Debreceni Egyetem
Informatikai Kar

A WEBES TARTALOM ÉS A FORMA ÖSSZEFÜGGÉSEI

Témavezető:
Dr. Bujdosó Gyöngyi
Egyetemi adjunktus

Készítette:
Szabó László
Informatikus-könyvtáros hallgató

Debrecen
2011

Tartalomjegyzék:

Bevezetés:

A szakdolgozatom a dinamikusan fejlődő webes környezet, elterjedt fájl típusainak sajátosságait hivatott vizsgálni. Összevetésre kerülnek a hagyományos dokumentumok felépítési szabályaival: elrendezés, szerkesztés, színhasználat, grafikus elemek, tipográfia, célok. Mindez kiegészül az elektronikus tartalmak létrehozásához szükséges programozási nyelvek ismertetésével.

Az informatika és az információtudomány rohamos fejlődése lehetővé tette, hogy rögzített tartalmaink a legkülönfélébb formában váljanak elérhetővé. Ha megfigyeljük, webes dokumentumaink, feltöltött, s megosztott tudásunk ugyanazt a célt szolgálja, amelyet egykoron a papírra vetett forrásaink közvetítettek: az informálódást, az eredmények megosztását, az ember tájékoztatását. Nincs ez másképp az interneten, és kisebb hálózatokon fellelhető rögzített adatok esetében sem. Egy sokkal szélesebb tárházát kapjuk a lehetőségeknek, mikor is a web fejlesztés útjára lépünk. Ezek közül néhányat említünk a következőkben.

Interaktív navigációt alkothatunk, mellyel könnyedén eligazíthatjuk elkészített termékünk felhasználóit. Hivatkozások rendszerén keresztül sokkal nagyobb fokú visszakereshetőséget alakíthatunk ki. Ezt a rendszert, ha a megfelelő rálátással és kooperációval sikerül kiépítenünk, olyan további szerzők tartalmára mutathatunk, amely releváns kiegészítések tömkelegét tárhatja elénk. Bár ezek az opciók, már egy könyv esetében is fennálltak, egy fontos szempont megváltozott. A kulcs fogalom, ami megfrissítette az ember, és rögzített tudása közti kapcsolatot: a sebesség kihasználtsága, a gyors válaszidő. Valamint a számítógép adta lehetőségeket tekintve, nem beszéltünk még ezer olyan dologról, amely megkönnyíti mindennapjainkat: fájlok megosztása, programozott modulok beépítése, kisegítő lehetőségek.

Ha megvizsgálunk egy papír alapú adathordozót, például egy könyvet, egy újságot, rengeteg olyan szabályosságot, logikai tagolást figyelhetünk meg, amelyet a mai elektronikus dokumentumokra is alkalmaznak. Ezek alapján felállíthatunk különféle követelményeket.

Egy jól megépített, online használható dokumentum kritériumai:

- A használni kívánt nyelvek ismerete, a támogatottság kihasználása.
- A kód megfelelő, hibamentes felépítése, szabvány kompatibilitás.

- A felmerülő problémák mihamarabbi megoldása.
- A megfelelő tagoltság és a velejáró átláthatóság.
- A tipográfia megfelelő fokú alkalmazása.
- A dokumentum céljának beteljesítése.
- Tesztelés, használat, üzemeltetés.

Ezen hét alapelv figyelembe vételével közelebb kerülhetünk a fő célkitűzésünkhöz: egy olyan webes alkalmazást hozhatunk létre, melynek felhasználói elégedettek, és elkötelezettek lesznek velünk, és termékünkkel szemben. Ezek a munka, s hosszas fejlődés, és a szakmában publikált kiadványok áttanulmányozása után kialakult javaslataim, észrevételeim, melyek alapján dolgozom. Fontos, talán az előző felsorolásomba bele is illeszthető tényező még a projektre ráfordított, és ráfordítandó idő is, mellyel a lehető leg optimálisabban kell gazdálkodnunk: nem szabad elsietnünk munkánkat, hiszen egy veszteséges alkalmazás eltörölheti eredményeinket.

Dolgozatom első részében egy általános összegzést olvashatunk a manapság kialakult webes tendenciákról; a felépítést tekintve: az alapozó ismeretektől egészen a specifikus, weblaptípus orientált kérdésekig kívánom feldolgozni a felgyülemlett tudást, rávilágítva olyan hibákra, amelyeket gyakran elkövetünk.

A dolgozat második fele pedig egy általam elkészített, a JAVA programozási nyelvhez kapcsolódó weboldal bemutatását jelenti, amelyben érvényesülnek eddigi tapasztalataim. A webalkalmazásom dokumentációját észrevételekkel, és megjegyzésekkel kívánom kiegészíteni, bemutatom az általam készített algoritmusokat, az általam választott script nyelven készített függvényeket.

Célok:

A dolgozatom célja, hogy segítséget nyújtson a következő generáció web fejlesztőinek, informatikus-könyvtárosainak. Habár a számítógépes lehetőségek exponenciálisan nőnek, és egy-egy webprogramozásról szóló munka hamar elévülhet, mégis úgy találtam, hogy dolgozatom hosszú távon hasznos lehet: nem csupán tömör programkódot, és programozási szabályokat, hanem általánosan elfogadott alapelveket, az évek során felgyülemlett tudásanyag kumulációját szeretném leírni. Mintát kívánok nyújtani, amely elgondolkodtató rálátásokat adhat a témában érdeklődőknek; olyan módszereket kívánok összegezni, amelyek elősegítik a rögzített információ átlátását, értelmezhetőségét.

Szeretnék egy általános képet adni a jelenlegi technikai fejlettségről. Dolgozatom témája a gyors technikai fejlődésnek köszönhetően a későbbiekben talán modernizálásra, aktualizálásra szorul, de tartalma akár egy retrospektív analízis alapjául is szolgálhat. A dolgozatom olyan jelenleg használt technikai részleteket is említ, amelyek az újabb szabványokban változtatva, vagy egyáltalán nem lesznek elérhetők. Jelen esetben betekintést nyerhetünk friss, és gyakran használt fejlesztési módszerekbe, megvizsgálva háttér információikat.

A dokumentum jó célt szolgálhat az Egyetem könyvtáros hallgatóságának számára is. Ajánlani tudom web programozó szakirányt kezdő tanulók számára, hiszen mind szintaktikáját, és szemantikáját érintve kívánom vizsgálni azokat a módszertanokat, amelyek szükségesek a webes tartalmak reprezentálásához. Néhány olyan eredményt kívánok feltárni, amelyet a szak végeztével összesítettem, és elindítottak afelé, hogy megfelelő informatikai szakember váljon belőlem.

Az általam fejlesztett alkalmazás, és a hozzá tartozó digitális erőforrások bemutatják a szakon elért eredményeimet.

1. A webes dokumentum

1.1. – A jelölőnyelvek és az általános felépítés

Mit is takar a weblap szó?

A weblap egy olyan elektronikus úton előállított, specifikus programozási nyelven megvalósított, multimédiás elemekkel, és hivatkozásokkal ellátott szellemi termék, amely a világhálón történő információtárolás és adatelérés egy rendelkezésünkre álló eszközét, szellemi termékét jelenti.

Mit jelent a web design kifejezés? Mik a főbb szempontok?

Ahhoz, hogy elkészítsünk egy jól működő, a felhasználók által rendszeresen igénybevett web alkalmazást, technikai és kreatív nyersanyagok kollaborációjára kell törekednünk, melyeknek nagy része még mindig felfedezésre vár. A képesség, hogy együttműködjünk, olyan felfedezésekhez vezethet, melyek hirtelen az iparág szabványaivá válhatnak. Mert interaktív, újdonságként a programozóknak, és kreatív szakmabelieknek egyaránt lehetősége nyílik arra, hogy új szemléleteket, és jövőképet állítsanak fel.

Célunk hogy az általunk tervezett alkalmazás könnyen kezelhető legyen, és hogy minél nagyobb közönséghez jusson el. Számos szempontont kell figyelembe vennünk ahhoz, hogy felhasználóink pozitív visszajelzéseket küldjenek felénk.

Dr. Sikos László, saját web portálján remekül összefoglalja teendőinket; rámutat arra, hogy amikor a web fejlesztésre gondolunk, számos probléma elkerülheti a figyelmünket:

„Gyakori tévhit, hogy a weblapfejlesztés csak a dizájnról és a megjelenésről szól. Valójában a weblapok fejlett technológiák használatával jelentős mértékben javíthatók a biztos funkcionalitás, az eltérő rendszereken való, kiszámítható működés, a szoftveresen kiolvasható jelentéstartalom, a keresőképekre optimalizált dokumentumok, az akadálymentesített tartalom stb. elérése érdekében. A webes szabványok optimális dokumentum-méretet, gyors letöltést és egyszerű karbantarthatóságot biztosítanak.” [1]

Ismerjük meg először a technikai hátteret.

Ismert programozási nyelvek, melyekkel előállíthatók a dokumentumok:

A web programozás tulajdonképp az IBM által kifejlesztett **GML**-nek (*Generalized Markup Language – Általánosított / Kiterjesztett jelölő nyelv*) köszönhetően indult el útján. A **GML** mintáján az általános szabvány a **SGML** (*Standard Generalized Markup Language – Általánosan kiterjesztett jelölő nyelv*) lett, amelynek makróiból kialakult a ma ismert, és használt web technológiák legtöbbje. A fontosabb, ismertebb nyelvek, alkalmazások:

HTML: *Hypertext Markup Language – Hipertext jelölő nyelv*. Az SGML egyik legelterjedtebb alkalmazása, mely mára az internet egyik alapvető szabványává vált.

Dave Raggett definíciója:

„Egy különleges szöveges dokumentumtípus, amit a web böngészők használnak szövegek és grafikák megjelenítésére. A szöveg olyan jelölőcímkéket tartalmaz, mint például a bekezdések kezdetét jelölő és azok végét jelölő . A HTML dokumentumokat gyakran hívják "weblapoknak" is. A böngészők a weblapokat webszerverektől kérik le, melyek az internetnek köszönhetően gyakorlatilag bárhol lehetnek a Földön.” [2]

Specifikusabban, a HTML a weblap építés alapköve: jelölő nyelv, amelynek elemei, tartalma dinamikusan módosíthatók, több másik programozási nyelvvel is összekapcsolhatók. Elterjedtségét remekül jellemzi, hogy a mai webalkalmazások nagy százaléka ezen a nyelven íródott, és hogy a script nyelvek is konkrét HTML kimenetet állíthatnak elő. Jelenleg az 5. szabványa készül, amely jelentős módosításokat fog bevezetni az előző verzióhoz képest.

XHTML: *Extensible Hypertext Markup Language – Kiterjeszthető hipertext jelölő nyelv*. Alapja az **XML** – *Extensible Markup Language – Kiterjeszthető jelölő nyelv*. A HTML kiszorította, így nem ez vált a legelterjedtebb szabvánnyá. A HTML-t az XML nyelv szintaktikájának megfelelően fogalmazza meg, így némileg különbözik tőle. (Case Sensitive – avagy állapot érzékeny kidolgozás: a kis- és nagybetűk használata megkülönböztetett. Pl.: a tagok minden esetben csak kisbetűket tartalmazhatnak.)

XML: *Extensible Markup Language – Kiterjeszthető jelölő nyelv*. A W3C szerinti hivatalos megfogalmazás (fordításom):

A bővíthető hipertext nyelv (**XML**) egy egyszerű szöveg alapú formátum, amely strukturált információkat ír le: dokumentumokat, adatokat, beállításokat, könyv információkat, tranzakciókat, árujegyzékeket, és sok egyebet. Egy régebbi szabványból, az **SGML**-ből (ISO 8879) származtatható, hogy kényelmesebb legyen webes használatra. [3]

Néhány XML alapú alkalmazás: Resource Description Framework, Web Ontology Language, XForms, Soap, Docbook.

JavaScript: a webes dokumentumok dinamikussá tételéhez leggyakrabban használt script nyelve, amely a JAVA programozási nyelven alapul. Szintaktikája némileg különbözik, a deklarálások nincsenek szigorúan szabályozva (például egy változó típusát nem kötelező megadnunk), valamint HTML parancsokat használó kimenetet is képes generálni.

AJAX: Asynchronous JavaScript and XML – Aszinkronizált JavaScript és XML: a szerver-oldalú programozás egy fontos állomása, amely igazán nem egy új programozási nyelv, csak a meglévő módszerek egy újfajta felhasználása. Segítségével képesek vagyunk úgy adatokat cserélni, frissíteni a honlapunkon, hogy azt nem kell újra betöltenünk.

PHP: *Hypertext Pre-processor – Hipertext elő feldolgozó*. Egy manapság igen kedvelt, nyílt forráskódú, szerver-oldalú, script nyelv, mely beágyazható a HTML-be. A hipertext jelölő nyelv és saját szintaktikájának ötvözésével generálja a megfelelő kimenetet. Működtetéséhez a szervernek támogatnia kell a PHP-t. Nagy előnye, hogy dinamikus tárolást, írást tesz lehetővé.

ASP, **ASP.NET**: *Active Server Pages – Aktív szerver-oldalak*. Egy a Microsoft által kifejlesztett szerver-oldalú web programozást igénylő script nyelv, amely manapság nagy népszerűségnek örvend. Az .asp kiterjesztésű fájlok ASP-t, míg az .aspx kiterjesztésű dokumentumok a .NET Frameworkön alapuló ASP.NET-et igénylik. Az ASP egy igen gyors, és dinamikus web portál elkészítésére ad nekünk lehetőséget.

JSP: *Java Server Pages – Java szerver-oldalak*. Egy speciális Java technológia, amely dinamikus weblapok generálására szolgál. A HTML-en, és az XML-en alapul.

Kiegészítő nyelvek, stílus nyelvek:

CSS: *Cascading Style Sheet – Fokozódó Stíluslap*. – A HTML objektumok grafikus, és elrendezési parancsait hivatott kiegészíteni. Célja, hogy elválassza a formázást a fájl logikai rendezésétől, így átláthatóbbá, és könnyebben módosíthatóvá tehető a forráskód. Jelenleg a 3. verzióján dolgoznak.

XSL, **XSD**: *Extensible Stylesheet Language – Kiterjeszthető stíluslap nyelv*. Az XML fájlok formázására létrejött szabvány, több formáját is ismerjük (**XSLT**, **XPath**, **XSL-FO**), az .xsd formátum pedig az XSL Séma leíró nyelvek egy reprezentánsa.

Az elkészített alkalmazásokat több-féle-képpen is csoportosíthatjuk. Következzék- egy felhasznált oldalfájl objektumok alapján készült felosztása:

Egyetlen fájlból álló dokumentum: a létrehozott lap kódja hibamentes, így a böngésző megnyitható oldallá fordítja azt: nem tartalmaz navigációs hivatkozásokat. Adatot tárol és jelenít meg. Tartalmazhat szöveget, képeket, multimédiás objektumokat, hivatkozásokkal, amelyek más weboldalakra, vagy fájlokra hivatkoznak. Script nyelvek általi utasításokat egyaránt magában foglalhat. Ezek valamilyen kimenetet generálnak: JavaScript, PHP, AJAX, ASP parancsok, függvények. Tekinthető a fejlesztés egy logikai alapegységének.

Több fájlból álló dokumentum: a létrehozott oldal kódja hibamentes, a böngésző megjeleníthető oldallá fordítja: tartalmazhat multimédiás elemeket (képeket, mozgóképeket, hangokat, amelyek akár linkelt tartalomra is mutathatnak), navigációs hivatkozásokat, avagy hiperlinkeket, melyek a holnap többi oldalára hivatkozna. Irányulhatnak újabb multimédiás objektumokra, vagy a weben elérhető más dokumentumra. Tartalmazhat script nyelvek által előírt utasításokat.

Egy modern webes portál az utóbbi szempontok szerint van felépítve. A szerver-oldali jelölőnyelv programozás lehetővé teszi, hogy megfelelő utasításokkal rendelkező weblap nem csak hogy új kimenetet állítson elő, hanem hogy tárolja is azt, a megfelelő adathordozón. A legtöbb web szabvány azt is lehetővé teszi, hogy egyéb nyelvekkel összekapcsolva, még részletesebb információt nyújtson a felhasználó számára. Példa erre: XHTML + SMIL (Média lehetőségek bővítése, tartalomszabályozás), XHTML + MathML + SVG (Matematikai jelölőnyelv, és a Méretezhető Vektor Grafikus nyelv implementálása.)

1.2. – A logikai tagolás

Ha tisztában vagyunk a szükséges programnyelvekkel, és az általuk kínált funkciókkal, kezdetét veheti a kidolgozás. Ez egy igen bonyolult és időigényes munkafolyamat (akár csak egy könyv esetében, gondoljunk a szerkesztési munkálatokra.), mely sok-sok részre bontható. Az első dolog, amit át kell gondolnunk az a lap *aktív felülete*, mely meghatározza majd az általunk elrendezni kívánt tartalom megjelenítését. Ügyelnünk kell a grafikai elemekre, a navigációs objektumokra, és a szövegdobozokra, akárcsak az alkalmazott színekre, és betűtípusokra. Mindezek egysége fogja majd befolyásolni a honlap kezelhetőségét. A két legfontosabb tényező:

- *Kezelhetőség*
- *Design*

Gyakori hiba, hogy valamelyik irányba dől a mérleg, lásd például:

- Dinamikus design elemek túlzott halmozása: a honlap kezelhetetlenül lassú lesz a kisebb konfigurációkon. Az ilyen objektumok, pl.: swf formátum: flash animációk, videók kombinációja rengeteg erőforrást felemészthet.
- Az esztétikus kinézet, felhasználó barát grafika hiánya: kellemetlen használati élmény, nehézkes olvasás.

Az optimum a mérleg egyensúlya: nem baj, ha esztétikus, és formabontó az oldal, amíg azt a közönsége kényelmesen tudja használni, s megtalálja, amit keres.

A felosztásnak több általános módszerét ismerjük, melyek közül néhányat már évszázadokkal ezelőtt is használtak. Ilyen felosztási lehetőséget kínál számunkra az úgynevezett golden ratio – arany metszés. Elve szerint, egy szakaszt kettéoszthatunk egy irracionális arányszám használatával. Ezt **φ**-vel jelöljük, megközelítő értéke: 1,6180339. A **φ** által jellemzett arányt megfigyelhetjük az emberi testrészek esetében is.

Ezt leegyszerűsítve a honlapunk felületét négyzetrácsosan eloszthatjuk, hogy segítse a tervezői munkánkat. *„Mikor az emberek többsége a rácsvonalazásra gondol, az építészet, és a mérnöki munka jut eszükbe. Akárhogy is a rácsvonalak alapozó segédeszközök a grafikus tervezéshez egyaránt.”* [4] Egy négyzetrácsos felosztás esetében különböző szimmetrikus, és aszimmetrikus megoldásokat dolgozhatunk ki, melyek segíthetnek meghatározni, az általunk

megosztani kívánt információ fizikai tárolását. Lényeges, hogy határozott egyensúlyt alakítsunk ki.

1. ábra: rácsozás és felosztás: a hármas osztás. [5]

Ha egy üres lapot arányosan, rácsok segítségével, felosztunk, pl. hármas osztással könnyedén kijelölhetünk rajta régiókat, amelyek a weblap navigációjának, s fő tartalmi részének csoportosítását, elrendezését biztosítják. Például ha a fenti ábrát vesszük figyelembe, a hármas osztással régiókat hozunk létre, a baloldalon körbefogott rácsokat használó rész ideális tartalom megjelenítő résznek, a jobb, vagy felső oldalt pedig használhatjuk navigációnak. A felosztást elképzeléseink szerint alakíthatjuk pl. szimmetrikus elrendezés esetén felezhetünk, hagyományos layout esetén harmadolhatunk – pl. három hasábra osztjuk a tartalmat. Érdemes minél több variációt kipróbálni. Ez az egyszerű módszer már egy tiszta képet biztosít számunkra, ami alapján elkezdhetünk dolgozni.

1.2.1 – HTML és CSS megvalósítás

A tartalom deklarálásához, pozicionáláshoz használnunk kell az adott nyelv lehetőségeit. Én most a HTML nyelvet kívánom demonstrálni: egy egyszerű oldal esetében különböző lehetőségek állnak rendelkezésünkre:

- tartalom - paragrafust definiáló parancs.
- tartalom - egy elkülönített logikai részt definiál a dokumentumban.

- tartalom - szövegelemek csoportosítására, és formázására használható parancs.

A létrehozott szövegblokkokat a fent említett tagek, HTML DTD-ben definiált attribútumaival formázhatjuk. Az alábbi parancsok használhatók mind a három tartalomkezelő tag esetében.

- align="érték" – lehetséges értékek: left, right, center. class="string" – egy osztálynév definiálása.
- dir="érték" – lehetséges értékek: rtl (right to left – jobbról balra), ltr (left to right – balról jobbra).
- id="string" – egyedi azonosító definiálása az elem számára.
- class="string" – osztálynév használatba vétele.
- style="string" – beépített CSS kód hozzáfűzése az elemhez.
- title="string" – kiegészítő információ megadására szolgál.
- xml:lang="nyelv_kódja" – nyelvi kód hozzákapcsolása a tartalomhoz, XHTML dokumentumokban.

A begépelt szöveg ezen túl is pozicionálásra szorulhat. Több módszer lehetséges a tartalom elrendezésére: a HTML nyelv lehetőségeit is használhatjuk (táblázatok, keretek, beépített keretek), vagy CSS stíluslap hozzákapcsolásával a HTML dokumentum külső parancsokkal rendezhető.

Táblázatok készítéséhez a <table></table> taget használjuk, ezen belül definiálhatunk sorokat - <tr></tr>, és oszlopokat - <td></td>.

Táblázatokhoz kapcsolódó definíciós kiegészítők:

<th></th>	fejléc definiálása
<caption></caption>	lábjegyzet készítése.
<colgroup></colgroup>	oszlop csoportok létrehozása.
<col />	attribútum érték beállítása, oszlophoz.
<thead></thead>; <tbody></tbody>;	tartalom csoportosítása a fejrészben.
<tfoot></tfoot>	és a lábrészben.

A tagoláshoz használhatjuk a **frame** - keret által nyújtott lehetőségeket is, amelyek a következő alapelven működnek. Létrehozunk egy úgynevezett keret definíciós .html fájlt, amelyet részekre oszthatunk, különböző parancsok segítségével:

```
<frameset rows=”sorok száma”, cols=”oszlopok száma”>
<frame src=”fájlnév” scrolling=”yes / no” border=”keretvastagság” padding=”párnázás
vastagsága”></frame>
…[<frame></frame>] // opcionális.
</frameset>
```

A <frameset> tag két legfontosabb attribútuma rows – sorok száma, és a cols – oszlopok száma. Ezek határozzák meg a keretünk méreteit. Értékként megadható konkrét szám, pixelekben, egy százalékos érték, amely a böngésző ablakhoz igazítja majd a keretrészek méretét, valamint a * azonosító, amely automatikusan kitölti a böngésző ablakát.

A frame tagokban meg kell jelölnünk egy forrás fájlt, a src=”fájlnév” attribútummal. Tulajdonképp önálló fájlok fogják képviselni a tagolt részeket. A keretek különböző attribútumokkal testre szabhatók: szegély vastagság, kipárnázás, gördülősáv megléte, csak hogy egy párat említsek.

Habár a keretrendszer nagy népszerűségnek örvendett, használatát manapság már nem javasolják, hiszen a HTML 5. szabványában a tag már nem támogatott: ugyanis úgy határoztak, hogy rontják a felhasználó barát, és könnyen elérhető fejlesztői környezet minőségét. Helyette használható az úgynevezett **Iframe** tag, amely egy táblázatba beépített keretfájl elérését biztosítja számunkra. Működési alapelve a hagyományos keretrendszeren alapul: szintén szükségünk van egy forrásdokumentumra, viszont maga a procedúra nem igényel külön keretdefiníciós fájlt.

```
…
<td>
<iframe src=”fájlnév”… [opcionális további attribútumok]></iframe>
</td>
…
```

Stíluslapok hozzárendelésével, pl. CSS segítségével pozícionálhatóvá válnak az általunk már ismert, és felhasznált tagok. Különböző stílus osztályokat rendelhetünk a paragrafusokhoz, a divíziókhoz, amelyek számos lehetőséget biztosítanak a formázásra. Mindehhez használhatjuk a CSS és a HTML nyelv közti kapcsot, az osztályok és egyedi azonosítók definiálását.

A poz.html kódjának részlete:

```
…
<head>
<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
</head>
…
<div class=”banner”>
… további tagoló elemek
</div>
…
```

A poz.css kódjának részlete, melyben egy osztályt definiálunk:

```
.banner {
width: 1000px;
margin: 0 auto;
padding: 0px 0px 0px 0px;
position: fixed;
}
```

Egy másik lehetőség, amiben egy ID-ra, azaz egy azonosítóra hivatkozunk.

```
#banner {
width: 1000px;
margin: 0 auto;
position: fixed;
}
```

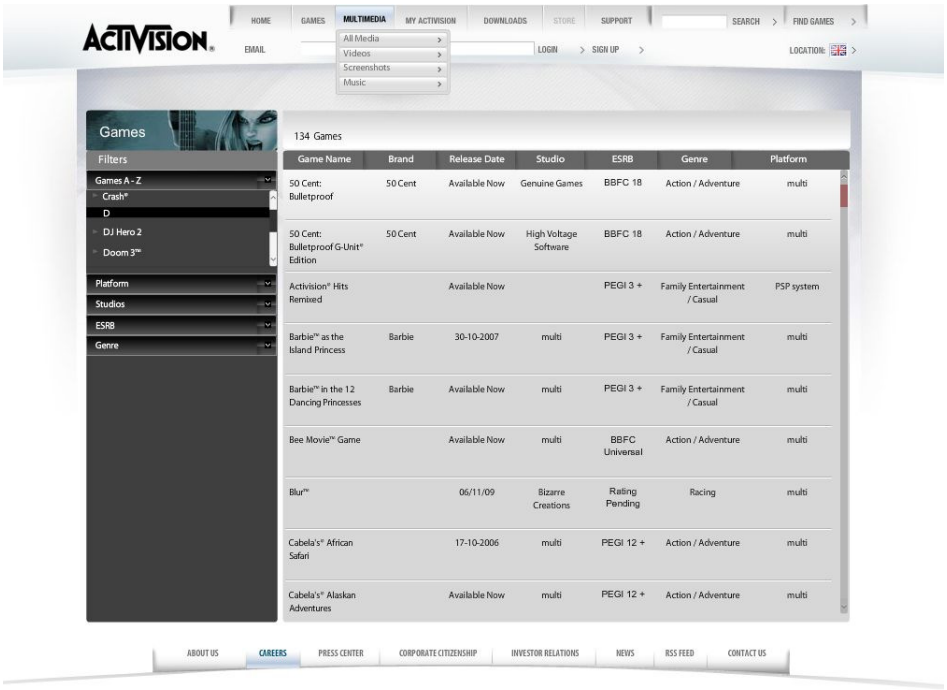
A kód alapján létrehoztunk egy rögzített pozíciójú „banner” azonosítóval rendelkező osztályt, amelyet a HTML kódban található egy taghoz kapcsoltunk. Ezáltal a tag örökli a CSS fájlban kialakított, megfelelő osztályhoz, vagy azonosítóhoz rendelt elrendezési előírásokat, pl. a böngésző ablakához viszonyítva rögzített pozíciót vesz fel, egy rögzített 1000 pixeles szélességgel.

További CSS pozíció lehetőségek: fixed (~rögzített), absolute (~a hozzárendelt tag előtti „szülő elemhez” viszonyított elrendezés), relative (~az objektum eredeti pozíciójához mért eltolás). Lehetőségünk adódik egy Z tengely menti rendezés is: tehát egy objektumot egy másik mögé, vagy elé rendezhetünk. Pl.: z-index:-1; (~a 0 és magasabb indexel rendelkező tartalmak az adott elem előtt jelennek majd meg.)

1.2.2. – Elrendezési ergonómia és hibái

A jó felosztás nem csak a kód megfelelő ismeretének függvénye. Az egyik legfontosabb szempont, amit számításba kell vennünk, az a felhasználóbarát környezet milyensége. Lehet egy honlap hibátlanul megvalósítva, ha a felhasználó nem találja meg rajta a kellő információt. Ez fakadhat a navigáció túlbonyolításából, a felelőtlen elrendezésből, vagy az arányok nem megfelelő használatából. Néhány példa:

Különféle layout (~elrendezés) kialakítási metódusokkal találkozhatunk nap, mint nap. Gyakori megoldás, hogy a navigációs menüsort a honlap bal vagy jobb oldalára igazítják, könnyű kezelhetőséget biztosítva ezzel a látogatóknak. Ezt nevezzük függőleges navigációnak. Új trendek közé sorolható vízszintes navigáció, amely a tartalmi rész fölött található. Lehetőségei – amennyiben több sort használunk, bővebbek is lehetnek, mint a megszokott jobb / bal elrendezésnek. (pl. több sornyi gomb, ha a tartalom igényli) A nagyobb, formabontó cégek honlapjain gyakran találhatunk újító megoldásokat, amelyek pl. a flash animációkhoz, és a script nyelvekhez köthetők.

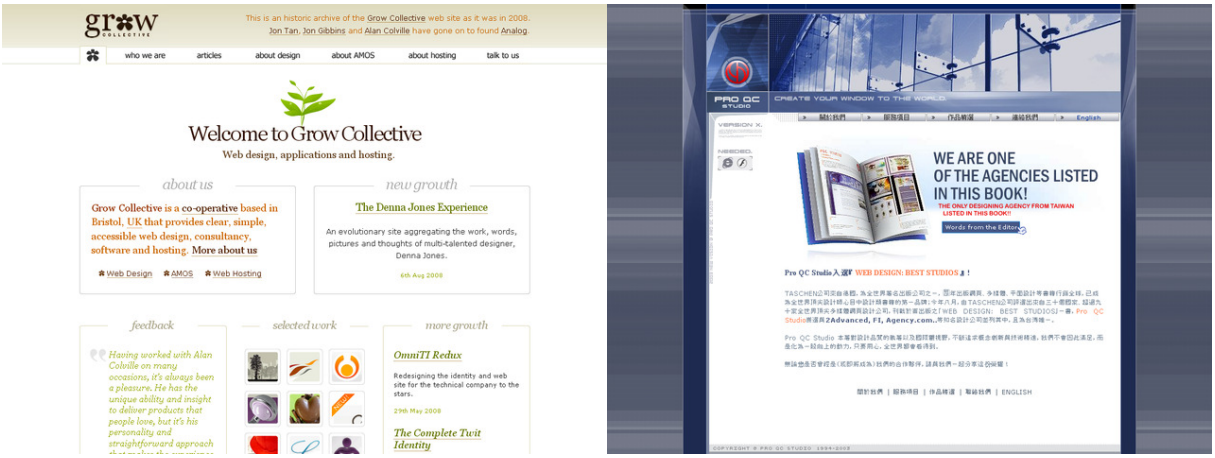


2. ábra: az Activision flash alapú oldala [6]

Kulcsfogalmak: a méret, és a láthatóság. Minden elkülönített rész rendelkezik egy rögzített, vagy egy relatív mérettel, amelyek meghatározzák, hogy mennyit látunk a honlapból, mikor azt letöltjük. Gyakori hiba, hogy az úgynevezett banner részek túl nagyok, így első betöltésre kevés információt láthatunk egy lap tartalmából. Bár ez nem tűnik túl nagy gondnak, az átlag felhasználót viszont elrettentheti munkánk használatától.

Ugyanilyen bosszantó probléma adódhat a menüsorokkal. Például ha egy hagyományos bal oldali navigációt készítünk, de a tartalmi megjelenítést egy rögzített magasságú Iframe használatával szeretnénk elkészíteni, akkor meg kell terveznünk a gombok számát, s felsorakoztatását. Sok helyen tapasztaltam, hogy az előre meghatározott sorszámú tartalmi rész miatt, a rögzített számú menüpontok közül az utolsó széle le lett vágva, mert már csak így fért az adott oldalra. Gyakori hiba, amely az előre meg nem tervezett, improvizált layout miatt jelentkezett. Ugyanilyen át nem gondolt megvalósítás a túlzott navigáció. Például az általunk meglátogatott weblap 3 hasábos felosztást használ: egy bal oldali menüsort, egy középső tartalom megjelenítőt, és egy jobb oldali menüsort. Csak akkor érdemes ezt az elrendezést használni, ha valóban szükség van rá. (Nem egy esetben találkoztam ugyanazon menüponttal mind a két oldalt.)

Az ember mindig egyfajta egyensúlyra kell, hogy törekedjen, hiszen az ilyen alkotásokat találjuk jól megvalósítottnak. A felosztott elemek alkothatnak szimmetriát és aszimmetriát egyaránt, ha az arányok megfelelőek.



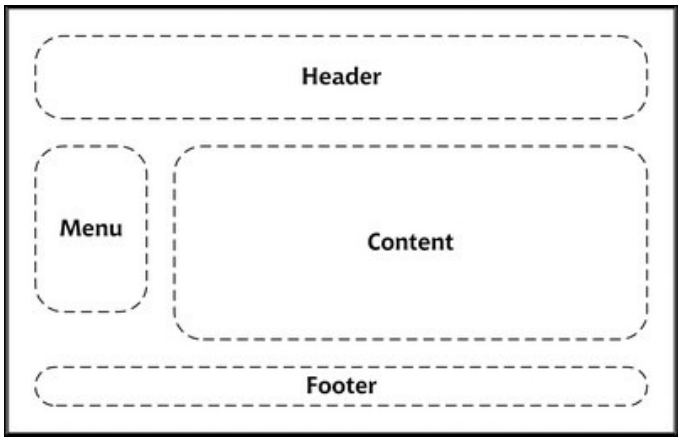
3. ábra: szimmetrikus[7] és aszimmetrikus megoldások [8]

A fentebbi két példa remek képviselője a megfelelő megoldásoknak. A szimmetriával nehezebb dolgozni, az aszimmetriával pedig könnyebb amatőr hibákat ejteni.

Fontos szempont a multimédiás elemek megfelelő elrendezése. A legfontosabb, hogy az elhelyezett grafikai elemek vezessék a szemünket a tartalom felé. Túlságosan figyelemfelkeltő képek, animációk, vagy videó objektumok nem megfelelő használata eltérítheti felhasználónkat eredeti céljától, célunktól: a megfelelő tartalom elérésétől. Nem is szólva arról, hogy sok lap túlságosan is halmozza az ilyen jellegű díszítő elemeket. Ezzel csak azt érik el, hogy az alkalmazás átláthatatlan, nehezen kezelhető, használhatatlan lesz. Így el is érkeztünk a web design kérdéskörének egy újabb fontos állomásához.

1.3. – Layout kialakítás

Kezdetnek tekintsünk át, egy hagyományos elrendezési koncepciót:



4. ábra: egyszerű web-layout séma [9]

A 3. ábra egy hagyományos felosztású weblap elgondolását mutatja be nekünk, amely egy header – avagy fejléc részből, egy menü oszlopból, egy content – tartalmi megjelenítő divízióból, és egy footer – lábléc részből áll.

A fejléc általában logók, bannerek és egy a honlaphoz tartozó címadatok megjelenítésére szolgál. Esszenciális része a weblapunknak, hiszen hiányában a felhasználó tájékozatlan marad. A menüsor segíti majd a hiperlinkek közti kommunikációt, igénybevétele esetén átirányításokat fog vezérelni. A tartalmi részről már beszéltünk: ez fogja megjeleníteni a számunkra elérni kívánt információkat. A lábléc egy opcionális tartalmi blokk, mely nem minden honlap esetében van jelen: általában üzemeltetői adatokat (a honlap készítője / készítői), vagy az azokhoz kapcsolódó elérhetőségeket biztosítja számunkra. Amint kiválasztottuk a nekünk tetsző elrendezési formát, nekikezdhetünk a megjeleníteni kívánt objektumok implementálásának.

Ha a szövegrészek, a navigációs eszközök, és egyéb megosztani kívánt tartalmak elrendezésével végeztünk, meg kell terveznünk a weblapunk arculatát. Fontos az egységesség, az összefüggő grafika használata, a színválasztás, a multimédiás objektumok minőségének, kiterjesztésének és sűrűségének meghatározása. A felhasználókat akár ilyen egyszerűnek tűnő dolgok is elterelhetik létrehozott munkánk használatától. Ilyenek például az alábbiak.

- Rossz színválasztás: befolyásolja az egész weblap hangulatát. A kitűzött célnak megfelelően kell bánnunk a palettával, különben kellemetlen órákat szerezhetünk vele az olvasóknak.
- Túlzott számú multimédiás elem: a grafikus elemek halmozása elvonja az ember figyelmét a tartalomról. Az audiovizuális objektumok akár zavaró hatással is bírhatnak, az animációs fájlok pedig ronthatják a honlap rendszerigényét.
- Tipográfiai hibák: a böngészők szöveges megjelenítési képességei nagyban függenek az adott számítógéptől. Sokan beleesnek abba a hibába, hogy weblapjukhoz olyan betűkészletet használnak, amely a másik ember számítógépén nem létezik. Ilyenkor az eredmény egy nem kívánt, igen kellemetlen főcím, vagy szövegtörzs. Ügyelni kell az olvasható betűméretre, és típusra is.

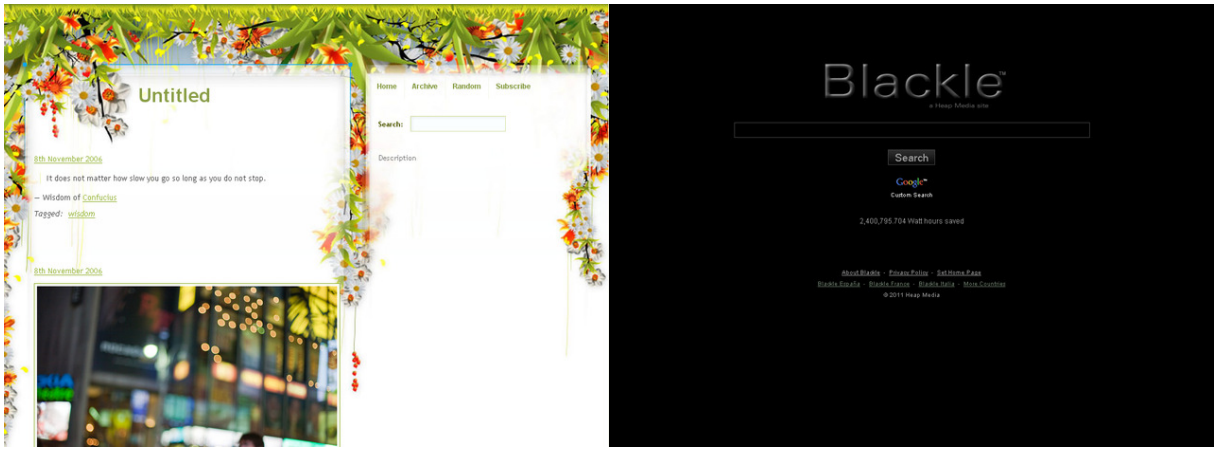
Ismerkedjünk meg a fontosabb szempontokkal!

1.3.1. – Összefüggő grafika: színek, textúrák.

Egy weblap arculatának megtervezésekor az első dolog, ami eszünkbe juthat, az az átfogó színválasztás: azaz milyen színvilággal is dolgozzunk? A dokumentumokat több szempont szerint is megkülönböztethetjük:

- Világos kompozíciók.
- Sötét, energiatakarékos árnyalatok.

Persze, ez csak egy alapvető csoportosítási forma. A világos színekkel dolgozó honlapok jobban olvashatók, közkedveltebbek, használatuk kényelmesebb a szemnek. Ebből fakadóan főleg olyan oldalak alkalmazzák ezt a designt, amelyek hosszabb olvasható információt kívánnak megjeleníteni. A sötétebb színek alkalmazása jelentősebb hibákhoz vezethet, hiszen az olvashatóság csak akkor nem romlik, ha sötét alapon valamilyen világos betűszínt alkalmazunk. Ez a fajta úgymond fekete-fehér kontraszt pedig jobban megterheli a szemünket. A sötétebb tónusokat inkább olyan lapok használják, amelyek látványos elemeket, grafikus objektumokat tárnak elénk, minimális szöveggel. (például a játékfejlesztők honlapjai, filmek hivatalos oldalai, stb.) Természetesen találkozhatunk olyan oldallal is, amely energiatakarékosság céljából döntött a fekete mellett, ilyen a Blackle nevű kereső oldal, amelyet a Google üzemeltet.



5. ábra: sötét és világos design, Bus full of Hippies[10] és a Blackle [11]

Mindig lebegjen előttünk a cél, amiért munkába kezdtünk: pl. egy könyvtári oldal elkészítéséhez semmiképp se használjuk sötét designt. Persze előfordulhat, hogy a megrendelő, avagy saját érdekeink megkövetelik, hogy egy hosszabb szöveges tartalom esetében is sötétebb textúrákat alkalmazzunk, ez esetben ügyelnünk kell a szöveg megfelelő olvashatóságára.

Egy-egy HTML alapú weboldal színvilága különbözőprogramozási eleme, és képfájlok összevonásával alakítható ki. Íme, néhány HTML, és CSS kód, amely segítségünkre lehet:

HTML nyelv:

```
<body bgcolor=”#ff00ff” background=”paper.jpg”>
```

A bgcolor egy kikevert háttérszín, míg a background attribútum egy háttérkép megadására szolgál. A bgcolor attribútum lehetséges értékei: egy hexadecimálisan megadott színkód, az RGB függvény által definiált szín, vagy egy előre deklarált szín konstans.

CSS lehetőségek:

```
Body{
background-color: #000000;
background-image: url(’bg.png’);
}
```

Gyakorlatilag ugyanazt valósítjuk meg, csak nem kell használnunk a <body> tag attribútumait. A háttérképek megválasztásánál ügyelnünk kell a méretre, és az

ismételhetőségre. Egy alapméretezetten betöltött háttérkép az x, és y tengely mentén ismétlődik. Ha a kép nagyméretű, és egy fix ablakban szeretnénk elhelyezni, akkor ezeket szükség szerint kikapcsolhatjuk pl. CSS-ben:

- background-repeat: repeat; - **ismétlődő** háttérkép.
- background-repeat: no-repeat; - a háttérkép csak egyszer jelenik meg, **nincs ismétlés**.
- backgrount-repeat: repeat-x; - a háttérkép csak az **x** tengely irányába lesz tapétázva.
- background-repeat: repeat-y; - a háttérkép csak az **y** tengely irányába lesz tapétázva.

Web programozás esetében, az általános színadásra több módszerünk is adódhat: HTML nyelv esetében vagy az előre definiált, de igen limitált számú szín konstansokat használjuk (pl. bgcolor=”red”, igényes designnal, arculattal rendelkező weblapok ezeket nem használják), vagy hexadecimálisan előállíthatjuk a megfelelő színértéket. Pl. <body bgcolor=”#ffffff”> - fehér háttérszín CSS esetében már egy sokkal összetettebb, de sokkal kényelmesebb megoldást is találhatunk. A fentebb említett módszerek mellett használhatjuk az **rgb();** függvényt: color: rgb(0%,100%,100%); vagy color: rgb(0,255,255); A color tag használható betűkre, keretekre, táblázat cellákra, egyszóval igen sok objektumhoz, míg HTML színérték megadásához a megfelelő tag DTD-ben definiált attribútumán keresztül kell hivatkoznunk.

Természetesen a CSS és a HTML összekapcsolható egy úgynevezett belső deklarációval. A HTML <head> - fejrészben megadott <style> taggal definiálhatunk osztályokat, azonosítókat, azon belül pedig CSS utasításokat, melyek később hozzákapcsolhatók azokhoz az elemekhez, amelyek létrehozásánál használjuk a **class**, vagy **id** attribútumot. Használata akkor célszerű, ha egyetlen fájlból álló dokumentumot hoztunk létre. Egy másik, működőképes, de nem javasolt módszer a style attribútum alkalmazása. Pl.: Fehér szöveg. Mindenképp érdemes egy külső CSS fájlt csatolnunk a weblapunkhoz, egyszerűen, mert sokkal átláthatóbbá teszi a forráskódunkat.

A lapon elhelyezett képek, animációs objektumok milyensége, mennyisége, és elrendezése is egy fontos kérdés. Ha egyszerűen csak a HTML adta lehetőségeket használjuk: problémákba ütközhetünk. Az <img> tag használatával meg tudunk jeleníteni egy URL-lel azonosított, vagy a számítógépünkön, tároló felületünkön megtalálható képfájlt. A képfájlokhoz különböző attribútumokat kapcsolhatunk. Íme, a még támogatást kapó lehetőségek listája:

- src: a fájl elérési útja.
- width: szélesség beállítása, felülbírálható a kép alapméretezett mérete.
- height: magasság beállítása, felülbírálható a kép alapméretezett mérete.
- longdesc: URL megadás, amely hosszabb leírást tartalmaz a képről.
- alt: a képet helyettesítő szöveges üzenet. (Ha a fájl valamilyen probléma folytán nem elérhető, ez a szöveg fog megjelenni helyette.)
- ismap: szerver oldalú térkép-tulajdonságok beállítása.
- usemap: térkép-tulajdonságok beállítása. (A későbbiekben a navigációnál lesz róla szó.)
- align: pozicionálási attribútum, amely három értéket vehet fel: left – balra, center – középre, és right – jobbra igazítás.
- hspace: úgynevezett **whitespace** – pozicionáló tér beiktatása, balról és jobbról.
- vspace: pozicionáló tér beiktatása, a kép tetejétől, és aljától.

A border, az align, a hspace és vspace használatát a w3schools oldal inkább a CSS adta lehetőségekkel javasolja megvalósítani.

CSS nyelv használatával a fentebb említett tulajdonságok listája kiegészül:

- border: keret típus, vastagság, és szín megadása. Működik HTML-ben is, de használatát már nem javasolják.
- width: auto; - automatikus képméret lekérdezés.
- height: auto; - automatikus képméret lekérdezés.
- position: a HTML align attribútumát helyettesíthetjük a fentebb már említett pozicionálási lehetőségekkel.
- margin: margó beállítások. Külön is kiszabhatók, például: margin-left: 3px;
- z-index: az objektumok z tengelyen való elhelyezése, és ezzel: egymás elé pozicionálása.

1.3.2. – Navigáció

A web designtól kicsit eltávolodva, a tartalom orientált szemlélet egyik legfontosabb aspektusáról is beszélnünk kell, ez pedig a visszakereshetőség, amit egy weboldal esetében a megfelelően beállított, átláthatóan elhelyezett, és kiépített hiperhivatkozás rendszer fog nekünk biztosítani. E linkek összességét fogjuk navigációnak nevezni.

Egy hivatkozás elkészítéséhez az anchor - horgony tagot kell ismernünk: <a>horgony. A horgony mutathat egy a dokumentumon belüli másik azonosítható horgonyra. Ez a tulajdonság arra használható, hogy hosszabb szöveges forrás esetén fejezetszerűen ugorhassunk egy a horgony által kijelölt dokumentum részre.

```
<a href="#fejezet1">Ugrás</a>
<a name="fejezet1">Szöveg</a>
```

A horgony href attribútuma jelöli magát a hivatkozást. Jelen esetben a fejezet1 azonosítóval jelölt horgonyra ugrunk (#: azonosító jele, mint a CSS nyelvben, az id attribútumokhoz társított tartalom formáló utasítások esetében.). Ha a weblapunkhoz egy másik fájlt szeretnénk társítani, legyen az egy betöltődő oldal, vagy egy letölthető objektum, vagy ha egy másik weboldal valamelyik fájlára szeretnénk mutatni, akkor a következő-képp kell eljárnunk.

```
<a href="tovabb.html">Ugrás</a>
<a href="pdf/szakdolgozat.pdf">Letöltés</a>
<a href="http://www.google.hu">Google</a>
```

További attribútum lehetőségek:

- charset: karakter kódolás beállítása. (UTF-8, Windows-1250, stb.)
- coords: link koordináták, kép térképek definiálásához használatos.
- hreflang: a linkelt dokumentum nyelvi beállításai.
- rel: a linkelt objektum és a link közti reláció, kapcsolat megadása.
- shape: szintén kép térképekhez használatos attribútum. Lehetséges értékei: rect – négyzet, circle – kör, poly – poligon, szokszög meghatározása.
- target: a hiperlinkek egyik fontos attribútuma, amely a megjeleníteni kívánt objektumnak egy megjelenítési célpontot jelöl ki. A célpont meghatározza, hogy a dokumentum: _blank – üres ablak megnyitása, _parent – az akutális ablakba, vagy keretbe tölti a tartalmat, _self – az aktuális ablak / keret tartalmának frissítése, _top - teljes ablakméretben való megjelenítés. Ezen kívül a target értéke lehet még egy **frame**, vagy **Iframe** neve: ezesetben a dokumentum abban a keretben fog megjelenni, amelyikre hivatkoztunk.

Kép térképek: ha egy képfájlon egy vagy több területet szeretnénk kijelölni, amelyek hiperhivatkozásként működnek, akkor egy képtérképet kell definiálnunk. Íme egy példa:

```
<img alt="Képfájl" src="kepek/logo.png" usemap="#ter1">
```

A kép deklarálásánál megadjuk az usemap attribútum értékét, amely a **ter1** nevű azonosító lesz. Ezután elkészítjük a térkép referenciát.

```
<map name="ter1">
<area shape="rect" coords="0,0,20,20" href="ugras.html" alt="Ez egy kijelölt terület!">
</map>
```

A térkép nyitó és záró tagjai között megadjuk az <area> nevű tagot, amely egy általunk kijelölt területet fog körbeölelni. Megfigyelhetjük az <a> tagnál használt coords, és shape attribútumokat. A shape a kijelölt terület határait fogja befolyásolni, azzal hogy egy síkbeli alakzat formáját veszi fel, míg a coords, a kezdő és végpontokat fogja megadni az x, és y tengely mentén. A tag úgy fog viselkedni, mint egy hiperhivatkozás, és amíg a kurzor a kijelölt pixel határokon belül tartózkodik, a linkelés működik.

A navigáció milyenségét speciális tagok is befolyásolják, amelyekhez különféle eseménykezelő attribútumok társulnak. Értékük egy, vagy több script utasítás, esetleg függvény meghívás lehet. Ezek a kattintást, vagy billentyű lenyomást észlelik, hasonlatosan a JAVA nyelvben használt ActionListener, KeyListener, és MouseListener interfészekhez.

- onclick: kattintás vizsgálata.
- ondblclick: dupla kattintás vizsgálata
- onmousedown: lenyomott gomb vizsgálata
- onmousemove: egérmutató mozgásának vizsgálata
- onmouseover: egérmutató pozíciójának vizsgálata: ha az objektumon rajta van.
- onmouseout: egérmutató pozíciójának vizsgálata: ha az objektumról lekerül.
- onkeydown: ha a billentyű le van nyomva: aktív.
- onkeypress: ha a billentyűt lenyomták, és felengedték: aktív.
- onkeyup: ha billentyűt felengedték: aktív.

Ezek a tulajdonságok a legtöbb elemhez használhatók. Néhány taghoz léteznek speciális attribútumok, amelyek csak adott parancsokhoz köthetők. Ilyenek az:

- onload: oldal betöltése esetén aktivizálódik. Elemek, melyekhez deklarálták: <body>, <frameset>
- onunload: oldal eltávolítása esetén aktivizálódik. Elemi: <body>, <frameset>
- onfocus: aktivizálódása egérmutatóhoz, vagy gyakori esemény bekövetkezéséhez köthető. Elemei: <a>, <area>, <label>, <input>, <select>, <textarea>, <button>.
- onblur: aktivizálódása a kiemelt szerep elvesztéséhez köthető. Elemei ugyanazok, mint az onfocusé.
- onsubmit: a formokhoz – avagy formaosztályokhoz kötődő attribútum, ha egy formaosztályt „elküldünk" akkor aktív. Elemi: <form>
- onreset: a formaosztály visszaállításánál, reseténél aktív. Elemei: <form>
- onselect: szövegkijelölésnél aktívra vált. Elemei: <input>, <textarea>
- onchange: ha egy formaosztály objektum értéke megváltozik, akkor aktívvá válik. Elemei: <input>, <select>, <textarea>.

A honlap navigációjához kapcsolódó, kissé már avuló, de továbbra is népszerű módszer a formok – formaosztályok, űrlapok használata. Az űrlapok olyan menü, és input lehetőségeket biztosítanak, amelyek alapjaivá válhatnak például a szerver-oldalú programozásnak. Ablakokat, szövegdobozokat hozhatunk létre vele, amelyek a megfelelő referencián: a forma nevén keresztül elérhetők a script nyelvek számára. Pl. a JavaScript esetében a böngésző ablaka is egy kezelhető, lekérdezhető osztály, amit window();-nak nevezünk.

Nézzünk egy általános űrlap deklarációt:

```
<form name="formao1">
<input type="text" name="szoveg">
</form>
```

A form nyitó és záró tagjai közt szerepelnek a megfelelő input mezők, vagy választó objektumok. A típusokat példákkal kívánom illusztrálni, melyekben feltűntetem a fontosabb attribútumokat is:

- <input type=”text” name=”nev” size=”20” maxlength=”35”> - egyszerű szövegmező, amelynek adhatunk egy nevet, méretet, és a maximálisan beírható karakterek számát.
- <textarea name=”szovdoboz1” rows=”10” cols=”50”> Szöveg! </textarea> - többsoros szövegdoboz, melynek neve, mérete (sorok és oszlopok száma) van. Nem input típusként, hanem önálló, a formba ágyazható objektumként érhető el.
- <input type=”password” name=”pinkod” size=”20” maxlength=”4”> - jelszó mező, melyben a karaktereket csillagok helyettesítik. Szintén méretezhető, és megszabható a maximális karakterszám.
- <input type=”button” name=”osszead” value=”+” onclick=”osszead()”> - nyomógomb, mely rendelkezhet egy névvel, egy értékkel, amely megjelenik az objektumon, valamint egy eseményfigyelő attribútummal.
- <input type=”checkbox” name=”kell” value=”yes” checked=”checked” onclick=”meghiv()”> - Jelölődoboz, mely rendelkezik egy névvel, értékkel, egy eseményfigyelővel, és egy checked – „bejelölt” nevű attribútummal. Ez utóbbi lehetőség arra szolgál, hogy alapméretezett értéket adjunk a doboznak. A HTML 5. szabványában az attribútum elhagyható, csak az értékre kell hivatkoznunk. Pl. <input type=”ceckbox” checked>.
- <input type=”radio” name=”jelol” value=”egyes” checked=”checked” onclick=”meghiv()”> - Rádiógomb, melynek jellemzője, hogy egyszerre csak egy reprezentáns lehet megjelölve. Rendelkezik egy névvel (ahhoz, hogy a gombok egységesen elérhetőek legyenek minden gombnak ugyanazt a nevet kell felvennie), értékkel, eseményfigyelővel, és a checked attribútummal (fontos, hogy azonos nevű rádiógombok esetén csak egy veheti fel a „checked” értéket.). A rádiógombok azonos név esetén egy tömb elemeiként érhetők el. Ez JavaScriptben a következő kép néz ki: document.formnev.radio.length(). Ezzel a példautasítással lekérdeztem a „formnev” nevű űrlap „radio” névvel ellátott gombjainak számát. Ezen tag checked attribútumát a HTML 5, a jelölődobozhoz hasonlóan kezeli: az attribútum neve elmarad, csak az értéket kell megadnunk.
- <input type=”reset” name=”torles” value=”clr”> - Egy értékkel, és névvel rendelkező űrlap tartalomtörlő gomb. A value értéke a gomb felületére íródik.
- <input type=”submit” name=”muvelet” value=”submit”> - Űrlap beküldő gomb, mely típusán kívül egy érték, és egy név mezővel rendelkezhet. Ezzel pedig a szerver oldalú programozás egyik állomásához, példájához érkeztünk.

Egy form tag, ha megfelelően van használva kommunikálni képes egy szerverrel, elküldése esetén. Ehhez azonban definiálnunk kell olyan értékeket, amelyek leírják, hogy milyen műveleteket szeretnénk végrehajtani.

```
<form method="post" action="mailto:example@thesis.hu">
```

A fentebbi példában deklaráltam egy post – elküldés metódust, amely egy egyszerű e-mail címmel áll kapcsolatban; egy submit gomb lenyomásával, a form elküldi az információkat a megadott URI-re. Amint láthatjuk ez a módszer remek szerver kommunikációs képességeket tár elénk. Fejlettebb web programozási nyelvekben, mint például amilyen a PHP, vagy a Microsoft .NET keretrendszerét támogató ASP.NET ezek a lehetőségek már sokkal differenciáltabbak, és gyakorlatiasabbak. Interaktív információtárolás, és lekérdezés valósítható meg velük. Ezen alapulnak a manapság nagy népszerűségnek örvendő blog oldalak, fájlcserélő portálok, internetes fórumok, vendégkönyvek, közösségi oldalak, és professzionális adatbázisok.

A választó lehetőségek közül kimaradt, egy –a textareahoz hasonlóan szintén nem az input típusokhoz tartozó lehetőség, a Select, és az ebbe ágyazható tag, az Option.

```
<select name="gordulo_menu" size="3">
<option value="egy" selected="selected">1. feladat</option>
<option value="ketto">2. feladat</option>
<option value="harom">3. feladat</option>
</select>
```

Létrehoztam egy „gordulo_menu'' névre hallgató, pontosan 3 elemszámú lekérdező menüt, amelyet feltöltöttem 3 példa lehetőséggel. Az elsőt alapméretezetten kiválasztottnak állítottam. Ezek az option sorok egy options nevű tömb értékeiként működnek. Lekérdezésük az űrlap, a gördülő menü nevén, és az options tömb nevén keresztül történnek: le szeretnénk kérdezni a második sorhoz tartozó szöveget, JavaScriptben:

```
var szov=document.formnev.gordulo_menu.options[2].text;
```

Ezzel áttekintettük a HTML adta navigációs lehetőségeink egy népes táborát, valamint lehetőségeket adtunk a JavaScript általi lekérdezések megértéséhez.

1.3.3. – WCAG – Akadálymentesítés:

Az úgynevezett „kisegítő lehetőségek” feltárása felhasználóink szempontjából egy nagyon fontos motívum lesz. Az internetet használó közösség tábora épp olyan sokrétű, mint maga a társadalmunk. Sokan rendelkeznek olyan hátrányokkal, amelyek figyelmen kívül hagyása negatív módon érintheti holnapunk felhasználói seregét. Észben kell tartanunk, hogy ha az esélyegyenlőség szellemiségében járunk el, a sikerélményünk jelentősen megnőhet.

Hogy néhány példát hozzak: sokan szenvednek olyan tartós, vagy ideiglenes állapotban, amely miatt –ha nem ügyelünk az alkalmazásunk megfelelő elkészítésére- ők egyáltalán nem, vagy csak mérsékelten tudják igénybe venni az általunk nyújtani kívánt szolgáltatásokat. A következőkben megpróbálok összeszedni néhány olyan problémahelyzetet, amelyek akadályozhatják a weboldal korlátlan használatát, valamint megoldásokat is keresek a felmerülő helyzetekre.

Mozgássérült, vagy egyéb testi fogyatékkal élő felhasználók köre: az ilyen jellegű betegségek társadalmunkat nagymértékben érintik. Beszélhetünk tartós, és ideiglenes formájáról. A valamilyen testi fogyatékossággal rendelkező használóink, vagy a különböző betegségeket, baleseteket elszenvedett használóink sok esetben csak a beviteli módok egyikét tudják használni. (billentyűzet / egér) Ezért biztosítanunk kell, hogy az alkalmazásunk megfelelően vezérelhető legyen. Hogy egy példát hozzak: a Microsoft közkedvelt Windows nevű operációs rendszer szériája vezérelhető akár egyedül billentyűzetről is. A különböző menük elérhetők a címeikben található, általában első betűiből. (Betűegyezés esetén ez lehet akár a második, vagy harmadik, stb. betű.). A layout megfelelő kialakításával szintén segíthetünk: pl. egy kisebb felbontáson is megfelelően működő honlapon, nem kell annyi mozgást végeznünk. Egy másik példa: a látássérült embereket is segíthetjük a honlap ergonomikus kidolgozásával: ha egy dokumentumot nyomtatóbarát verzióban is elkészítünk, azt kinyomtatva a szemnek máris egy olyan, kevésbé terhelő aktivitást teremtünk, amely felhasználók ezreinek lehet kényelmesebb. Egy másik példa: a megerősítési protokollokhoz, a CHAPTA rendszer alkalmazásával különböző érzékszerveket vonhatunk be. (hallás, látás, stb.)

Szellemi fogyatékkal élő felhasználók: az ilyen jellegű betegségekkel élők életük során különösen sok figyelmet és törődést érdemelnek, ezért nekünk is hasonlóképp kell eljárnunk.

A navigációs lehetőségek, a honlapunkon található felíratok, cím adatok, a segítségek egyértelműek kell, hogy legyenek. Ne használjunk szlenget, egyéb zavaró kifejezéseket ilyen tartalmi objektumoknál, törekedjünk az egyszerűségre, a tömör fogalmazásra, és a szándékunk világos kifejezésére.

Korosztályok: hasonlóan hátrányos helyzetűek a gyermekek, és az idősek, és az olyan emberek, akik társadalmi pozíciójukból kifolyólag nem jártasak a számítógépes készségekben. Egyértelműen a tinédzser korosztály még nem tudja megfelelően használni a bonyolultabb webes lehetőségeket, az idősebbek nagy része pedig kimaradt az informatika térhódításából: nagyon nehezen alkalmazkodnak a jelenleg uralkodó körülményekhez. Hasonlóképp kell eljárnunk, mint az előző esetben: a világos fogalmazás, az egyértelműen felépített menüszerkezet, az információs blokkok segíthetik a számítógépet, s lehetőségeit még ki nem aknázó felhasználókat.

Hardverfüggő problémák: jelentős hibák, amellyel sokszor a profibb webes szakemberek sem törődnek megfelelően. A felhasználók konfigurációi igen eltérőek, és ehhez érdemes alkalmazkodnunk, ha azt szeretnénk, hogy alkotásunk a lehető legnagyobb tömeghez jusson el. Gyakori tény, hogy a szabványoknak nem megfelelő dokumentumokat bocsátunk ki, ennek köszönhetően a honlapjaink a különböző böngészőknél különféle kompatibilitási hibákat generálhatnak kimenetként. A formázások szétcsúszhatnak, a pozícionált szövegrészek részei levágódhatnak, a böngésző által nem támogatott parancsok nem hajtódhatnak végre, így hibákat okozhatnak. Másik tipikus probléma, amit már említettem a layout kialakításánál, az az, hogy a web fejlesztők hajlamosak megfeledkezni arról, hogy nem csak a saját rendszerigényeikhez mérten kell végrehajtaniuk a feladatot: kimarad a kód, és az elrendezés optimalizálása, ezáltal egy-egy weblap használata lassulhat, vagy akár teljesen meg is akadhat. Az ezt okozó jellemző objektumok általában multimédiás jellegűek: animált .gif formátumú mozgóképek, flash objektumok, amelyek hangfájlokkal is el lehetnek látva. Ez utóbbi probléma különösen zavaró tud lenni, ha a portál menürendszere a Macromedia által kívánt flash lehetőségeken alapul, meg is béníthatja a navigációt.

A W3C külön szabvány bocsátott ki ennek szellemében, mely már magyarul is elérhető számunkra. Ez a WCAG 2.0 – Web Content Accessibility – Web Akadálymentesítési Útmutató. [13] A dokumentum olyan szabályokat tartalmaz, módszertant teremt, amelyeknek

előírásait betartva akadálymentesíthetjük honlapunkat. Rengeteg olyan lehetőséget említ, amelyet én is érintettem.

A dokumentum tanulmányozását, s elvi mondanivalójának elsajátítását minden webes felületet fejlesztő szakembernek javasolják; jómagam is megpróbáltam átvenni, s alkalmazni ezeket a hasznos tanácsokat, amely az alkalmazásomon megfigyelhető.

1.3.4. – Multimédia:

Bár a multimédiás elemek használatát nem igényli minden honlap típus, ezek a fájlok jelentős szereppel bírnak manapság a webes technológiák szabványaiban.

Mit jelent a multimédia?

Általában médiumnak nevezzük az információk terjesztésére és bemutatására szolgáló eszközöket. [12]

A médium többes száma a média: a multimédia egy olyan gyűjtőfogalom, amely több médiumot foglal magában. Az informatika területén a multimédia olyan információközlő objektumokat, fájlokat jelent, amelyek egyszerre akár több érzékszervünkre kívánnak hatást gyakorolni.

Az interneten használható és elterjedt képfájlok kiterjesztései a **.jpg**, vagy **.jpeg**, az animálható **.gif**, és az alfa csatornával felszerelhető **.png** kiterjesztés. Használható a Windows bitmap képfájlja is, avagy a **.bmp** kiterjesztés, de mivel ez egy tömörítetlen információtárolást jelent, a fájl mérete a pixelek számának növekedésével drasztikusan megnőhet. Ezt leszámítva a többi kiterjesztés mind valamilyen tömörítési eljárást használ. A jpeg kiterjesztés veszteséges tömörítéssel jár: a kép minősége, egy konvertáláskor jelentősen romlik, cserébe a fájlméret minimálisra csökken. A png, és a gif veszteségmentes tömörítési eljárást használ, így a kép minősége alig, vagy egyáltalán nem romlik, méretük viszont valamivel nagyobb lehet, mint a jpg kiterjesztés esetében. A mostanra legelterjedtebb képformátumok a jpg és a png, a gif fájlokat inkább animációk formájában használják, bár manapság ez is egyre kevésbé jellemzőbb, hála a flash animácók térhódításának.

Manapság egyre nagyobb szerepet kapnak a mozgóképek, az animációs fájlok, a videók. Kezdetben a fentebb említett gif formátum látta el az animált objektumok szerepét, de hála a

flash technológiák térhódításának, ez a formátum kiszorult a piacról, elavult. Az Adobe cég flash lehetőségei gyorsan nőttek, hála a vektorgrafika használatának.

Ha a számítógépi grafikát vitatjuk, két alapvető megjelenítési forma juthat eszünkbe: a pixelgrafika, és a vektorgrafika. A pixelgrafika egy raszteres állományformán alapul, amely a pixelek egy négyzetrácsos felosztását jelenti. A vektorgrafika ezzel ellentétben geometriai primitíveket vesz alapul a grafikus objektumok leírásához: pontokkal, görbékkel, szakaszokkal dolgozik. A két grafikus módszer közt hamar észrevehetjük a különbséget, amint egy képet méretezni próbálunk. A pixelgrafikával készített kép egy bizonyos nagyítási fok után erős minőségromlást mutat, míg a vektor grafikusan megtervezett objektum nagyításánál, könnyed felépítéséből adódóan a számítógép újraszámolja a képpontok helyeit: nincs minőségromlás.

A flash technológia a vektorgrafikán alapul. Természetesen a vektorgrafikus programok is képesek pixelgrafikus állományokat, bitmapokat kezelni, de egy nagyítás esetében, ugyanazt a minőségromlást eredményezik (hiszen leírási módjuk különbözik). A flash fájlok mérete kicsi, ezért remekül alkalmazhatók a webes felületen. Méretük persze megnőhet, ha audió fájlokat társítunk hozzájuk, de akár a képfájlok, a hanganyagok is remekül tömöríthetők.

A weben nem használhatunk audió, vagy videó fájlokat külső szoftveres segítség nélkül: mindig szükségünk van valamilyen lejátszó program, plugin segítségére. Erre több módszerünk is adódhat. Használhatjuk az operációs rendszerek segédprogramjait, pl.: a Microsoft Windows estében a Windows Media Playert. Egy kényelmesebb, és gyorsabb elérést biztosít számunkra a különféle flash alapú lejátszók használata. Ezeket beágyazhatjuk a honlapunkba, hogy lejátszható tartalomként üzemeljenek. Ezt a HTML 5 <object>, <param>, és <embed> tagjainak használatával tehetjük meg.

```
<object width=”120” height=”20”>
<param name=”nev” value=”#”></param>
<param name=”allowscriptacces” value=”always”></param>
<embed allowscriptacces=”always” height=”20” width=”100%” src=”#” type=”application/x-shockwave-flash”></embed>
</object>
```

Az <object> tag friss része a manapság fejlődő webes környezetnek, a legtöbb böngésző egyelőre csak részelegesen (objektum típustól függően) támogatja. Egy objektumot hoz létre. Eredetileg az <img> és az <applet> tagokat is helyettesíteni kívánta, de egyelőre a HTML 5. szabványának fejlesztői különféle hibákba ütköztek. A <param> tag egy változót, vagy egy paramétert deklarál az object tagnak. Szükséges attribútuma a name – névmegadás, de megadhatjuk a MIME típust is, adhatunk neki értéket, azonosítókat. Érdekesség: a HTML nyelvben ennek a tagnak nincs záró eleme, míg az XHTML-ben kötelező lezárnunk. Az <embed> tag maga a beágyazó parancs, amellyel egy fájlt, vagy plugint betölthetünk egy meghatározott helyre az oldalon. Megadhatók a beágyazott objektum méret adatai, a MIME típus, és maga a forrásfájl elérési útja.

A beágyazott tartalmak használata egy összetettebb része az objektumok weblapra töltésének, és megjelenítésének, de manapság rengeteg fájlmegosztó portál, és rendszer biztosítja a felhasználó számára a feltöltés után generált embed kódot, amelyet csak be kell másolnunk saját fájlunk forrásába. Ilyen lehetőségeket kínál számunkra, a népszerű videó megosztó webalkalmazás a YouTube, vagy a SoundCloud (hangfájl megosztó), a PhotoBucket (képmegosztó portál).

A flash tartalmak habár potenciálisan megnövelik multimédiás lehetőségeinket rengeteg hátránnyal járnak.

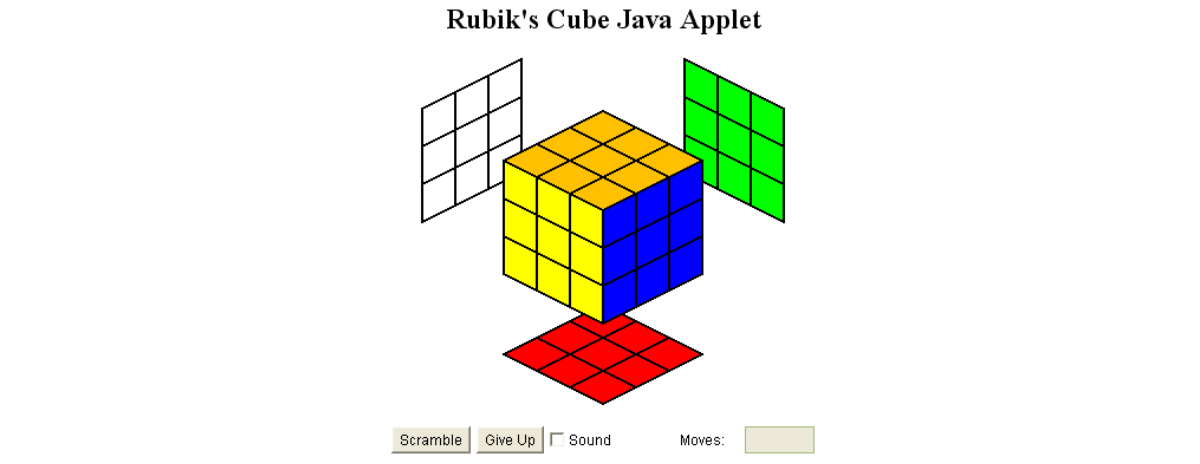
- Rontják az akadálymentesítést: a HTML oldalak esetében előfordul, hogy valamennyi erőforrás nem mindig elérhető. Ilyen esetre találták ki, pl. a képfájlok esetében használatos alt attribútumot, amely, ha a képfájl nem tud betöltődni, egy alternatív szöveget jelenít meg helyette, amely, például ha a kép egy horgonyba van ágyazva, ugyanúgy navigációként működik. A flash objektumok esetében ez nem működik. Ha egy menü valamilyen multimédiás flash platformon íródott, és nem elérhető, megbéníthatja a hiperlinkek kezelését.
- Rontják a teljesítményt: ez a technológia nagyobb gépigényt ír elő, mint az átlagos webes dokumentumok, és médiája. Minél több ilyen elem jelenik meg a weblapon annál több számítógépes erőforrást fog igénybe venni az üzemeltetésük (tartsuk észben, hogy ezek a fájlok nagyobb képekből is építkezhetnek). Ez az alkalmazás teljes megbénításához is vezethet.
- Blokkolás: néhány böngésző automatikusan blokkolja a flash tartalmakat (pl. a Konqueror), ugyanis használatuk olyannyira népszerű lett néhány webportál esetében, hogy

rosszindulatú felhasználásuk nagyobb veszélyt is jelenthet. Ezek az úgynevezett flash-addok, pop-up alkalmazások, amelyek az internetes férgek, vírusok, és az adathalászat leggyakoribb elemei.

A multimédiás lehetőségek konkrét programozási nyelvhez köthető reprezentánsai a grafikus JAVA objektumok, az úgynevezett Appletek. Ezek különálló objektumok, amelyek webes használatra lettek elkészítve a Sun cég JAVA programnyelvén. Futtatásukhoz a Java Runtime Enviroment - Java Futtatási környezet, és a Sun JAVA Plug-in szükséges. Beágyazásuk egy sokkal egyszerűbb módon történik:

<applet code=”program.class” width=”640” height=”480”></applet>

Egy JAVA applet megjelenítéséhez a tagben meg kell adnunk a program lefordított .class kiterjesztésű fájlját, a code nevű attribútum értékeként. Meg kell adnunk a méreteket is, így egy általunk definiált ablakban fog megjelenni a program. Ezen kívül adhatunk alt – alternatív szöveget, horizontális, vertikális pozíciót, nevet, elrendezést az objektumnak, de ezek már opcionálisak.



6. ábra - Applet megjelenítése: rubik kocka

A HTML 5. szabványában az <applet> tag használatát elavultnak ítélték, jelenleg azon dolgoznak, hogy helyét az <object> vegye át. Egyelőre különböző hibák merülnek fel az <object> taggel (JAVA plugin használata szükséges). a Sun véleménye szerint, az <object> használata nem konzisztens, javasolják, hogy jelenleg használjuk tovább az <applet> taget.

1.3.5. – Tipográfia

Elérhető szöveges tartalmunk megjelenítésének egy fontos szempontja a megjelenítés, és az ebből fakadó olvashatóság. Ahogy egy kézzel írott levél, vagy egy nyomtatott könyv, úgy egy elektronikus dokumentum is olvashatatlanná válik, ha összecsapjuk a tervezési folyamatokat, vagy nem gondoljuk át az aktuális programozási nyelv adta lehetőségeket.

A betűket csoportosíthatjuk típusok szerint. Két alapvető forma a serif - talpas és sans-serif - talp nélküli betű. Az előbbit hosszabb szövegek rögzítése esetén használjuk, hiszen a talpak vezetik a szemünket, segítenek az információk feldolgozásában. A serif betűk rövidebb szövegek megjelenítésére alkalmasak, számítógépen jobban olvashatóak. Nem tűnhet túl súlyos problémának, ha a nem megfelelő típusokat alkalmazzuk, ennek ellenére erősen ronthatják a felhasználói élményt.

Egy-egy betűtípus kiválasztásánál, programozási nyelvek esetében észben kell tartanunk azt, hogy nem minden számítógép rendszere egyforma. A saját operációs rendszerünk készletei egy másik számítógépen talán nem is léteznek. A CSS nyelv például remek alternatívákat kínál a típushelyettesítésekre.

```
p.formazas{
font-family: arial narrow, arial, sans-serif;
}
```

Megadhatunk egy konkrét betűtípust, egy azt helyettesítő betűcsaládot, valamint egy alternatívát, miszerint ha az aktuális számítógépen nem található meg egyik általunk megadott típus, akkor használjon: serif, sans-serif, vagy monospace (rögzített szélességű betűket.)

HTML nyelvben hasonlóan járhatunk el:

```
<font face="Times New Roman, Courier New, Arial">string</font>
```

Bár a <font> tag még működőképes, használatát ma már nem javasolják, a HTML 5-ben már nem lesz elérhető.

CSS esetében a szövegek beállításaiért két parancs felelős: a font, amely a betű beállításaival foglalkozik, és a text, amely a konkrét szöveg pozicionálását, és egyéb tulajdonságait hivatott módosítani.

A text parancsok nagyobb csoportokra oszthatók.

- font-style: stílus beállítások. Értékei: normal, italic – dőlt, oblique – a kurzív betű egy másik típusa.
- font-size: méretek, amelyek megadhatók pixelekben, pontokban, emphasisban (1 em = 16px), és százalékos értékben.
- font-variant: kiskapitális megjelenítés.
- font-weight: kihangsúlyozás. (félkövér)

A text parancsok igen hasznos formázási paraméterekkel rendelkeznek.

- color: színmegadás.
- direction: a szövegolvasási, és írási irányát adja meg. (Különböző kultúrákban különböző lehet az olvasás iránya, például: Japán.) Értékei: ltr – left-to-right, rtl – right-to-left, és inherit (egy szülő elemhez viszonyul).
- letter-spacing: a karakterek közti távolság beállítása. Értékei: normal, inherit, vagy megadunk egy értéket pixelekben.
- line-height: sormagasság. Értékei: normal, inherit, egy számérték, vagy pixel, pt, cm, százalék.
- text-align: szöveg pozicionálás. Értékei: left – balra, center – középre, és right – jobbra rendezés, ezen kívül beállíthatunk justify – sorkizárást, vagy használhatjuk az inherit értéket.
- text-decoration: szövegdíszítő elemek hozzáadása. Értékei: overline – szöveg fölötti áthúzás, line-trough – áthúzott szöveg, underline – aláhúzás, blink – villogás beállítása. (Ez utóbbi sok böngésző által nem támogatott.)
- text-indent: a bekezdés első sorának eltolása. Értékei lehetnek pixelek, egy százalékos érték, vagy az inherit.
- text-shadow: árnyékolás.
- text-transform: kapitális, vagy felső, alsó index beállítások.

- unicode-bidi: a megváltoztatott írásirány hibáit hivatott kiküszöbölni. Értékei: inherit, normal, embed, bidi-override.
- vertical-align: horizontális elrendezés. Értékei: top – tető, middle – közép, bottom – alul.
- white-space: a pozicionáló tér beállításai. Értékei: normal, nowarp – hosszú szövegek törésének kikapcsolása, pre – a böngésző határozza meg, sortörésnél, pre-line – ha szükséges, és sortörésnél, pre-warp – a böngésző határozza meg, és sortörésnél, inherit.
- word-spacing: szövegek közti távolság beállítása. Értékei: normal, inherit, és egy pixel, cm, em, pt érték.

Bár ezek a parancsok nagyfokú módosítást engednek, a mai napig felmerülnek kompatibilitási hibák, például az inherit értéknél, amelyek a Microsoft Internet Explorer nevű böngészőjének régebbi verzióinál jelentkeznek. A blink utasítást a legtöbb böngésző nem támogatja. Ezek függvényében kell elkészítenünk weboldalunkat.

2. Java Strata – Java oktató weboldal

A szakdolgozatomhoz készített alkalmazásom témájául a Java nyelv programozásának oktatását, rendszerezését választottam. Ennek keretében egy weboldalt készítettem, amely a webfejlesztés legelterjedtebb programozási, és script nyelveit használja.

7. ábra - Java Strata

Az oldalt **Java Strata**nak neveztem el, ugyanis a strata jelentése tároló, konténer. A weblap elkészítése, és felépítése során két főbb alapelvet vettem figyelembe. Egyik az oldal felhasználók szerinti alakíthatósága. Sok esetben a közösség által mutatott példák legalább annyira hasznosnak bizonyulnak, mintha a problémát saját elképzeléseink szerint próbáljuk megoldani. A Java Strata nem csak tudást, hanem konkrét fájlokat, és az azokhoz kapcsolódó magyarázatos leírásokat kíván megosztani a közönséggel. A másik szempont a tartalom megfelelő magyarázata, visszakereshetősége. A kiépített hivatkozás rendszer könnyen átláthatóvá, értelmezhetővé teszi a megosztott forrásfájlokat.

2.1. – Az oldal bemutatása

A Java Strata egy keretrendszerbe ágyazott, rögzített szélességű weboldal, amelynek főbb felülete HTML nyelven íródott. A formázásokért egy külső CSS fájl felelős, a szerver oldali kommunikációt pedig PHP scriptek bonyolítják le. A honlap dinamikus elemeinek megjelenítéséért az általam készített JavaScript függvények felelősek. Felépítése a tartalom gyors megosztását teszi lehetővé. Egy banner részt, egy vízszintes, és egy jobb oldalra rendezett függőleges navigációs felületet, valamint két lábléc területet tartalmaz. Az elrendezés a különböző felbontásokhoz adaptálódik. Egyes böngészőkkel apróbb kompatibilitási hibákat mutat, de a lap mindegyiken működőképes, és gyorsan használható volt. Mind a HTML dokumentumok, mind a CSS fájlok a W3C validálási lehetőségeit kihasználva készültek, ezért a HTML 4.01 és a CSS Level 2 szabványainak szigorúan megfelelnek.

Az első navigációs felület a weblap főbb funkcióit reprezentálja:

- Főoldal: egyszerű visszatérés a honlap hírmegosztó fájljához.
- Célok: rövid bemutató oldal, amely információt nyújt a honlap történetéről, céljairól, az üzemeltetőről, és a teszt eredményekről: mely felbontásokon, és böngészőkön volt kipróbálva, ha valamilyen probléma merült fel, mi is az.
- Oldaltérkép: a weblapon található linkeket hivatott összegyűjteni, rendszerezve, a megfelelő kategória címekkel. Nem csupán a fájlokra történő hivatkozások érhetők el, hanem a különböző fájlokon található lapozó horgonyok.
- Lexikon: definíciókat, főbb metódusokat, eljárásokat, interfészeket, és az általuk előírt programelemeket tartalmaz. Célja, hogy rövid magyarázatot adjon a honlapon fellelhető tartalom programkódjaihoz, megoldásaihoz.
- Kezdőknek, használóknak, haladóknak: ezeken a menüpontokon keresztül érhetők el a feladatok, amelyek rendelkeznek egy azonosítóval, egy beküldési dátummal, egy beküldővel, és egy kategóriával. Egy-egy programhoz társíthatók címkék, amelyek alapján az aktuálishoz hasonló feladatokat kereshetünk. A feladatok három fő részből állnak: egy bevezető rész, amelyben leírásra kerülnek a főbb komponensek (import csomagok, főbb függvények, interfészek, osztályok), és a célok. A következő tartalmi blokk a programkódot, vagy böngészőben futtatható alkalmazás esetében (pl. Applet) magát a kipróbálható állományt

jeleníti meg. Ez alatt, működési elv néven megosztásra kerülnek azok az instrukciók, magyarázatok, amellyel a program elkészült.

- Beküldés: utolsó menüpontként lehetőségünk nyílik feladatot küldeni az üzemeltető(k)nek, egy e-mail formájában, vagy feltöltéssel.

A jobb oldali menüsor a gyors navigációért felelős. A gyakorlott felhasználók feladat típusonként ugorhatnak az első olyan feltöltött tartalomhoz, amelynek a típusára kattintottak, vagy rendezhetik azokat a hozzájuk rendelt tagek – címkék szerint. Az egyéb lehetőségek olyan menüpontokat kínálnak, amelyek szintén hasznosak lehetnek számunkra.

- Híreink: ugrás a főoldalra, használata akkor lehet praktikus, ha a felhasználó a kódot böngészi. Ilyen esetben nincs rá szükség, hogy visszagörgessük a weboldalt a böngészőablak tetejéig.
- Friss feltöltések: a honlapra feltöltött fájlokat listázza ki, időrendi sorrendben: a legelső sor mindig a legfrissebb programot mutatja.
- Keresés: a Google Custom Search szolgáltatását felhasználva létrehoztam egy egyedi keresőmotort, amely az általam legjobbnak ítélt Java programfejlesztéssel foglalkozó oldalak tartalmai közt keres egy megadott kifejezést.
- Fórum: egy telepített PHPBB fórummotor, amely a felhasználóknak kommunikációs felületet biztosít.
- Linkek: a Java fejlesztéssel foglalkozó hivatalos, és nem hivatalos portálokat, oldalakat, blogokat gyűjtöttem össze.
- Statisztikák: a MyStat.hu ingyenes webstatisztikai oldal szolgáltatását használó, oldal betöltéseket számláló script információkat nyújt az alkalmazás látogatottságáról.

Az ezen kívüli navigációs lehetőségek a következők: az első láblécben található adatoknál megtalálható a dolgozatom címe, amely a Debreceni Egyetem Elektronikus Archívumára mutat. A második láblécben található link pedig a logóban található fénykép fotográfusának webes albumát hivatott megjeleníteni. A felhasznált kép ingyenesen használható, és általam lett módosítva.

2.1.1. – Hírfolyam

A honlap betöltésekor a „híreink” gombra is kattintva megjelenített hirek.html oldallal kezdődik, amely a táblázat egyik cellájába töltődik be egy <iframe> használatával. A hír szövege alatt található lábrészben egy bal és egy jobb információs részt találunk. Bal oldalt a beküldési adatokat találjuk, jobb oldalt pedig két lehetőségünk adódik.

A „bővebben” link a hír szövegét bontja ki, ugyanis a megjelenítésre kerülő információk alapból csak figyelemfelkeltő jellegűek. A tartalom kibontását egy JavaScript kezeli.

```
<a href="#" onclick="
if(document.getElementById('bov1').className=='viz2'){
document.getElementById('hir1').className='c_hidden';
document.getElementById('bov1').innerHTML='Bővebben';
document.getElementById('bov1').className='viz';
}
else{
document.getElementById('hir1').className='c_display';
document.getElementById('bov1').innerHTML='Vissza';
document.getElementById('bov1').className='viz2';
}">
<span id="bov1" class="viz">Bővebben</span>
</a>
```

A link egy üres # azonosítóra mutat, ugyanis nincs konkrét oldal, amit be kell töltenie. Az onclick eseménykezelő attribútum egy scriptet tartalmaz, amely feltétele azt vizsgálja, hogy a bov1 azonosítóval rendelkező elemhez rendelt osztály neve viz2 e. Hogy megértsük a feltételes elágazást, célszerű először az else ágat megvizsgálnunk, ugyanis ez fog végrehajtódni először. Az első utasítás a hir1 azonosítójú elemhez rendelt osztály nevét c_displayre állítja. A CSS adta lehetőségek megengedik, hogy a tartalmat megjelenítsük, vagy elrejtsük. Ezért a display parancs felelős, amelynek két fontos értékét fogjuk használni. Ezek a: display:block; - blokk szerű megjelenítés, és a display:none; - tartalom elrejtése (az elrejtett elem nem foglal üres helyet). A c_display osztály megjeleníti a tartalmat. A következő sor az innerHTML parancs segítségével a bov1 azonosítójú elemet „Visszára” változtatja. A bov1

azonosítót a „Bővebben” linkhez rendeltem egy taggel. Ezzel azt értem el, hogy a „Bővebben” felírat „Vissza” lett, amely jelzi, hogy az aktuálisan olvasott hír ki lett bontva. Az utolsó parancs megváltoztatja a linkhez rendelt osztály nevét, és ezzel a legközelebbi kattintásra teljesül a feltétel igaz ága. Ez az elágazás pedig elrejti a tartalmat, és visszaállítja a feliratot, valamint a link osztálynevét.

A „Hozzászólások” nevű link az előző megjelenítési algoritmuson alapul, viszont más jellegű tartalmat jelenít meg. Egy formot és egy táblázatba ágyazott iframe-et, azon belül is egy php oldalt jelenít meg, amely a hírekhez érkezett hozzászólásokat mutatja. A form összeköttetésben áll a php oldallal, egy <textarea> - több soros szövegbeviteli mezőt, egy <input type=”text”...> - egy soros input mezőt, és egy <input type=”submit”...> elküldő gombot tartalmaz. A hozzászólások kezeléséért két php fájl felelős, melyeket egy ingyenes script felhasználásával készítettem. [14] Ezek: a megjelenit.php, és a hozzaad.php.

A hozzaad.php kódja:

```
<?php
if ($message!=''){
	$message=str_replace("\n","<br>",$message);
	$message=strip_tags($message, '<br>');
	$new_row='<div>' . ($message) .
	'<br>' . date('Y.m.d - H:i') . ' - ' . Hozzászóló . ': ' . strip_tags ($name) .
	'</div>';
	$old_rows=join('', file('hozzaszolas.txt'));
	$fileName=fopen('hozzaszolas.txt', 'w');
	fputs($fileName, $new_row . chr(13) . chr(11) . $old_rows);
	fclose($fileName);
}
include ("megjelenit.php");
?>
```

A script egy feltétellel indul, amely megvizsgálja, hogy a $message változó üres e, tehát írtunk e valamilyen üzenetet. A igaz állítás esetében, a str_replace() függvény beiktatja a sortöréseket, majd a strip_tags() engedélyezi a használatukat. Megkezdődik a hozzászólás, és

adatainak kialakítása. a $new_row változóba letárolunk egy taget amely a következőket tartalmazza: a $message változót, majd következő sorban a dátumot (év,hónap,nap – óra, perc), a Hozzászóló felirat után pedig a HTML formban megadott nevet, amely a $name változóban lesz tárolva. Az $old_rows változóban tároljuk a friss sort, hozzákapcsolva az esetleges régiekhez. Az összekapcsolást a join(); függvény végzi. A $fileName változóban az fopen() függvénnyel megnyitom a hozzaszolas.txt szöveges állományt, és a w paraméterrel jelzem, hogy tartalma módosulni fog (w - write). Az fputs() összekapcsolja a változóimba tárolt adatokat, az fclose() pedig lezárja a fájlműveleteket. Végezetül az include meghívja a megjelenit.php-t.

A megjelenit.php kódja:

```
<?php
$fileName=file("hozzaszolas.txt");
$rows=count($fileName);
if($rows>10){
	if(!isset($row)){
		$row=0;
	}
	print("<table><tr><td width=\"50%\">");
	if($row>0){
		echo "<div> << <a href=\"megjelenit.php?row=" . ($row-10) . "\" target=\"hirek.html\">A következő 10 hozzászólás =></a></div>";
	}
		print("</td><td width=\"50%\">");
	if(($rows-$row)>10){

		echo "<div class=\"elozo_oldal\"><a href=\"megjelenit.php?row=" . ($row+10) . "\" target=\"hirek.html\">Az előző 10 hozzászólás</a> >> </div>";
	}
	print("</td></tr></table>");
	for($j=$row; $j<($row+10); $j++){
		echo $fileName[$j];
```

```
        }
}else{

        for ($i=0; $i<$rows; $i++){
                echo $fileName[$i];
        }
}
?>
```

A script elején deklarálok két változót. Az első a hozzaszolas.txt szöveges állományt tárolja, ez a $fileName. A $rows nevű változóba a $fileName-ben letárolt hozzaszolas.txt állomány sorait számoltatom meg a count(); függvénnyel.

A következő feltétel megvizsgálja, hogy a $rows változó értéke nagyobb e tíztől. Ez azért szükséges, mert szeretném, ha egy oldalon egyszerre tíz hozzászólás jelenne meg. Egy lapozási funkciót fogok kiépíteni, amellyel megtekinthetők a régi hozzászólások is. Ha a változó értéke nem nagyobb tíztől, egy for(); ciklussal kiíratom a $fileName tömbként funkcionáló változó értékeit, amelyek a sorokat fogják jelenteni.

A következő feltétel azt vizsgálja, hogy a $row változónak van e értéke. Ha nincs, akkor adunk neki egy 0 kezdőértéket. A print utasítással készítünk egy táblázatot, ugyanis ezzel célszerű megjeleníteni a hozzászólások lapozásához szükséges linkeket. Az if($row>0) elágazásnál elkészítjük a „következő oldal” linket, hiszen az első feltételünk igaz ágát tervezzük, ami szerint több mint tízsornyi bejegyzés szerepel a hozzászólásoknál. A $row változó értékéből kivonunk 10-et. Nyitunk egy új <td> - oszlopot. A következő feltétel megvizsgálja, hogy a sorok számból ha kivonjuk a $row változó értékét nagyobb lesz e az érték, mint 10. Ha igen, akkor létezik előző oldal. Elkészítjük az „Előző oldal” linket, a $row változóhoz pedig hozzáadunk 10-et. Lezárjuk a táblázatot, majd egy for(); ciklussal kilistázzuk a hozzászólásokat.

Ez a script adatbázis kezelés nélkül lehetővé teszi, hogy a felhasználók hozzászólást hagyjanak a főoldalon. A hozzászólások a hozzaszolas.txt nevű fájlban tárolódnak.

2.1.2. – Lexikon

Ez a menüpont azért jött létre, hogy segítséget nyújtson a Java programozási nyelv fogalmai közt bizonytalanul mozgó felhasználóknak. Rövid definíciókat tartalmaz, amelyek érinthetnek a környezetben használt fogalmakat, osztályokat, mezőket, metódusokat, eljárásokat, interfészeket, vagy formokat, stb. A felületen kategóriák találhatók, ezekhez vannak beosztva a linkek. A hivatkozások a hozzájuk tartozó megfelelő definícióra mutatnak, amelyek egy-egy a HTML nyelvben használatos elválasztó vonallal, a <hr> taggel vannak elkülönítve.

Az elemi típusoknál ki van listázva a megfelelő maximum és minimum értékek, valamint példákat írtam a deklaráláshoz. A fogalmak esetében is megpróbáltam szemléltető sorokat írni. A metódusok, eljárások esetében el vannak magyarázva a paraméterek, a használatuk, ha tartoznak valamilyen interfészhez, az szintén meg van említve. Az import csomagoknál feloldottam a rövidítéseket, az interfész kategóriában pedig leírtam, hogy melyik milyen metódusok deklarálását írja elő.

2.1.3. – Kezdőknek, használóknak, haladóknak

A beküldött feladatokat találhatjuk meg ezekben a kategóriákban, megvalósítási szintjükhöz mérten csoportosítva. Mint a menüsorok általános bemutatásánál említettem, egy feladat megjelenítő lap 3-4 részből áll: egy fejlécből, amint szerepel egy azonosító, a beküldés dátuma, a beküldő, és a program típus. Ide sorolható még a programokhoz megadott címkék sora is. Ezek a linkek egy a címkék szerint rendezett listával rendelkező oldal megfelelő horgonyára mutatnak, így ilyen módon is visszakereshetők a feltöltött alkalmazások. Ezt követi a program címe, bemutatása, főbb komponenseinek listája. Ezek alatt megjelenik a programkód, vagy ha böngészőben futtatható állományról van szó, a tesztfelület. Végül a programkód alatt megtalálható a feladathoz tartozó magyarázat, működési elv címen.

A programkód sorszámozva van, és színkiemeléseket tartalmaz: a fontosabb sorok pirossal, míg az üres sorok szürkével vannak jelölve. A működési elv leírásában a fontosabb fogalmak, műveletek, metódusok, vagy előírások összeköthetők a programkód soraival, így miközben a program miértjeit olvassuk, nem kell elvesznünk, egyből a megfelelő sorra ugorhatunk. Ez az ugrás működik a program bevezető részében található fontosabb komponensek felsorolásánál is. Ez a funkció beküldés menüpontnál, a megfelelő kitöltött mezőknél alapból beállítható. A feladatok megjelenítésénél fontosnak tartottam, hogy egy-egy parancs, vagy művelet könnyen

visszakereshető legyen. A felhasználó egy sokkal kényelmesebb használati élményben részesülhet, ha nem kell manuálisan minden egyes olyan pontot visszakeresnie, ami érdekli. A weblapon jelenleg az általam beküldött, és készített, Java programozási kurzusokhoz készített feladatok találhatók.

2.1.4. – Beküldés

A beküldés menüpont felelős a feltöltendő tartalom kezeléséért. A feltölteni kívánt program egy kiépített <form> rendszeren keresztül postázható e-mailben az üzemeltető által megadott címre. Az input mezők közt találhatók kötelezően kitöltendő és opcionális mezők. A kötelező inputokat egy * jelöli a címük előtt. Ha az űrlapot üresen, vagy nem megfelelő mennyiségű kötelezően kitöltendő adattal próbáljuk meg elküldeni, akkor a weblap alert ablakokkal jelzi nekünk, hogy programunk leírása hiányos. Néhány mező esetében lehetőségünk nyílik a mezők formázására, ezzel is megkönnyíthetjük a feltöltött tartalmat felügyelő, és feltöltő személy, személyek munkáját. Az import csomagok, főbb metódusok, és főbb osztályok mezőinél lehetőségünk nyílik horgonyokat beállítani, amelyek majd a kódban megjelölt sorokra hivatkoznak. A kódblokkban pedig lehetőségünk nyílik megadni a horgonyok nevét, és egy piros színnel kiemelhetjük a számunkra fontosnak ítélt sorokat. Ezeket a <textarea> mezők fölött látható <a>, , illetve a „!” és „/!” gombokkal tehetjük meg. Működésükért egy egyszerű JavaScript parancs felelős.

```
…onclick="document.forms[0].kod.value+='<a name=&quot;&#35; &quot;>';"…
```

A 0. form kod nevű mezőjének értékét módosítom, hozzáfűzök egy <a href=” ”>, vagy <a name=” ”> taget, attól függően, hogy hivatkozást, vagy horgony nevet szeretnék e beállítani.

A form végén lehetőségünk nyílik egy .java fájl feltöltésére, amelyet egy php script fog elvégezni.

```
<?php
$konyvtar='/feltolt/';
$fajlnev=$konyvtar . basename($_FILES['fajl']['name']);

if($_FILES[fajl][size]<1000){
```

```
if(move_uploaded_file($_FILES['fajl']['tmp_name'], $konyvtar)){
        echo "Sikeres feltöltés!";
}else{
        echo "Probléma történt a feltöltés során!\n A szerver nem fogadta a fájlt!\n";
}
echo "A fájl információi: ";
print_r($_FILES);


}else{
echo "Hibás fájlmegadás! A fájl vagy nem .java kiterjesztésű, vagy túl nagy a mérete.";
}
?>
```

Létrehoztam egy $konyvtar nevű változót, amely azt az elérési utat tartalmazza, ahová a fájl a szerveren kerülni fog. A $_FILES globális változót felhasználva hozzáadom a feltöltésre kerülő fájl nevét a $konyvtar változóban tárolt elérési úthoz, majd beiktatok egy tartalomfigyelő elágazást. A feltétel if($_FILES[fajl][size]<1000) figyeli, hogy mekkora a mérete a feltöltött objektumnak. A megadott kritérium 1000 kbyte. Ha a méret nem megfelelő, egy hibaüzenetet generál.

A if(move_uploaded_file($_FILES['fajl']['tmp_name'], $konyvtar)) feltétel azt figyeli, hogy a fájl feltöltése sikerült: tehát az ideigelenes nevet kapott, letárolt fájl, a $konyvtar változóban definiált helyre került e. A move_uploaded_file(fájlnév,célkönyvtár); függvény végzi az állománymozgatást PHP-ben. Sikeres feltöltés esetén igaz értéket ad vissza. Az else ág egy hibaüzenet küld, majd leírja a feltöltendő fájl adatait: a fájl nevét, típusát, a kapott ideiglenes nevét, a hiba kódját, és a méretét kbyteban.

A feltöltés sikeressége a szerver engedélyeitől, és a rajta található mappák írás / olvasási attribútumaitól is függ. A legtöbb ingyenes tárhely szolgáltató például valamilyen módon tiltja az ilyen típusú kommunikációt, és habár hibák nem merülnek fel a feltöltésnél, a fájlok nem jelennek meg a szerveren. A temp és a feltöltés mappáihoz érdemes teljes írhatósági jogot rendelni. Ez megtehető PHP utasítással, vagy egy FTP kliensen belül. A PHP erre a chmod(); függvénnyel képes. Például:

chmod(„/temp/feltolt”,777);

- Tulajdonos engedélyek: olvasás, írás, végrehajtás.
- Csoport engedélyek: olvasás, írás, végrehajtás.
- Nyilvános engedélyek: olvasás, írás, végrehajtás.

A PHP fájlműveletek egy gyakori problémája az úgynevezett Safe Mode. Ez egy biztonsági intézkedést, amely a megosztott szerverek problémáit hivatott kiküszöbölni. A PHP legújabb verziójában ezt a lehetőséget már érvénytelenítették, de egyenlőre a mai napig használják ezt a beállítást, ami gátat is szab a szerver-oldali kommunikációnak.

2.2. – Design:

Az oldal kialakításánál egy világos, asszimetrikus design összetételére törekedtem, a zöld és fehér színek dominálnak. Mivel az oldal témájául egy programozási nyelv bemutatását választottam, az arculat ehhez igazodik: a hangsúly a könnyed kezelhetőségen, az átláthatóságon alapul. A honlap tartalma egy a felbontáshoz igazodó táblázatba van rendezve, mely remekül adaptálódik a körülményekhez.

A grafikai elemek kidolgozásában felhasználtam magát a programozási nyelvet is: a banner részben található logót a Java nyelv AWT – Active Window Toolkit import csomagjának használatával készítettem el. Az appletek rajzlap generáló metódusának főbb parancsait használtam: g.fillRect(); - négyzet rajzolása, kitöltő színnel, g.drawImage(); - egy URL és egy Image osztályból felépülő kép kirajzolása. A felirat mögötti kép egy Gimppel készült alfa csatornával rendelkező, áttetsző hátterű .png kiterjesztésű képfájl, amely a CSS lehetőségeivel lett pozícionálva. (nem ismétlődő, jobbra igazított háttérkép.)

A weboldal megjelenítéséért két CSS fájl felelős, amelyek a W3C által kibocsátott CSS Level 2 szabvány kompatibilitását figyelembe véve készültek. Ezek a main.css, amely tartalmazza a tagekhez kapcsolt formázásokat, a kialakított osztályokat, és egy két esetben, az egyedi azonosítókhoz kapcsolt parancsokat. A másik fájl, a search.css a keresőoldal megjelenítéséért felelős.

A vízszintes navigációs menüsor elemei animáltak, a képek cseréjét egy JavaScript parancs végzi:

```
<img border="0" alt="Célok" src="kepek/dj_celok.png"
onmouseover="this.src='kepek/dj_celok_omo.png';" onmouseout="this.src='kepek/dj_celok.png';">
```

Az onmousover eseménykezelő hatására, hivatkozok az aktuális példányra, a this objektummal, amely jelen esetben a képfájl maga, és az src attribútum értékét változtatom meg. Az onmouseout pedig visszacseréli az src értékét az eredeti képfájlra. Ez a megoldás kiszűri a hibát, amely a flash alapú objektumok esetében fennállhatnak: a rendszerigény alacsony, és a menüpontok a szerveren felmerülő hibák ellenére is kezelhetők maradnak.

A programok megjelenítésénél megjelenítem a programkódot, és ha a körülmény engedi magát a futtatható programot. A honlapon végezhető műveleteket ikonok segítik. Ilyenekkel találkozhatunk a letöltések esetében, a beküldés menüpontnál található formázó gomboknál, és a célok menüpont „Kompatibilitás böngészőkkel, és különböző felbontásokkal” című fejezeténél megjelenített képek.

A megjelenített grafikai objektumok az egységességre törekednek. Nincsenek túl kitűnő, élénk, nem oda illő díszelemek, a figyelmet a tartalomról elvonó dekorációs objektumok. Minden objektum a főleg zöld, és a fehér színekből álló designt hivatott reprezentálni. A tipográfia ehhez igazodik: a linkek színe nem üt el a honlap designjától, de jelzés értékűek. A számítógépen jobban olvasható talpatlan betűtípust használom. A néha előforduló hosszabb tartalom megjelenítése ellenére a lap kényelmesen olvasható.

2.3. – Akadálymentesítési törekvések

Alkalmazásom elkészítésénél megpróbáltam a WCAG – akadálymentesítés 2.0 szabvány néhány fontosabb szempontját figyelembe venni.

A navigációra két különböző lehetőséget adtam: egy vízszintes, és egy jobbra sorolt függőleges menüsort készítettem. Egyik a honlappal ismerkedő friss felhasználóknak, valamint a programozási nyelvvel ismerkedőknek lehet hasznosabb, a másik a gyakorlott olvasóknak, akik a feladatokat típusok szerint kezelik. A feladatok forráskódja, és a leírások interaktív olvasási lehetőségeket biztosítanak, valamint a táblázatos elrendezés egy rögzített szélességet biztosít. Mind ezen technikai megvalósítások összessége egy gyors mozgást biztosít a felhasználóknak, amely könnyíthet a mozgássérült közönség böngészésén.

A különböző kitölthető mezőkhöz megpróbáltam megfelelő mennyiségű segítséget biztosítani, így például a beküldés menüpont alatt található feladat továbbító űrlap mezőinél meg vannak jelölve a kötelezően kitöltendő adatok, valamint minden <textarea> és <input> fölött fellelhetők a magyarázó bekezdések. Ezzel figyelembe veszem az akadálymentesítés egy fontos aspektusát, a megfelelő mennyiségű segítség nyújtását az oldal használatához.

Az általam készített alkalmazás nem használ flash technológiát, így a rendszerigénye igen alacsony. Kialakításánál arra törekedtem, hogy a legtöbb felbontáson megjelenítési hibák nélkül fusson. Az arculatot megpróbáltam úgy kialakítani, hogy figyelembe vegye a különböző böngészők közti eltéréseket. A legtöbb program esetében a weboldalam remekül teljesített. A tipográfiai formázások esetében figyelembe vettem a számítógépeken meglévő különböző fájlok meglétét, ezért ha egy font nem található, a rendszer azt helyettesíti. A weblap a HTML új, 5. szabványát veszi figyelembe, nem használt elavult tageket, sem azok lezárásait. Például az <img> egyetlen tagből áll, amelyet a következőkép zártunk le: . Ezt az új szabvány már nem támogatja. A záró tag elhagyását a W3C is javasolja. A fájlok mindegyikét a W3C által biztosított Validator - érvényesítő alkalmazás segítségével teszteltem és módosítottam, így megvalósult az akadálymentesítés egy újabb előírása: az alkalmazás adaptálódik a különböző konfigurációkhoz.

Összefoglalás:

A szakdolgozatom és a hozzá készített alkalmazás példát ad arra, hogy milyen tartalmi tagolások, formai sajátosságok figyelhetők meg egy webes dokumentumokból, és egyéb programozási nyelveken íródott alkalmazásokból álló weblaprendszer esetében. Készítése során arra törekedtem, hogy egészen a weblapkészítés, a webes fejlesztés és design alapjait felhasználva, a manapság elterjedt szerver-oldalú programozást is lehetővé tévő script nyelvekig rendszerezzem összegyűjtött tudásanyagomat. Jól lehet, a lehetőségek tárháza jóval bővebb, mint amennyit én felhasználtam: dolgozatom és a JAVA Strata bemutathatja a webes technológiák jelenlegi trendjeit, és mintát adhat a jelenleg fejlesztés alatt álló szabványok áttekintéséhez. A fájlfeltöltést sajnos nem sikerült kipróbálnom a szolgáltatóm PHP safe mode használata miatt, és a hozzászólások megjelenítésénél is fellépnek apróbb hibák, de mindezek ellenére, az oldal működőképes. A jövőben alkalmazásom hibáit javítani fogom, továbbfejlesztése során kiegészülhetne új dinamikus elemekkel. Ilyenek: a feladatlapokon módosítható tag – címke adatok, konkrét hozzászólás rendszer feladatonként, a Lexikon adatbázisának fejlesztése, a kijelölt kategóriáknak megfelelő programok feltöltése, és a témák további bővítése. Megvalósíthatnám a WCAG 2.0 szabvány által javasolt elvek azon részeit, amelyek jelenleg nem érvényesülnek a weblapom esetében. A program elkészítése során betekintést nyerhettem a szerver-oldali programozás rejtelmeibe, frissíthettem a script nyelvekkel kapcsolatos tudásomat, és egy olyan átfogó képet szereztem a HTML alapú webes fejlesztés sajátosságairól, szabályairól, amely remélem a hallgatóságnak is legalább olyan jó célt szolgálhat, mint nekem tette.

Felhasznált irodalom:

Magyar nyelvű irodalom:

- Az én weblapom szabványos! És az Öné? / Karl Dubost [ford. Sikos László]. – W3C, 2002. – http://www.w3c.hu/forditasok/sikos/minoseg/minoseg.html. - Letöltés ideje: 2011. 04. 08.
- Felhasználó és Design : reklámpszichológia 2. / Bujdosó Gyöngyi. – http://www.inf.unideb.hu/~bujdoso/kurzusok/webdesing/ea/Webdesign06_reklampszicho_2b.pdf. - Letöltés ideje: 2011. 03. 03.
- WCAG Elemzés : Miért kell akadálymentesíteni? / Pataki Máté. – W3C Magyar Iroda, 2010. – http://www.w3c.hu/szolgaltatasok/miertkellakadalymentesiteni.html. - Letöltés ideje: 2011. 04. 05.
- Web Akadálymentesítési Útmutató 2.0 : W3C Ajánlás 2008. december 11. / W3C, 2008. - http://www.w3c.hu/forditasok/WCAG20/. - Letöltés ideje: 2011. 04. 06.
- Web-desing / Jakob Nielsen; [ford. Nyisztor Andor, Tölgyesi Zsuzsanna]. – 427 p. - ISBN-10 963-9548-16-2; ISBN-13 978-963-9548-16-9
- Web design: studios / Ed. Julius Wiedemann. – Taschen, 2005. – 191 p. - ISBN 978-3-8228-4041-2

Angol nyelvű irodalom:

- Constructing ASP.NET Web Pages / Christian Darye, Wyatt Barnett. - SitePoint Pty. Ltd., 2008. – 7 p. - http://articles.sitepoint.com/article/constructing-asp-net-web-pages. - Letöltés ideje: 2011. 03. 30.
- PHP: Hypertext Preprocessor / http://www.php.net. – Letöltés ideje: 2011. 02. 24.
- The principles of beautiful web design / Jason Beaird. – Collingwood: SitePoint Pty. Ltd., 2007. – 170 p. - ISBN-10 0-9758429-6-3; ISBN-13 978-0-9758429-6-3
- W3C : HTML5 differences from HTML4 : W3C Working Draft 13 January 2011 / http://www.w3.org/TR/html5-diff/. - Letöltés ideje: 2011. 03. 12.
- XML Essentials / http://www.w3.org/standards/xml/core. - Letöltés ideje: 2011. 04. 06.
- W3C Schools Online Web Tutorials / http://www.w3schools.com. – Letöltés ideje: 2011.02.21.

Hivatkozások:

[1] Dr. Leslie Sikos informatikus : Webes szabványosítás, szemantikus Web, akadálymentesség, multimédia / http://hu.lesliesikos.com. - Letöltés ideje: 2011. 04. 06.

[2] Bevezetés a HTML-be : HTML alapok / Dave Raggett; [ford. Sikos László]. – W3C, 2005. - 1 p. - http://www.w3c.hu/forditasok/sikos/bevezet/start.htm. - Letöltés ideje: 2011. 04. 06.

[3] XML Essentials / http://www.w3.org/standards/xml/core. - Letöltés ideje: 2011. 04. 06.

[4] The principles of beautiful web design / Jason Beaird. – Collingwood: SitePoint Pty. Ltd., 2007. – 170 p. - ISBN-10 0-9758429-6-3; ISBN-13 978-0-9758429-6-3

[5] The principles of beautiful web design / Jason Beaird. – Collingwood: SitePoint Pty. Ltd., 2007. – 170 p. - ISBN-10 0-9758429-6-3; ISBN-13 978-0-9758429-6-3

[6] Activision / http://www.activision.com. – Letöltés ideje: 2011. 04. 14.

[7] Grow Collective : Bistrol Web Design Co-operative / http://2008.gr0w.com. - Letöltés ideje: 2011. 03. 12.

[8] Pro QC Studio / http://www.pqcstudio.com/. - Letöltés ideje: 2011. 03. 12.

[9] Constructing ASP.NET Web Pages / Christian Darye, Wyatt Barnett. - SitePoint Pty. Ltd., 2008. – 7 p. - http://articles.sitepoint.com/article/constructing-asp-net-web-pages. - Letöltés ideje: 2011. 03. 30.

[10] Bus full of Hippies template / http://safe.tumblr.com/theme/preview/441. - Letöltés ideje: 2011. 03. 31.

[11] Blackle : Energy Saving Search / http://www.blackle.com/. - Letöltés ideje: 2011. 03. 31.

[12] Multimédia : modulfüzet / Valentinyi András. – Tessedik Sámuel Főiskola, 2003. – 215 p. - http://cgip.inf.unideb.hu/multimedia.pdf. - Letöltés ideje: 2011. 04. 07.

[13] Web Akadálymentesítési Útmutató 2.0 : W3C Ajánlás 2008. december 11. / W3C, 2008. - http://www.w3c.hu/forditasok/WCAG20/. - Letöltés ideje: 2011. 04. 06.

[14] Rubik’s Cube Java Applet / Michael Schubart. – 1995. – http://www.schubart.net/rc. - Letöltés ideje: 2011. 04. 25.

[15] Simple PHP Guestbook / http://www.havia.net/guestbook. - Letöltés ideje: 2011. 04. 22.

Ábrajegyzék:

Köszönet nyilvánítás:

Szeretnék köszönetet mondani dr. Bujdosó Gyöngyi belső témavezetőmnek, a rengeteg segítségért, szakmai tanácsért, és a gyors válaszokért, valamint dr. habil. Boda Istvánnak és Iszály Györgynek, a Java programozási nyelvből való alapos felkészítéséért.

További köszönetet mondok a Debreceni Egyetem Matematikai és Informatikai Épület könyvtáros munkatársainak, segítségükért.