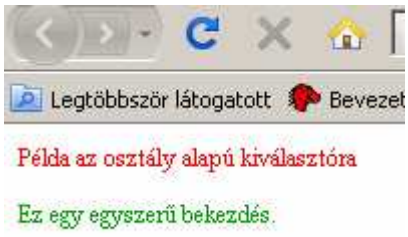
Itt az első három szintű címsornak határoztuk meg a színbeli megjelenését.

5.2.9 Osztály alapú kiválasztás

A HTML elemeket lehetőségünk van osztályokba sorolni. Ekkor az adott osztályba sorolni kívánt elemhez egy *class* tulajdonságot rendelünk hozzá, amelynek az értéke az osztálynevünk lesz. A stíluslapban ehhez az osztályhoz határozhatunk meg megjelenést. Az egyes kiválasztókat keverhetjük is.
Nézzünk egy példát:

```
<html>
<head>  <title>CSS példa</title>
<style type="text/css"><!--
p { font-size: 12px; color: #00a000; }
.fontos { color: #ff0000; }
 --></style>
</head>
<body>
<p class="fontos"> Példa az osztály alapú kiválasztóra </p>
<p>Ez egy egyszerű bekezdés.</p>
</body>
</html>
```

Ennek eredménye:



5.2.10 Azonosító alapú kiválasztás

Az azonosító alapú kiválasztás nagyon hasonló az osztály alapúhoz, egy lényeges különbség van: csak és kizárólag egy elemet tudunk egy bizonyos azonosítóval megjelölni egy HTML dokumentumon belül.

A stíluslapban a kettős keresztet használjuk az azonosításhoz, a HTML forrásban, pedig az *id* tulajdonságot. Bármi legyen is a dokumentumnyelv, egy ID attribútumnak egyértelműen azonosítania kell elemét. A kiválasztókat itt is keverhetjük (bár mivel egy azonosító csak egy konkrét elemre vonatkozhat, ezért nincs rá szükség).

Az alábbi azonosító szelektor például az összes első szintű címsorra érvényes, ahol az azonosító attribútum értéke fejezet1:

```
H1#fejezet1 {text-align: center}
```

5.2.11 Helyzetérzékeny kiválasztás

A stíluslapban lehetőségünk van a HTML fában betöltött helye alapján meghatározni egy elemről, hogy hogyan nézzen ki. Kevésbé bonyolultan megfogalmazva: a helyzetérzékeny kiválasztásnál a fában egy adott elem alatt levő elemhez rendelhetünk stílust, például megmondhatjuk, hogy a táblázatokon belül levő félkövér betűk sötétkékek legyenek, miközben a normál bekezdésekben levőknek nem fog megváltozni a színe. Ilyenkor először azt az elemet kell leírnunk, amin belül, majd egy szóközt követően azt, amire szeretnénk megadni a stílust. Több szintet is felölelhet a kiválasztónk, illetve keverhetjük a korábbiakban megismert kiválasztókat.

5.2.12 Szülő-gyermek kiválasztás

A szülő-gyermek kiválasztás abban különbözik a helyzetérzékeny kiválasztástól, hogy az elemeknek a fában betöltött helye szerinti közvetlen szülő-gyermek kapcsolata esetén tudjuk megadni a stílusmeghatározást.

5.3 JavaScript

A JavaScript a HTML lapok írói számára készült, célja, hogy kevés munkával interaktív funkciókkal lehessen bővíteni a HTML lapokat[4]. Ily módon nem csak változatosabbá tehetjük a megjelenő információkat, de az űrlapok (*forms*) kezelésénél a Web szervert is tehermentesíthetjük.

A JavaScript programok egyik leggyakoribb felhasználása az űrlapokhoz kötődik. A HTTP protokoll lehetővé teszi, hogy a kitöltött űrlapot elküldjük szerverre, ahol egy feldolgozó program indul (ún. CGI script) el és fogadja az információkat. Ez a program általában azzal kezdi a tevékenységét, hogy ellenőrzi a kapott paramétereket, és hiba esetén visszaküld valamit a böngésző előtt ülő felhasználónak. Ám ez felesleges hálózati terhelést jelent, a feldolgozó program is potyára indult el. Egy kis JavaScript program segíthet a dolgon, elküldés előtt de akár az egyes mezők kitöltésekor még a böngészőben ellenőrizheti, hogy megfelelő adatokat adtunk meg, figyelmeztethet, korrigálhat, ha kell.

A JavaScript felhasználható arra is, hogy a megjelenő HTML lapot vagy annak részeit dinamikusan, a felhasználó környezetének, általa megadott információknak megfelelően állítsuk elő. A programból létrehozott HTML formátumú szöveget a böngésző ugyanúgy jeleníti meg, mintha az a szerverről érkezett volna. Futás közben is változtathatjuk a lap egyes jellemzőit, mint például a színeit vagy egyes űrlap mezők tartalmát.

Segítséget kaphatunk ahhoz is, hogy lapjainkat egyedi navigációs lehetőségekkel egészítsük ki. Nyomógombokat definiálhatunk, amelyek különböző lapokat hívhatnak elő, a lapunkat részekre, keretekre (*frame*) oszthatjuk és az egyes keretek tartalmát függetlenül változtathatjuk. Különböző kiegészítő tevékenységeket végezhetünk egy lapra be- illetve kilépéskor, hivatkozások (*link*) követésekor.

JavaScript programok korlátozott mértékben befolyásolhatják a böngészőnk működését, megjelenését is. Figyelmeztető (*alert*) és megerősítést kérő (*confirm*) ablakokat jeleníthetünk meg, a fent emlegetett kereteken túl akár önálló böngésző ablakokat nyithatunk meg, sőt ezek fontosabb megjelenési paramétereit pl. mérete, vannak-e menüi, stb. szabadon megadhatjuk. Tájékoztatható szövegeket jeleníthetünk meg az állapotsorban.[17]

5.3.1 JavaScript és HTML

A JavaScript programokat a HTML formátumú dokumentumokba kell beágyazni, erre vezették be a következő szintaxist.

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

Ahogy a többi HTML elem esetében, itt sem jelent különbséget, ha kis- illetve nagybetűket használunk, azaz a <script> ... </script> ugyanúgy megtenné, viszont a programon belül már a kis- és nagybetűk eltérő jelentéssel bírnak.

[17] Kiss István: A Jáva testvérkéje: JavaScript http://www.cab.u-szeged.hu/WWW/java/kiss/javascr.html

A <script> olyan HTML tag, amelyet le kell zárni, ezért egy JavaScript-et értő böngésző el tudja különíteni a programot a lap többi részétől. Azonban ha olyan böngészőt használunk, amely ezt még nem érti, az ugyan magát a <script> és </script> tagokat figyelmen kívül hagyja, viszont a program szövegébe belegabalyodhat. Ezért célszerű a fenti példa szerint a teljes programot HTML megjegyzésbe (<!-- ... -->) zárni.

JavaScript programrészlet a HTML dokumentumban bárhol akár többször is előfordulhat. A programszöveg értelmezése, végrehajtása a <script> tag teljes betöltése után kezdődik meg. A JavaScript programban függvények definiálása és végrehajtandó utasítások sorozata keveredhet. A függvénydefiníciók kiértékelése csak a definíció "megjegyzésével" jár, a függvényeket a HTML lap más pontjain, általában a későbbiekben megismert eseménykezelőkben hívhatjuk meg. Célszerű a lapon felhasznált összes függvényt a lap fejében a <head> és </head> között definiálni.

5.3.2 Adatok, változók és kifejezések

A JavaScript programban előforduló adatok csak a következő négy különböző típusba tartozhatnak: objektum, számérték (amely egyaránt jelenthet egész vagy valós számokat), szöveg (karaktersorozat) és logikai érték (*true* és *false*).

Kifejezések: az egyes értékeket megjelenítő literálokon túl változók illetve függvényhívások is előfordulhatnak bennük. Kifejezések írásához felhasználhatjuk a C-ből megismert összes szokásos operátort, beleértve az olyan egzotikusakat is, mint az értékadó operátorok (pl. =+), vagy a feltételes kifejezést (feltétel ? érték1 : érték2) is. Az aritmetikai és összehasonlító operátorok jól tűrik a szövegeket is, szükség esetén szó nélkül konvertálnak szövegek és számértékek között ide-oda, de a *parseInt* és *parseFloat* függvényekkel explicit konverziót is előírhatunk, az *eval* függvény pedig egy szöveget, mint JavaScript kifejezést értékel ki. Szövegekre itt is értelmezett a + operátor.

Természetesen változókat is használhatunk. Nem kell az egyes változók típusát deklarálni, ugyanazon nevű változó élettartama során tetszőleges típusú értéket hordozhat. Változókat a következő két módon hozhatunk létre.

var név [= érték]

vagy

név = érték

Az első szerkezet, ha elhagyjuk az értékadást, akkor nem definiált értékű változót hoz létre. A változók lehetnek globálisak, az egész programban felhasználhatók illetve lokálisak, amelyek csak az adott függvényen belül használhatók, ott viszont elfedhetik az azonos nevű globális változót. Az első fajta deklaráció az előfordulási helyétől függően lehet globális és lokális is, a második az előfordulásától függetlenül mindig globális.

5.3.3 Utasítások

A JavaScript csak a következő alapszavakat használja az utasításokhoz:

break	continue	for
for…in	function	if…else
new	return	var
while	with	

Azonban az összes Jáva alapszót is fenntartott, programozók által nem használható szóként kezeli. Az utasítások egy C programozó számára nagy meglepetéseket nem jelentenek, legfeljebb hiányzik nekik a *switch* szerkezet. Újdonság a *for..in*, *new* illetve a *with* utasítások, ezekről majd az objektumoknál szólok.

Függvények definiálására a

function fnév (paraméterek) { ... return érték }

forma szolgál. A függvény visszatérési értékének típusát nem kell definiálni, ez is dinamikus, attól függ, hogy a *return* milyen értéket ad vissza. A függvény törzsében az egyes paraméterekre szokásos módon a nevükkel hivatkozhatunk, de a paramétereket egy tömb elemeiként is elérhetjük, pl. az

```
fnév.arguments[0]
```

az első paraméterre hivatkozik, míg az

```
fnév.arguments.length
```

a paraméterek számát adja meg, így lehetőséget kapunk változó számú paraméteres függvények írására.

A függvényhívásnál a paraméterátadás érték szerint történik, azaz a függvény ugyan módosíthatja a paraméterként megkapott értéket, de ennek a hívóra semmilyen hatása nem lehet. Aki imádja mellékhatásokkal (side effect) áttekinthetetlenné tenni a programjait, az használjon gyakorta globális változókat.

5.3.3 Objektumok

A JavaScript objektum-modellje sokak számára furcsának tűnhet, engem leginkább a LISP nyelv objektumorientált kiterjesztéseire emlékeztet. A JavaScript objektum <tulajdonság, érték> párok halmaza. Az egyes tulajdonságokhoz (property) az ismert

```
object.property
```

szintaxissal hivatkozhatunk, az értékét használhatjuk, módosíthatjuk. Egy objektum létrehozására a new utasítás szolgál, amely által lehívott függvény kitöltheti az objektum tulajdonságainak értékeit. Például:

```
function person (first, last, age)
{
  this.first_name = first;
  this.last_name  = last;
  this.age        = age;
}
Dezso = new person("Dezső", "Nagy", 23);
Dezso.age = 24;
```

utasítássorozat létrehoz egy objektumot, amely értéket rendel a *first_name*, *last_name*, *age* tulajdonságaihoz, majd életkorát megváltoztatja. A függvényen belül a this az aktuális, esetünkben az éppen létrehozott objektumra hivatkozik.

A JavaScript objektumokat asszociatív tömböknek is tekinthetjük, ahol a tulajdonság nevekhez értékek tartoznak, amelyek a nevek alapján hozzáférhetők. Ezt az analógiát erősíti az is, hogy a tulajdonságokhoz az indexelésnek megfelelő szintaxissal is hozzáférhetünk, pl. a fenti példában írhattuk volna azt is, hogy

```
Dezso["age"] = 24;
```

Ezek után nem meglepő, hogy JavaScript-ben külön tömbök nincsenek. Tömbnek olyan objektumot nevezünk, amely tulajdonságai a 0-tól kezdődő egész számok, ezen felül van egy *length* nevű tulajdonsága a tömb "hosszának" tárolására. Lássunk egy trükkösebb tömb-konstruktor függvényt és felhasználását, amely kihasználja a változó számú paraméteres függvényt:

```
function initArray()
{
  this.length = initArray.arguments.length;
  for (var i = 0; i < this.length; i++)
  this[i+1] = initArray.arguments[i];
}
mixed = new initArray( "text", 1, 23, Dezso);
```

Érdekes a példa, látható, hogy a tömb értékei nem feltétlenül azonos típusúak, sőt egy objektum tulajdonságának értéke lehet egy másik objektum is. Természetesen mutatók itt sincsenek, de a *null* "üres objektum" itt is létezik.[18]

Az egyes objektumokhoz függvényeket is rendelhetünk, így a szokásos módszereket kapjuk. Hosszas magyarázkodás helyett bővítsük a fenti person példát.

```
function sayYourName ()
{
  document.write("<B>Szia, "
            + this.last_name + " " + this.first_name
            + " vagyok.</B><P>");
}
function person (first, last, age)
```

[18] Moncur, Michael: Tanuljuk meg a JavaScript használatát 24 óra alatt : [24 egyszerű, egyórás lecke], Kiskapu, 2006

```
{
  this.first_name = first;
  this.last_name  = last;
  this.age        = age;
  this.introduce  = sayYourName;
}
Dezso = new person("Dezső", "Nagy", 23);
Dezso.introduce();
```

A *person* által létrehozott objektumoknak lesz egy *introduce* tulajdonsága, amihez egy függvény tartozik. A program vastagon szedve kiírja, hogy "Szia, Nagy Dezső vagyok."
Az objektumok összes tulajdonságán a *for...in* ciklussal lehet végighaladni, a következő függvény például előállít egy olyan HTML szöveget, amely a paraméterként megadott (*obj*) és megnevezett (*obj_name*) objektum összes tulajdonságát és a hozzá tartozó értéket felsorolja.

```
function dump_props(obj, obj_name) {
  var result = ""
  for (var i in obj) {
    result += obj_name + "." + i + " = " + obj[i] + "<BR>"
  }
  result += "<HR>"
  return result
}
```

5.3.6 Beépített objektumok

Ezek a programozási lehetőségek nem sokat érnének, ha nem lennének a JavaScript-ben előre definiált objektumok, hozzá tartozó tulajdonságaikkal és módszereikkel. Szerencsére számos ilyen objektum van itt fel sem tudom mindet sorolni , ezek jó része a HTML lap egyes elemeire hivatkozik. Megadom a HTML lapokhoz tartozó egy objektumok hierarchiáját. A hierarchiának nincs köze az öröklődéshez, csupán azt jelenti, hogy a hierarchiában fentebb álló objektum valamely tulajdonsága a lenti objektumot tartalmazza (hivatkozik rá).

- *window*: a böngésző ablaka,
    - *frames*: keretek az ablakban,
    - *location*: a megjelent lap URL-je,
    - *history*: a böngésző által bejárt lapok,
        - *document*: a konkrét HTML dokumentum,
            - *forms*: űrlapok,
                - űrlap elemek: *text field, textarea, checkbox, password, radio, select, button, submit, reset*
            - *links*: hivatkozások
            - *anchors*: dokumentum címkék

Ha egy beépített objektumnak több azonos típusú eleme van, akkor azokra vagy tömbszerűen lehet hivatkozni, vagy az elem dokumentumban megadott neve alapján (NAME= opció). Bár minden egyes objektumhoz sok hasznos tulajdonság, módszer tartozik, aminek részletes ismertetésére nincs mód, a példák talán ízelítőt adnak a lehetőségekből.

A fenti objektumokon túl van még 2 segéd objektum, a *Date* (dátum és időkezelés), *Math* (matematikai függvények és konstansok).

A JavaScript szövegei *string* objektumok, amelyeken számos transzformációs, HTML-nek megfelelő formázó függvény van értelmezve. Például a

```
var worldString="Hello, world!"
document.write("<P>" + worldString.fontcolor("red"))
```

programrészlet a "*Hello, world!*" szöveget piros színben jeleníti meg.

5.3.7 Eseménykezelés

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.

A következő táblázat megadja a lehetséges eseménykezelők nevét és a lehívását kiváltó esemény.

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

A következő példa egyszerű kalkulátort valósít meg, ahol a két mezőbe számokat írhatunk, a közöttük lévő legördülő listában a 4 alapművelet közül választhatunk, majd az egyenlőségjellel ellátott nyomógombot megnyomva az eredmény mezőben megjelenik a számítás eredménye.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function compute(form)
{
  var operator = form.op.options[form.op.selectedIndex].text
  form.result.value = eval(form.A.value + operator + form.B.value)
}
//-->
</SCRIPT>
</HEAD>
```

Eddig tartott a dokumentum feje. A *compute* függvény belsejében a paraméterként megkapott űrlapobjektum egyes részeire a nevükkel hivatkozunk: *A* és *B* a két (bementi), *result* az

(eredmény) szövegmező, *op* pedig a műveletet kiválasztó legördülő lista. A listán belül a kiválasztott elem sorszámát a *selectedIndex* módszer adja vissza, ennek segítségével indexelve megkaphatjuk a műveleti jelet, mint szöveget. Az *eval* pedig szöveget vár és megpróbálja kifejezésként kiértékelni.

```
<BODY>
<FORM NAME="calculator">
<INPUT TYPE="text" NAME="A" SIZE=8>
<SELECT NAME=op>
<OPTION>+ <OPTION>- <OPTION>* <OPTION>/
</SELECT>
<INPUT TYPE="text" NAME="B" SIZE=8>
<INPUT TYPE="button" VALUE=" = " onClick="compute(document.calculator)">
<INPUT TYPE="text" NAME="result" SIZE=8>
</FORM>
</BODY>
</HTML>
```

Az egyenlőségjel feliratú (*VALUE=" = "*) nyomógombhoz rendeltünk egy eseménykezelőt, amely akkor aktiválódik, ha rákattintottunk (*onClick*), ekkor meghívja a *compute* függvényt, átadva neki az aktuális űrlapot.

5.3.8 Állapotsor kezelés

Gyakran használható fortély, hogy a böngésző állapotsorában különböző kiegészítő üzenetek jelennek meg. Első példaként megmutatom, hogyan lehet az állapotsorba valami magyarázó szöveget kiírni akkor, ha az egér egy hivatkozás fölött áll (normálisan a Netscape az URL-t mutatja meg.) Legegyszerűbben a következő módon járhatunk el:

```
<A HREF="URL" onMouseOver="window.status = 'magyarázó szöveg';
return true">kapcsolat</A>
```

A fenti HREF tag a *"kapcsolat"* szó fölé állva az egérrel meghívódik az *onMouseOver* eseménykezelő, amely egyszerűen az állapotsorba írja a *"magyarázó szöveg"*-et, majd igaz logikai értékkel visszatér.

A megoldás szépséghibája, hogy ha elmozdulunk az egérrel, az állapotsorban ott marad a kiírt szöveg mindaddig, amíg mást nem írunk mi vagy a böngésző oda. Sajnos olyan eseménykezelő nincs, amely akkor indulna be, ha elmozdulunk az egérrel, de az alábbi két egyszerű függvénnyel adott időzítés után letörölhetjük az állapotsort.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
function stat(txt)
{
  window.status = txt;
  setTimeout("erase()",1500);
}
function erase()
{
  window.status="";
}
// -->
</SCRIPT>
```

A *stat* függvény először felülírja az állapotsort, majd a *setTimeout* hívással egy JavaScript programrészlet itt az *erase* függvény meghívását időzíti 1500 msec-kel későbbre. A fenti függvények segítségével a *HREF* tagba már elegendő a *stat* függvényt lehívni:

```
<A HREF="URL" onMouseOver="stat('magyarázó szöveg');
return true">kapcsolat</A>
```

Néhol az állapotsort "fényreklámként" használják, az állapotsorban valamilyen szöveget görgetnek jobbról balra.

6. A weboldal bemutatása

A weblap alapvető feladata, hogy a dolgozóknak segítséget nyújtson a munkájuk során használt iktatóprogram biztos kezelésében, illetve a program oktatása során jól használható digitális tananyag legyen.

6.1. A weboldal bemutatása az e-learning oldaláról

A weboldal elkészítésénél igyekeztem az e-learninggel szemben támasztott követelményeknek megfelelni.

A weblap CD-re kiírva a korai e-learning formának, a CBT (Computer Based Training) feltételeinek tesz eleget, míg az internetre vagy intranetre történő elhelyezéskor a WBT (Web Based Training), a hálózaton keresztül végbemenő tanulás valósítható meg.

A tananyaggal a dolgozók oktatása történhet szinkron és aszinkron módon. A szinkron képzési formánál, mint tankönyv használható. Az aszinkron képzésnél a tananyag hálózaton keresztül bármikor elérhető.

Az oldal bármilyen keretrendszerben elhelyezhető, akár LOM vagy SCROM szabványrendszere szerint és a mai kornak teljesen megfelelő digitális tananyaggá fejleszthető.

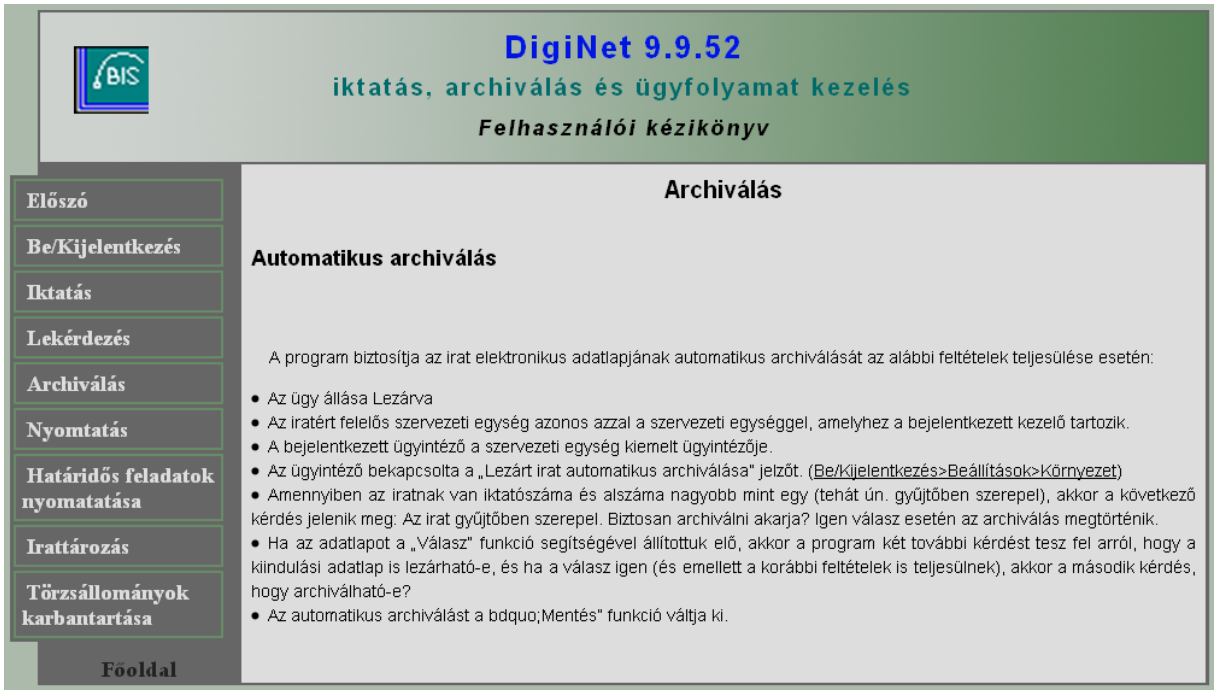
A tananyag didaktikai tagolását nézve egy önálló kurzusnak felel meg. A szöveg összeállításánál a rendszerorientált szövegformázást használtam, mivel a tanulónak csak meg kell tanulnia a szöveget és ezáltal elsajátíthatja a program használatát.

Egy további módszertani szempont, a hypertexes tagolás szerint nézve az oldal csak belső linkeket tartalmaz. Külső link használatát nem tartottam szükségesnek, mivel ez egy speciálisan szűk rétegnek, a Pénztár dolgozóinak szóló oktatóanyag. Későbbi fejlesztési terveim közzé tartozik a weboldalt videó elemekkel kiegészíteni. Ott, ahol lehetőség van rá a kézikönyv által ismertetett folyamatokat videóként megjeleníteni.

6.2. A weboldal bemutatása az informatika oldaláról

Az oldalt viszonylag gyorsan kellett elkészítenem ezért a HTML nyelvet választottam egyszerűsége és könnyű használhatósága miatt.

Az oldal szerkezetének alapját egy táblázatba beépített fejléc és bal oldali menü képezik (2. ábra).



2. ábra
Az oldal felépítése

A fejléc nagyon fontos minden weboldalon. Ez tartalmazza az oldal azonosításához szükséges adatokat és az oldal célját, feladatát, bizonyos esetekben felhasználói kört is kijelölhet. A fejlécből ki kell derüljön, hogy az oldal releváns-e számunkra. Ha egy adott intézmény honlapját látjuk, akkor elérhetőséget (telefonszám, cím, levelezési cím stb.), logót, aktuális információkat, nyelvváltási lehetőséget is fel lehet tüntetni a fejlécben.

A bal oldali menü megszületése a szimpátiának köszönhető, számomra átláthatóbb, mint a vízszintes menü, ez a kedvenc oldalszerkezetem. A menüt egy JavaScript vezérli, míg stílusát és elhelyezkedését egy belső CSS írja le:

```
<style type="text/css">
<!--
#link1
	{
		position: absolute;
		top: 230px;
		left: -150px;
	}
#link2
	{
		position: absolute;
		top: 145px;
		left: -150px;
	}
...
#link9
	{
		position: absolute;
		top: 85px;
		left: -150px;
	}
.menulink
	{
		color: #bbbbbb;
		text-decoration: none;
	}
.menulink2
	{
		color: #222222;
		text-decoration: none;
		font-size:18px;
		font-weight: bold;
```

```
                font-family: Book Antiqua;
        }
.menutable
        {
                height: 23px;
                width: 545px;
                bgcolor: #666666;
                border-width:2px;
                border-color:#668b66;
                border-style:solid;
        }
//-->
</style>
```

A JavaScript *nyit()* és *csuk()* függvényei gondoskodnak a felugró ablakok helyes megjelenéséről, valamint a menü és a felugró ablakok pozíciójáról. Nézzünk mindkettőre példát:

```
else
  if(menu1==9)
        {
                menu3=false;
                document.getElementById('link9').style.left="173px";
        }

else
  if(menu2==4)
        {
                document.getElementById('link4').style.left="-150px";
        }
```

Az *if...else* többirányú feltételes elágazásban dől el, hogy mikor melyik ablak nyílik meg és melyik záródik be. A JavaScript megvizsgálja a feltételt, ha értéke *false*, akkor vizsgálja a következőt, ha annak az értéke is *false*, akkor vizsgálja a következőt, egészen addig, míg egy igaz (*true*) feltételt nem talál. Ha megtalálja, akkor végrehajtja a hozzá tartozó utasítást, ha nem, akkor kerül sor az *else* ágra.

A tartalom iframe-ben jelenik meg, mely a táblázat egyik cellájában foglal helyet:

```
<td align='center' rowspan='2'>
<iframe name='main' width='800' height='430' src='kezdooldal.html' scrolling='auto'
frameborder='0'></iframe>
</td>
```

Ez azért praktikus, mert a tartalom az oldaltól függetlenül gördíthető, ami biztonságérzetet, átláthatóságot biztosít a felhasználónak. Az alapszerkezet, a tartalomjegyzék mindig szem előtt marad, bármikor könnyedén válthatók az oldalak között.

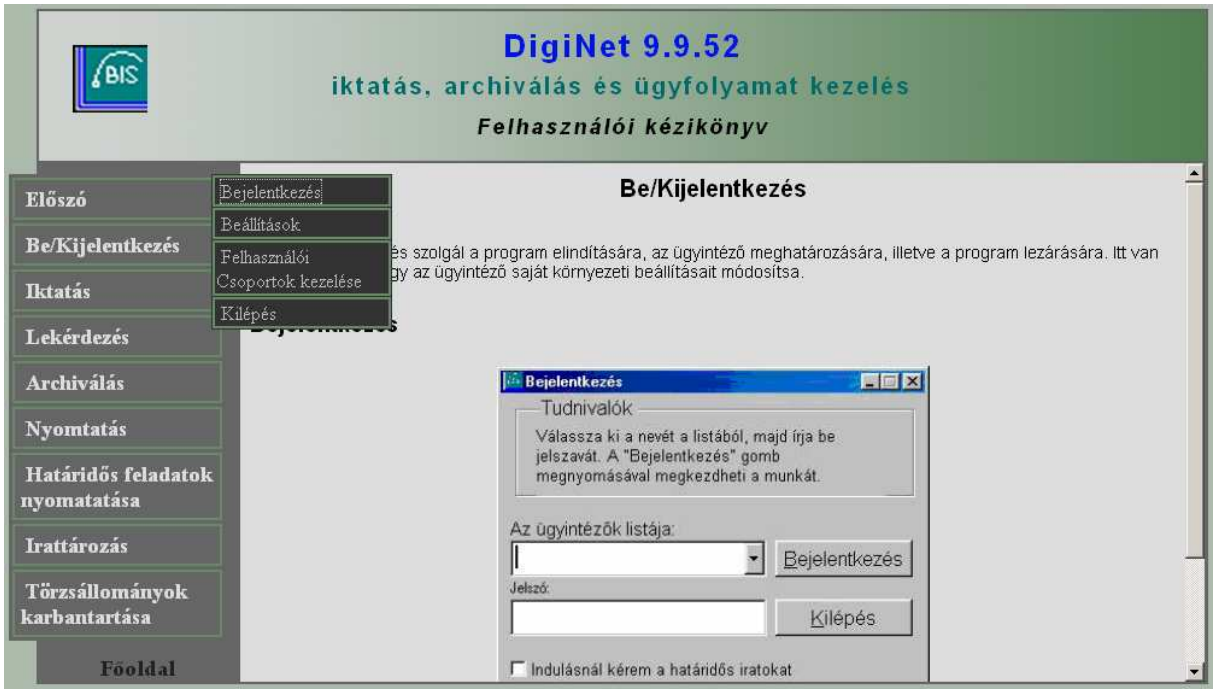
Egy adott linkre ráállva az *onMouseOver*, *onMouseOut* utasítások teremtik meg a kapcsolatot a JavaScript függvényekkel és hívják meg a megfelelő utasítást:

```
<div id="link5" onmouseover="nyit(5)" onmouseout="nyit(-5)" onmousedown="nyit(5)">
<table style="width: 150px;" border="0" bgcolor="#333333" cellspacing="1"
cellpadding="1">
<tbody>
<tr>
<td class="menutable"> <a href="autoarch.html" class="menulink"
target='main'>Automatikus archiválás</a></td>
</tr>
</tbody>
</table>
</div>
```

A színválasztást az egyszerűség, a jól olvashatóságra való törekvés vezérelte.

6.3. Az oldal használata

Az oldal az index.html fájl futtatásával indul. A képernyő baloldali részén található menüpontokban a felhasználói kézikönyv fejezetei érhetőek el. Az egér mutatóját, ha rávisszük valamelyik menüpontra, akkor a felhasználói kézikönyv fejezetéhez tartozó alcímek jelennek meg, melyekre rákattintva olvashatjuk az alcímhez tartozó leírást (3. ábra). Az alcímeket tartalmazó almenü mindaddig látható, amíg az egérmutatót vagy a fejezet címén vagy a megnyíló almenün tartjuk. A fejezetcímeket tartalmazó főmenü mindig látható, ez a könnyebb használatot szolgálja.



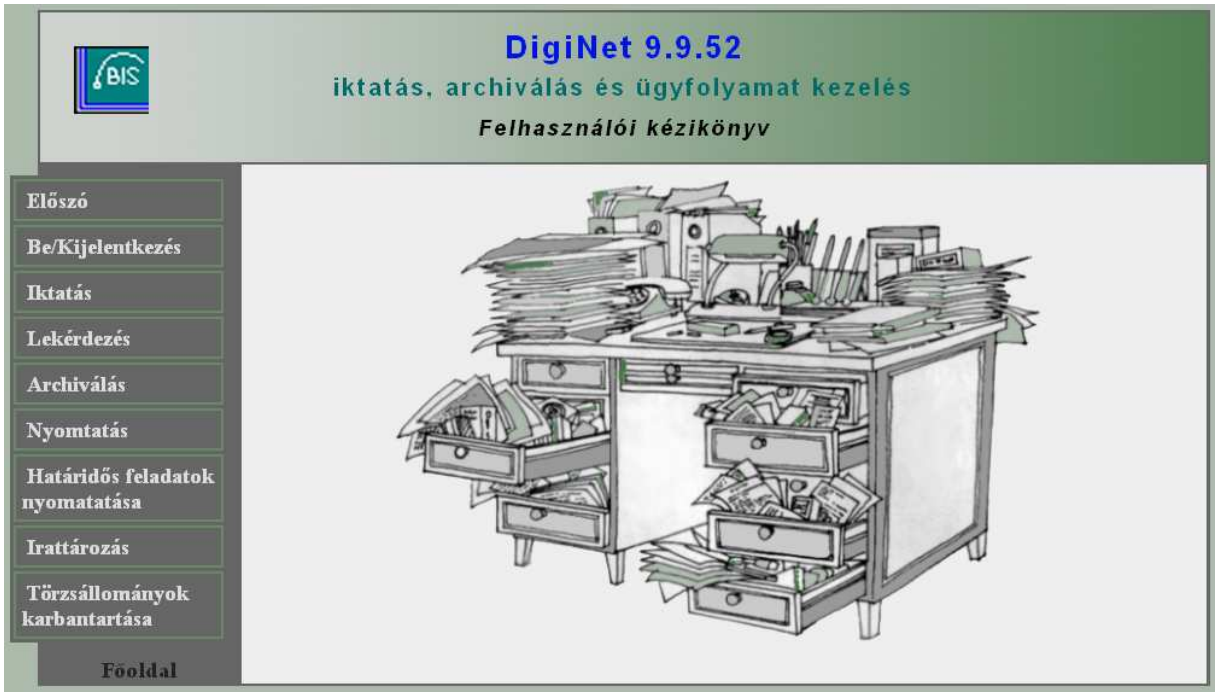
3. ábra
Almenü bemutatása

A kezdőoldalra bármikor visszajuthatunk, ha a menü alatt található „Főoldal” linkre kattintunk.

6.4. Tesztelés, üzembe helyezés

Egy weblap fejlesztésénél fontos lépés a tesztelés, ezért megkértem néhány kollegámat, hogy segítsenek tesztelni az oldalt. Munkahelyem elhelyeztem a helyi intranetes honlapunkon egy linket, mellyel el lehet érni az oldalt.

Először egy üzembe helyezési tesztet hajtottam végre. Üzembe helyezéskor azt vizsgáltam, hogy amikor a böngészőben meghívtam az oldalt, akkor rendben betöltődött-e a kezdő oldala. Amennyiben ez sikeres, akkor az alábbi kép jelent meg a böngészőben (4.ábra).



4. ábra
Kezdőoldal

Az üzembe helyezési teszt után egy funkcionális tesztet hajtottam végre. Ebben az esetben azt ellenőrizzük, hogy az oldal helyesen működik-e. Mindig a kiválasztott menüponthoz tartozó tartalom jelenik-e meg.

A weblap helyes működéséhez nincs szükség különleges hardver, illetve szoftver igénynek sem kliens, sem pedig szerver oldalon. Az oldal Google Chrome 0.4+ és Mozilla Firefox 2+ böngészőkre és 1024*768 képernyőfelbontásra optimalizált.

A tesztelés és üzembe helyezés eredményes volt, a kollégáim visszajelzései pedig pozitívak.

7. Összefoglalás

Bízom abban, hogy dolgozatom célkitűzését -digitális tananyag készítése- sikerült teljesítenem. Igyekeztem egy a gyakorlati életben számomra és munkatársaim számra is hasznosítható segédeszközt készíteni.

Korábban már írtam, hogy a Hajdú-Bihar Megyei Egészségbiztosítási Pénztárnál dolgozom, mint szakmai rendszergazda. A munkaköröm nagy részét a Pénztárnál használt DigiNet elnevezésű iktató, irat- és ügyfolyamat-kezelő rendszer rendszergazdai feladatainak az ellátása teszi ki, amibe beletartozik az ügyintézők oktatása is. A programhoz tartozó papír alapú felhasználói utasítás helyett elképzeltem egy mindenki számára elérhető, költséghatékonyabb, gazdaságosabb módszert, mely segítségül szolgálhat a dolgozóknak. Ezt valósítottam meg szakdolgozatomban, kihasználva az e-learnig nyújtotta lehetőségeket. Fontos szempont volt számomra, hogy a tananyag mindenki számára elérhető legyen valamilyen formában. Az elkészített weboldal használható helyi számítógépekre telepítve vagy számítógépes hálózaton keresztül (internet, intranet) bárki számára elérhető.

Az oldalt elhelyeztem munkahelyem helyi intranetes hálózatán és munkatársaim visszajelzései alapján igazolni látszik célkitűzésem, ugyanis olyan visszajelzéseket kapok, mely szerint a papír alapú oktatási anyaghoz képest sokkal gyorsabban, hatékonyabban tudják elsajátítani a munkájukhoz szükséges ismereteket. Ily módón az e-learnig hathatós segítséget jelent munkájukban.

Irodalomjegyzék

Könyvek, cikkek, tanulmányok

- Harangi László, Kelner Gitta, Csoma Gyula:Az e-learning szerepe a felnőttoktatásban és –képzésben, Magyar Pedagógia Társaság, Budapest, 2003
- Orbán Brigitta: Az e-learning nemzetközi és magyarországi tendenciái, Szakdolgozat, Budapesti Közgazdaságtudományi és Államigazgatási Egyetem, 2003
- Dr. Hutter Ottó, Dr. Magyar Gábor, Dr. Mlinarics József: E-learning 2005, Műszaki Könyvkiadó, 2005
- Gál András, E-learning, Szakdolgozat, Debreceni Egyetem, 2007
- Lengyel Zsuzsanna Mária: E-learning: tanulás a világhálón keresztül, Szakdolgozat, Debreceni Egyetem, 2007
- Kovács Ilma: Új út az oktatásban?, Budapesti Közgazdaságtudományi Egyetem és a Professzorok Háza Felsőoktatási Koordinációs Iroda, Budapest, 1997
- DeBolt, Virginia: HTML és CSS : webszerkesztés stílusosan, Kiskapu, 2005
- Meyer, Eric A.: CSS zsebkönyv : webtartalmak formázása stíluslapokkal, Kiskapu, 2006
- Síkos László: Stíluslapok a weben: CSS kézikönyv, BBS-INFO Könyk. és Inform. Kft, 2005
- Moncur, Michael: Tanuljuk meg a JavaScript használatát 24 óra alatt : [24 egyszerű, egyórás lecke], Kiskapu, 2006
- DigiNet 9.9.52 iktatás, archiválás és ügyfolyamat kezelés, Felhasználói kézikönyv, BIS-Inform Kft., Budapest, 1991-2008

Internetes adatgyűjtés

- E-learnig siker vagy bukás?
  http://www.fn.hu/archivum/20030617/e_learning_siker_vagy_bukas/
- Egyéni tanulmányi környezet (PLE)
  http://www.crescendo.hu/node/5
- Fattyú utód volna az e-learning 2.0?
  http://www.crescendo.hu/node/13
- Nincs e-learning pedagógia nélkül
  http://www.crescendo.hu/node/76

- Mengyelejev és az e-learning - az oktatási molekulák modelljéről
  http://www.crescendo.hu/2006/03/22/mengyelejev-es-az-e-learning
- Az e-learning kettő pont nullás forradalma
  http://www.crescendo.hu/az-e-learning-ketto-pont-nullas-forradalma
- Tanuló-központú szemlélet az e-learningben
  http://www.crescendo.hu/tanulo-kozpontu-szemlelet-az-e-learningben
- Hazai e-learning fejlődéstörténet
  http://www.crescendo.hu/2007/06/19/hazai-e-learning-fejlodestortenet
- A vállalati képzési modellekről
  http://www.crescendo.hu/2007/06/28/a-vallalati-kepzesi-modellekrol
- A hagyományos e-learning szemléletről
  http://www.crescendo.hu/2007/11/12/a-hagyomanyos-e-learning-szemleletrol
- Integratív e-learning
  http://www.crescendo.hu/2007/11/19/integrativ-e-learning
- Az e-learning
  http://ip.gallup.hu/elearning/index.htm
- e-learning
  http://www.enc.hu/1enciklopedia/fogalmi/ped/elern.htm
- Az e-learning magyarországi helyzete
  http://www.hrportal.hu/hr/az-elearning-magyarorszagi-helyzete-20080918.html
- Komenczi Bertalan: Didaktika elektromagna? Az e-learning virtuális valóságai
  http://www.oki.hu/oldal.php?tipus=cikk&kod=2004-11-ta-Komenczi-Didaktika#top
- LCMS
  http://www.kontis.hu/uvod_soucasti_lcms.asp?menu=elearning&submenu=soucasti&subsubmenu=lcms
- Dr. Magyar Bálint: Frontális pedagógia helyett „konstruktív” pedagógia
  http://www.okm.gov.hu/letolt/kozokt/digitalis_pedagogia2.ppt
- Advanced Distributed Learning (Fejlett Elosztott Tanulás) Megosztható Tartalom Objektum Hivatkozási Modell (SCORM) 1.2 verzió SCORM áttekintés
  http://elearning.sztaki.hu/repository/18.pdf
- Digitális taneszközök használata a szakképzésben
  http://www.sulinet.hu/ikt/17_szakmacsoport/rendeszet/rendeszet_06.html

- Kiss István: A Jáva testvérkéje: JavaScript
  http://www.cab.u-szeged.hu/WWW/java/kiss/javascr.html
- Bártházi András: CSS alapjai I-VII.
  http://weblabor.hu/cikkek/cssalapjai1
- Almási Pál: A HTML dokumentumokról
  http://www.szabilinux.hu/htmlhelp/p13.htm