Weblapfejlesztés

Kvaszingerné Prantner Csilla – Nagy Dénes

MÉDIAINFORMATIKAI KIADVÁNYOK

Weblapfejlesztés

Kvaszingerné Prantner Csilla – Nagy Dénes

Eger, 2013

Korszerű információtechnológiai szakok magyarországi adaptációja

TÁMOP-4.1.2-A/1-11/1-2011-0021

Nemzeti Fejlesztési Ügynökség
www.ujszechenyiterv.gov.hu
06 40 638 638

MAGYARORSZÁG MEGÚJUL

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Lektorálta:
Nyugat-magyarországi Egyetem Regionális Pedagógiai Szolgáltató és Kutató Központ

Felelős kiadó: dr. Kis-Tóth Lajos
Készült: az Eszterházy Károly Főiskola nyomdájában, Egerben
Vezető: Kérészy László
Műszaki szerkesztő: Nagy Sándorné

Tartalom

1. BEVEZETÉS

1.1 CÉLKITŰZÉSEK, KOMPETENCIÁK A TANTÁRGY TELJESÍTÉSÉNEK FELTÉTELEI

1.1.1 Célkitűzés

A hallgatók ismerkedjenek meg a statikus weblapok készítésének **technikáival, szabványaival és szoftvereivel**. Ismerjék meg és tudják használni a HTML, az XHTML és a HTML5 leírónyelvet. Ismerjék meg a HTML leírónyelvek kialakulásának történetét, tudják azt, hogy mi vezetett el a szabványok kialakulásához. **Vegyék figyelembe a W3C ajánlásait és a webes szabványokat a tervezés és weblapfejlesztés folyamán**. Ismerjék a HTML alapvető és gyakran használt szerkezeteit és azokat adekvátan tudják alkalmazni a weblapfejlesztés során. Hatékonyan tudják a médiaelemeket kezelni HTML-ben és ismerjék meg a HTML5 médiakezeléssel kapcsolatos új technikákat. A weboldalak megjelenését CSS stílusfájlok alkalmazásával érjék el, ismerjék az alapvető CSS technikákat és azok alkalmazásait, képesek legyenek CSS layoutok kialakítására. Ismerjék a JavaScript működésének elvét, tudják milyen feladatok végezhetőek el a scriptnyelv segítségével, ismerjék ennek alapvető adattípusait és vezérlési szerkezeteinek pontos szintaxisát. Továbbá precízen tudják kezelni azokat a szoftvereket és rendszereket, melyek a weblapfejlesztés egyes feladatkörinek elvégzéséhez szükségesek. Előfeltétel az, hogy gyakorlottan kezeljék a leggyakrabban használt különböző cégek által készített böngészőket, ismerjenek többféle editort és weblapfejlesztő-szoftvert, tudjanak alapvető feladatokat elvégezni különböző médiaszerkesztő (kép-, hang- és videó-szerkesztő) programokkal, továbbá tudjanak biztosan kezelni egy FTP-zésre is alkalmas fájlkezelőt. Értsék meg azt, hogy miért fontos egy weboldal **tartalmának, működésének és megjelenítésének az elkülönítése,** tudják e három tényezőt mind elméletben, mind gyakorlatban – azaz a kódolás szintjén is – külön kezelni. Képesek legyenek e külön kezelt három összetevőt egy összefüggő rendszerbe kovácsolni, azaz egy egységes weboldalba integrálni, közöttük a kapcsolatot megteremteni úgy, hogy egy könnyen karbantartható, módosítható és fenntartható weboldalt kapjanak eredményül.

1.1.2 Kompetenciák

A kurzus során számos kompetenciát kell elsajátítaniuk a hallgatóknak mind a tudás, az attitűdök és a képességek területén. A kompetenciák elsajátítása révén a diákok képessé válnak a célkitűzések elérésére.

A hallgató a **tudás** terén rendelkezzen a rendszerben való gondolkodás képességével annak érdekében, hogy képes legyen jól működő weboldalak megtervezésére. Előrelátással, lényegkiemeléssel és problémamegoldó képességgel kell rendelkeznie ahhoz, hogy könnyen frissíthető, módosítható és karbantartható weboldalakat valósítson meg. A weblapfejlesztéshez feltétlenül rendelkeznie kell hierarchikus gondolkodással, mely az oldalak kapcsolatának a megtervezéséhez, a site-struktúra megalkotásához és a felületen megjelenő navigációk kialakításához szükséges. A szakma gyors változásának követése érdekében fejlődőképességre, önfejlesztésre, továbbá általános hallgatói képességekre van szükség. Mindezeken felül fontos, hogy rendelkezzen a hallgató gyakorlatias feladatértelmezési és az igények felmérésére való képességgel egyaránt.

Az **attitűdök és a nézetek** terén a hallgató megfelelő felelősségtudattal rendelkezzen a tervezési folyamatok véghezvitele céljából. A hallgató megbízható és precíz legyen a működőképes és használható weboldalak létrehozása érdekében. A weblapfejlesztői munka nagyfokú együttműködési képességet is feltételez, hiszen a fejlesztés leggyakrabban team-munka keretében zajlik. A weblapfejlesztésben részt vevő szakemberek esetében fontos, hogy képesek legyenek az együttműködésre, a toleranciára, a rugalmasságra és a pontos fogalmazókésségre. Továbbá fontos, hogy a leendő szakember legyen nyitott a weblapfejlesztés újabb technikai megvalósításai irányában, hogy a készítendő website-ok és portálok működésükben megfeleljenek a korszerű weblapokkal szemben állított követelményeknek.

A **képességek** terén a hallgatók tudjanak önállóan elkészíteni egy összefüggő, multimédiás elemeket is tartalmazó weboldalt. Megbízhatóan és kreatívan legyenek képesek olyan weboldalak megtervezésére és megalkotására, amelyek megfelelnek a szoftverminőségi faktor elvárásainak.

1.1.3 A tantárgy teljesítésének feltételei

Az elméleti ismereteket magába foglaló feladatlap eredményes kitöltése.

Gyakorlati weblapfejlesztői feladat eredményes megoldása.

A megadott témák valamelyikében egy komplex működőképes weblapfelület elkészítése.

1.2 A KURZUS TARTALMA

Témakörök:

1. A Weblapfejlesztés eszközei

2. Webes szabványok és a W3C
3. A HTML alapjai
4. Gyakran használt HTML szerkezetek
5. Médiaelemek kezelése a HTML-ben
6. A CSS alapjai
7. Gyakori CSS technikák
8. Layout készítés CSS-sel
9. A JavaScript alapjai
10. Weblapfejlesztés a gyakorlatban

Az első két témakörben szó esik a weblapfejlesztéssel kapcsolatos alapvető tudnivalókról és nélkülözhetetlen háttérinformációkról, a webes szabványokról és a W3C által megfogalmazott ajánlásokról. Fény derül arra, hogy miért és hogyan érdemes szétválasztani a weblapfejlesztés során a tartalmat, a működést és a megjelenítést.

A következő három témakör az XHTML leírónyelvvel foglalkozik részletesen. Ebben az egységben megfogalmazásra kerülnek a HTML és XHTML közötti különbségek, alkalmazásuk és az XHTML validálása. Bemutatásra kerülnek a leggyakrabban használt XHTML szerkezetek és a médiaelemek megjelenítésének parancsai, módszerei és trükkjei, továbbá szó esik a weboldalakon használatos hang- és video-formátumokról.

Az XHTML fejezetek után egy újabb három fejezetből álló nagyobb blokk következik, amely a CSS használatáról, szintaxisáról, szövegtulajdonságokról, a színek CSS-ben való megadásáról és a szabályos CSS kódról szólnak. Még ebben a blokkban találkozhatunk a gyakori CSS technikák bemutatásával és a CSS-sel történő layout-készítés módszerével. A CSS részletes ismertetése után egy fejezet keretén belül bemutatásra kerülnek a JavaScript alapjai, adattípusai és az általa használt vezérlési szerkezetek. Végül, annak érdekében, hogy a hallgató teljes egészében lásson egy weblapfejlesztési folyamatot: a tananyag utolsó témaköreként egy konkrét weboldal elkészítésének a folyamata kerül lépésről-lépésre bemutatásra.

1.3 TANULÁSI TANÁCSOK, TUDNIVALÓK

Szerencsés, ha a tanuló a megadott sorrendben dolgozza fel az egyes leckék anyagát, hiszen az egyes leckék tartalmai egymásra épülnek. A tananyag

számos kódrészletet tartalmaz, sok esetben kódokhoz tartozó képernyőképek jelennek meg, amelyek a szövegben leírt elméleti anyag megértését segítik. Számos példa található a tananyagban, melyek egy-egy adott problémafelvetést vagy leírt elvet/módszert példáznak, az érthetőség elősegítése érdekében tehát érdemes alaposan áttekinteni őket.

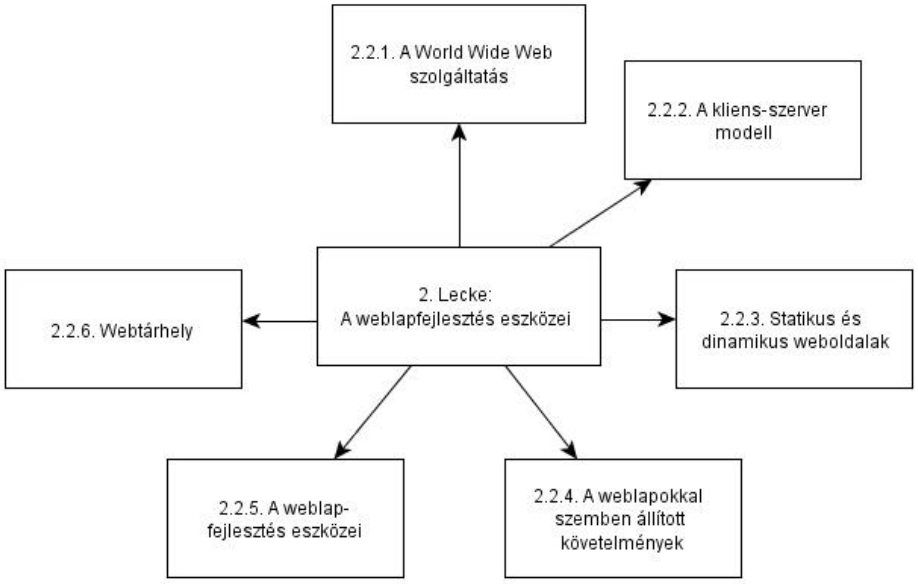
A leckében a fogalmak és definíciók bekezdései előtt egy fektetett, nyitott könyv képe látható kicsiben, a példák bekezdései előtt pedig egy állított kisméretű lap képe van, a szövegrészek a példák esetében kurzívan szedettek, a kódok a könnyebb átláthatóság és tagolhatóság érdekében Courier New betűtípussal lettek a tananyagban megjelenítve; ez utóbbiak halványszürke háttere segít a kódok kiemelésében. A tananyagban keretbe foglalva hasznos tippek és trükkök találhatóak.

2. A WEBLAPFEJLESZTÉS ESZKÖZEI

2.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Cél, hogy a hallgató ízelítőt kapjon a weblapok kialakulásáról, a weboldalak letöltődésének menetéről, átlássa a weblapfejlesztéshez szükséges eszközök tárházát, tudja mely eszközzel milyen folyamatok oldhatóak meg és képes legyen az eszközök adekvát alkalmazására. A lecke az eszközök bemutatása előtt egy rövid áttekintést tartalmaz a weblapok világáról, utána pedig egy rövid áttekintést a fejlesztés során elvégzendő tevékenységekről.

2.2 TANANYAG



1. ábra: A leckéhez tartozó fogalomtérkép

2.2.1 A World Wide Web szolgáltatás

A World Wide Web (röviden www vagy web) egy olyan szolgáltatás az interneten, amely gyakorlatilag egy nagyméretű összefonódó dokumentumrendszer. Ennek a dokumentumrendszernek alapvető eleme a weblap, ami nem más, mint egy HTML kiterjesztésű elektronikus dokumentum. A World Wide Web alapelveit 1989-ben Genfben, **Tim Berners Lee** a **CERN** részecskefizikai kuta-

tóközpont munkatársa fektette le. A cél az volt, hogy a kutatók könnyen és olcsón meg tudják osztani egymással publikációikat és kutatási eredményeiket, azaz egy-egy kiadott tudományos cikk a világ bármely pontján elérhetővé váljon az internethálózatnak köszönhetően.

A böngészőprogramok segítségével jeleníthető meg a weboldalak tartalma. A weboldalak úgynevezett hipermédiás dokumentumok, amelyek láthatóan szövegeket, képeket és egyéb multimédia-elemeket, továbbá linkeket tartalmaznak. A HTML dokumentumok viszont csak szövegeket tartalmazó fájlok és nem foglalják magukba a képeket, a grafikákat, mint például egy jól szerkesztett szöveges dokumentum vagy a hangokat és videoelemeket, mint például egy prezentáció. A weboldalak csak a médiaelemek **elérésének útját** (ún. path-ját) tartalmazzák, a médiaelemek a HTML dokumentum mellett a háttértáron, egy könyvtárban foglalnak helyet, és a webböngészők azok, amelyek képesek a HTML kód értelmezésére, és a benne megadott útvonal segítségével az adott médiaelemek megfelelő helyen való megjelenítésére. A HTML dokumentumban találhatóak a weboldalakon megjelenő szövegek, a médiaelemekhez vezető hivatkozások és utasítások arra, hogy mindez milyen formában és elrendezésben jelenjen meg; a weblapok szövege és médiaelemei tehát a böngészőprogramokban állnak össze egy egésszé.

2.2.2 A kliens-szerver modell

A www szolgáltatás három szabványra épül.

1. Az **URL (Uniform Resource Locator)** írja le, hogy milyen egyedi címmel rendelkezzenek az egyes weboldalak.

2. A **HTTP (Hyper Text Transfer Protocol)** egy információátviteli protokoll, amely megadja, hogy a kliens és a szerver között hogyan történjen az információtovábbítás.

3. A **HTML (Hyper Text Markup Language)**, a könnyen tárolható és továbbítható információkódolás módja, egy leírónyelv, melynek segítségével az adott információk sokféle eszközön könnyen megjeleníthetővé válnak, ez a négy karakter adja a weboldalak állományainak kiterjesztését.

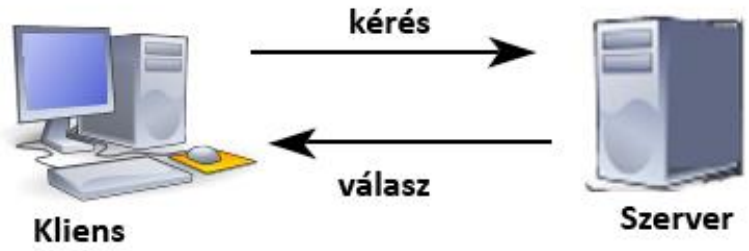
A weboldalak megtekintésének folyamata egy ún. **kliens-szerver modell** szerint zajlik. Ennek működése úgy fest, hogy mi felhasználók a számítógépünk böngészőjébe (kliens) beírjuk a megnézendő weboldal URL címét, melynek hatására a számítógépünk egy kérést küld annak a webszervernek (szerver), amelyről a szükséges weboldal információit le szeretnénk kérni; majd a szerveren futtatott szoftver a kérésünkre elküldi a megfelelő weboldalt, amelyet megjelenít a gépünkön lévő böngészőprogram (azaz a kliens). Az, hogy melyik webszerver felé küldje számítógépünk a kérelmet, az kiderül az URL címből.

Tehát ha egy weboldalt meg szeretnénk nézni a böngészőnkben, akkor csak be kell írjuk annak URL címét a címsorba, minden egyébről a HTTP protokoll gondoskodik.

Az URL cím egy olyan összetett szabványosított cím, amelyben benne van a protokollnak a neve, amit a kommunikációhoz használunk (pl.: http); benne van annak a szervernek vagy tartománynak a neve, amivel kommunikálni szeretnénk (pl. ahol a megnézni kívánt weboldal van); benne van annak a portnak a száma, amin a szolgáltatás elérhető és még az adott weboldal kiszolgálón (szerveren) lévő elérési útja is.

A böngésző címsorába beírandó URL egy alfanumerikus (azaz betűket, számokat illetve speciális karaktert tartalmazó) cím, melynek egy része már eleve annak a szervernek a címe, amelyen található a weblap.

Az internet egységes címzési rendszert használ a számítógépek elérésére, amit **IP (Internet Protokoll)** címzésnek hívunk. Minden egyes internethálózatban részt vevő számítógépnek egyedi IP címe van, a szerver számítógépeknek az IP-jük állandó. A számítógépek az IP címeknek a segítségével azonosítják be egymást a hálózaton belül, de mi emberek nyilvánvalóan azért a webcímeket használjuk, mert könnyebben meg tudjuk jegyezni a szavakat a hosszú számsoroknál. A webcímeket így IP címekké kell alakítani, amit egy ún. **DNS (Domain Name System)** végez el. Ezt úgy kell elképzelni, mint egy címtárat ahol megvan minden internetre csatlakoztatott eszköz elérhetősége. A DNS rendszertől a böngésző megkapja a hivatkozott oldal címét, és egy kérést küld a cím által jelölt géphez, majd várakozik az onnan érkező válaszra. Ha minden megfelelőképpen működik, a szerver visszaküld egy választ, amely először egy visszaigazolás arról, hogy minden rendben történt, majd küldi magát a weboldalt is, ezt egy **HTTP fejléc** tartalmazza. Ha bármilyen probléma van a kérés/válasz során akkor egy **HTTP hibát** kap a böngésző, ezzel szokott találkozni a felhasználó is, ha nem található az adott weboldal. Érdemes még megfigyelni az URL-ek esetében, hogy ha egy oldal kezdőlapját nyitjuk meg, akkor nem látunk fájlnevet utána. Előfordulhat, hogy az oldal további mélységében sem találunk fájlnevet, amire a cím hivatkozik. Ez azért van így, mert a szerveren futó kiszolgálószoftver kezeli a címeket és az rendeli hozzá a megfelelő állományokat, így annak ellenére, hogy nem látunk hivatkozást fájlra, mindig egy HTML állományt látunk. Az URL címek beírása esetében elhagyható a http:// kifejezés a cím elejéről, sok weblap esetében a www is.



2. ábra: A kliens-szerver modell

2.2.3 Statikus és dinamikus weboldalak

A weboldalakat statikus és dinamikus weboldalak csoportjaira szokás osztani.

A **statikus weboldalak** olyan weboldalak, amelyek tulajdonképpen teljesen változatlanok maradnak, azaz ahogyan a készítőjük elhelyezte őket a webszerveren, abban az állapotban maradnak meg és minden lekérdezésnél ugyanazzal a tartalommal és ugyanabban a formában jelennek meg. Ezek egyszerű HTML oldalak. Ugyanazt a HTML kódot tölti fel a weboldal készítője a szerverre, ami letöltődik majd a kliens számítógépekre. Ezek általában egyszerűbb kinézetű és felépítésű oldalak. A statikus oldalak onnan ismerhetőek fel, hogy nem tartalmaznak semmiféle programozást igénylő funkciót, mint például regisztrációt, fórumot, blogot, webshopot, nem kérnek be adatokat a felhasználótól, így nem is tárolnak el adatbázisban semmilyen információt. Ezek a típusú weboldalak zömmel céges bemutatkozó oldalak.

Abban az esetben készítenek **dinamikus weboldalakat**, amikor változó tartalomra van szükség. Ezek az oldalak bemeneteket képesek fogadni és a megjelenő tartalmak jelentős része adatbázisból kerül az oldalakra. A weboldal készítője a dinamikus weboldalak esetében kész HTML helyett **programkódot tölt fel** a szerverre (leggyakrabban PHP vagy ASP nyelven írtat), amelynek szerveren lévő futtatása közben generálódnak a HTML oldalak a felhasználók kliens gépeire. A weboldalra látogatók tehát ugyanúgy csak egy HTML oldalt látnak, sőt, csak a generált HTML-ek kódját tudják megtekinteni a böngészőjükben, a szerver oldalon lévő programkódot nem. Honnan ismerhetőek fel a dinamikus weboldalak? Gondoljunk arra, mikor egy weboldalra bejelentkezik egy felhasz-

náló az accountjával, ekkor mindig az adott felhasználó adatai töltődnek le a böngészőbe; azaz változik a tartalom a felhasználótól függően. A blogok, fórumok oldalai is dinamikus oldalak, továbbá azok az oldalak, ahová egyáltalán regisztrálni lehet, azaz adatokat adhatunk meg, amelyek tárolásra kerülnek egy adatbázisban, a levelezést megvalósító oldalak, a közösségi weboldalak, vagy például amelyek szavazást tartalmaznak, vagy vásárolni lehet rajtuk – mindmind dinamikus oldalak. A gyakran frissülő híroldalak is dinamikusak, csak azok a felületek, ahonnan az újságírók a cikkeiket feltöltik nem nyilvánosak számunkra, de abban az esetben is adatbázisba kerülnek az adatok és adatbázisból kerülnek ki a friss hírek az oldalakra.

Összegezve: a különbség a statikus és dinamikus weboldalak esetében tehát szerveroldalon mutatkozik meg, a szerver számítógépen statikus oldalak esetében ugyanaz a kód van tárolva, mint ami megjelenik a kliens gépeken, dinamikus weblap esetében viszont más a szerveren tárolt és a kliensen megjelenített kód.

A tananyagban kizárólag statikus weboldalak készítésével fogunk foglalkozni.

2.2.4 A weblapokkal szemben állított követelmények

A weblapfejlesztés legfőbb célja, hogy egy olyan weboldalt készítsünk, amely amellett, hogy *értékes információtartalommal bír*, *vonzó és igényes megjelenésű,* még **jól használható** is legyen. A használhatóság (usability) fogalmához sok követelménynek kell megvalósulnia egy weboldalon, melyek az alábbiakban olvashatók:

1. gyors és könnyű információszerzés;
2. jól megfogalmazott szövegek az oldalon (helyes nyelvhasználat);
3. olvashatóság;
4. válogatott, szemléletes médiaelemek;
5. a site tartalmi áttekinthetősége;
6. átlátható és logikus oldalelrendezések;
7. logikus site-struktúra;
8. egyértelmű navigáció;
9. könnyen kezelhető, felhasználóbarát felület.

Nem elég viszont csak az, hogy megalkotunk egy weboldalt, ami szép, működőképes és még a felhasználói igényeknek is eleget tesz, hanem azt is biztosítani kell a weblapfejlesztőknek, hogy az a későbbiekben:

10. *könnyen bővíthető, módosítható legyen;*
11. *különböző böngészőkben és felbontásban egyaránt jól jelenjen meg és*
12. *különböző eszközökön jól jelenjen meg.*

A weblapkészítés nem csak abból áll, hogy egy weboldalt előállítunk, ami jól működik; hanem az is fontos, hogy olyan weboldalt állítsunk elő, amely a későbbiekben

- könnyen karbantartható,
- könnyen bővíthető, módosítható és
- könnyen fenntartható, azaz könnyedén elvégezhetőek rajta a napi frissítések.

Emiatt kell mindig okosan és átgondoltan felépíteni oldalainkat a kódolás szintjén is. Ha a későbbi bővíthetőségre és fenntarthatóságra nem gondolunk, akkor egy „halott” weboldalt fogunk üzemeltetni, aminek nincs változó tartalma, emiatt nem is igazán látogatott.

2.2.5 A weblapfejlesztés eszközei

A weblapfejlesztés egy igen összetett folyamat, mely számos, jól elkülöníthető részfolyamatra osztható, az egyes részfolyamatok végrehajtásához más-más eszközökre, más-más jellegű szoftverekre van szükség. A weblapfejlesztéshez szükséges elengedhetetlen eszközök az alábbi listában összegyűjtve találhatóak meg:

1. Böngészők;
2. Editor vagy weblapszerkesztő programok;
3. Médiaelemek előállítására szolgáló programok;
4. Fájlkezelők.

Weblapjainkat a böngészőprogramok segítségével tudják megtekinteni a felhasználók. Editor vagy weblapszerkesztő program a HTML és CSS állományok elkészítéséhez szükségesek. A képeket, a hanganyagokat és a videókat kép-, hang-, és videoszerkesztő-szoftverek segítségével kell előkészíteni a weboldalon való megjelenítéshez. Ahhoz, hogy az elkészített weboldal a világ bármely pontján, bármely internethez csatlakozó számítógépen elérhető legyen, annak minden elemét egy webszerverre kell feltölteni, melyhez egy olyan fájlkezelő szoftverre van szükségünk, mellyel az adatok átvitele egy távoli szerverre megoldható.

1. Böngésző

Böngészőnek (angolul browser) azokat a programokat nevezzük, amelyek segítségével az interneten található tartalmakat, túlnyomórészt weblapokat lehet megtekinteni, vagy azok valamilyen internetes szolgáltatását használni. A webböngészők HTTP protokollt használnak, amelynek segítségével kéréseket küldenek a szervernek, majd az válaszként elküldi a kért weblapot és a böngésző megjeleníti azt. Általában képes más protokoll kezelésére is, mint például a HTTPS vagy az FTP. A HTML állományok mellett a böngészők sok mást is támogathatnak, mint a weben használt képformátumokat, sokuk már a saját ablakukban nyitják meg például a .pdf dokumentumokat. Manapság szinte az összes böngészőhöz találhatunk kiegészítéseket, ezzel együtt fel is készíthetjük további állományok kezelésére is, ilyen lehet a Flash .swf formátuma vagy a különféle XML alapú megoldások is. Két nagy csoportba sorolhatjuk a böngészőket: a karakteres, és a grafikus típusba. A karakteres böngészőkkel nem sűrűn találkoznak a felhasználók, ezek konzolos felületen futnak, ahol csak karakterek megjelenítésére van lehetőség. A grafikus böngészők támogatják a multimédiás megvalósításokat, a Flash animációk, a hangok és videók lejátszását, valamint a képek megjelenítését. Emellett gyakran találkozhatunk különböző internetes szoftverek beépülő moduljaival is. Általános elvárás még a böngészővel szemben az, hogy egységesen kezeljék a szabványokat. Sokféle böngésző létezik manapság, és szerencsére egyre előrébb járnak a szabványkövetésben.

Tudnunk kell, hogy a Flash állományok weblapokon való megjelenése napjainkban – a HTML5 elterjedésével párhuzamosan – egyre inkább a háttérbe szorulnak, hiszen a mobiltelefonon internetezők nem tudják azokat megnézni, az Apple termékek használói pedig szintén nem, mert a cég nem támogatja a formátumot, így manapság már egyre inkább csak reklámok és webes játékok esetében használnak Flash animációkat és Flash videót.

A böngészőprogramokkal tudják tehát a felhasználók megtekinteni az interneten elérhető weboldalainkat; és a böngészők segítségével tudják a webfejlesztők is megnézni, hogy az általuk készített weboldalakat hogyan látják a felhasználók majd a saját gépükön, a saját böngészőjükben. Természetesen a felhasználók más-más böngészőt használnak, így weblapunkat az összes, aktuális időszakban használatos böngészőben le kell tesztelnünk, azaz a megjelenítést megtekintenünk és a működést kipróbálnunk.

A mai legismertebb, rendelkezésre álló webes böngészők a következők: Mozzilla Firefox, Safari, Opera, Google Chrome és az Internet Explorer.

3. ábra: A legnépszerűbb böngészők ma

A böngészők nagy harcot vívtak és vívnak ma is a felhasználókért. Minden böngésző más-más területen erős.

Egy weboldal elkészítése után legalább a felsorolt öt böngészőben meg kell tekintenünk oldalunk megjelenését, mielőtt azt egy webszerverre feltöltenénk, mindezt annak érdekében kell tennünk, hogy megbizonyosodjunk a helyes és egységes megjelenésről. A böngészőknek egy nagyszerű, részletes összehasonlítása található az interneten[1].

2. Karakteres editorok és grafikus weblapszerkesztő programok

Ahhoz, hogy a weboldalakat megalkossuk, szükségünk van valamilyen programra, amelyikben létre tudjuk hozni a HTML kódokat. Erre a célra kétféle típusú program létezik: vannak **karakteres editorok**, melyek segítségével könnyedén begépelhetőek a kódok, és léteznek **grafikus weboldalszerkesztő programok**, amelyekben WYSIWYG felületen egérkattintásokkal összerakhatóak a weboldalak. Ez utóbbi programok lehetőséget nyújtanak a kódban módosításra is.

Egyszerű kódok bevitelére vagy módosítására még a Windows Jegyzettömbje (Notepad) is alkalmas, ha éppen nem áll rendelkezésünkre egy komolyabb editor vagy nincs jogosultságunk az adott gépre telepíteni, bár kétségtelenül nem a Jegyzettömb a legjobb választás. Ha a Jegyzettömbnél jobb eszköz mégsem áll a rendelkezésünkre, akkor figyelnünk kell arra, hogy a HTML kódokat UTF-8 kódolásban mentsük el az alapértelmezett ANSI helyett.

Nagyon jó editorok állnak a rendelkezésünkre manapság, mint pl. a Windowshoz fejlesztett TextPad[2] vagy az EditPlus[3]. Kiváló editor az Oxygene[4] és a Crimson Editor[5] egyaránt, melyeket weblapfejlesztők használnak, az editorok

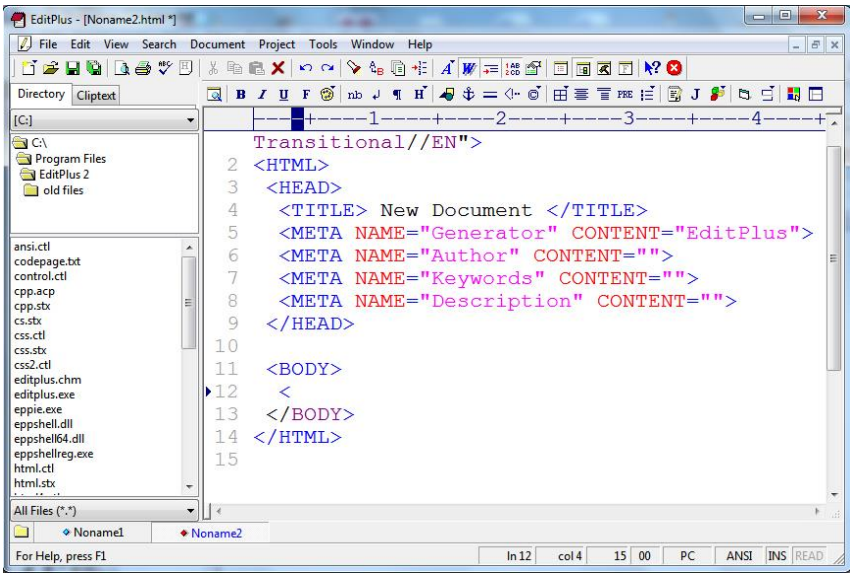
[1] http://www.windowsmix.com/_nw/3/17276510.jpg
[2] http://www.textpad.com/
[3] http://www.editplus.com/
[4] http://www.oxygenxml.com/
[5] http://www.crimsoneditor.com/

sokat segítenek a fejlesztőknek. Nagyon jó eszköz a Notepad++[6], amely ingyenes elérhető program. A weboldalak szerkesztéséhez nagyon ajánlott megbízható, jó eszköz, és létezik magyar nyelvű fordítása is.

A mintában az EditPlus program képe látható. Új, üres HTML oldal létrehozása esetén például már a HTML dokumentum kódjának szerkezetét kapjuk meg a lapon, ennyivel is kevesebb kódot kell nekünk begépelni. A kód egyes elemei, úgy, mint:

- a tagek,
- a paraméterek,
- a paraméterek értékei,
- a kommentek és
- a böngészőben megjelenő szövegek karakterei

mind-mind különböző színekkel vannak megjelenítve. A sorok számozva, a programok által beszúrt kódrészletek eleve strukturálva jelennek meg a lapon, így a kód igen könnyen olvasható és jól áttekinthető. Az EditPlusban egyes ikonok kiválasztásával például HTML kód részletek szúrhatóak be, mint például kép vagy táblázat esetében – ezzel a lehetőséggel is gépelés alól mentesülünk.



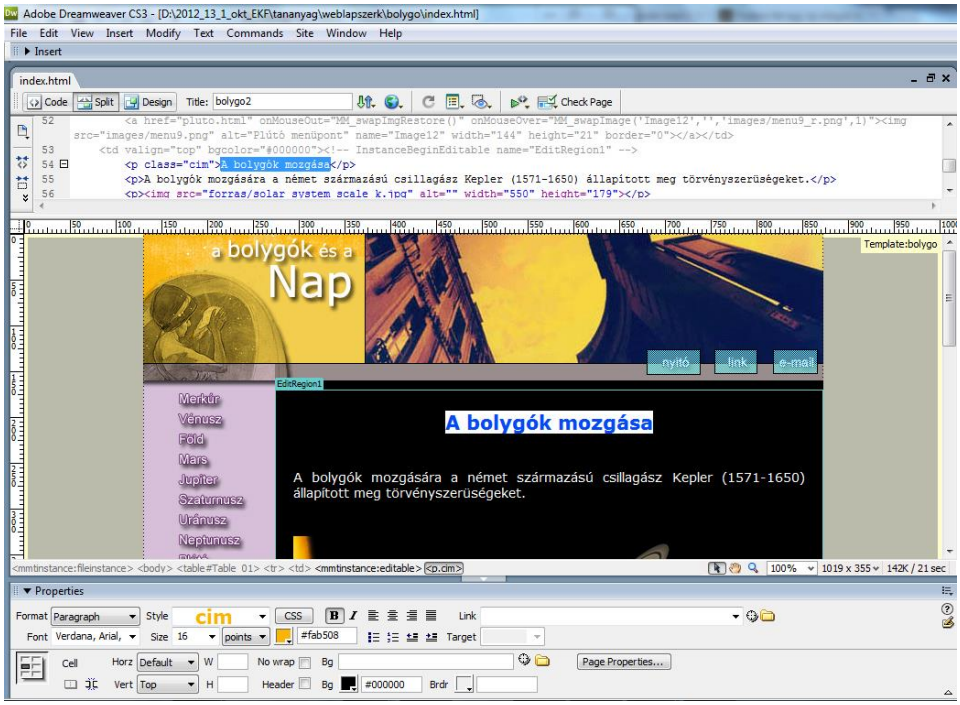
4. ábra: Az Edit Plus képe új HTML dokumentum létrehozása esetén szintaxiskiemeléssel

[6] http://notepad-plus-plus.org/

A legtöbb karakteres editor esetében adott az a lehetőség is, hogy a gépelni kezdett parancsok és paraméterek listáját kidobja egy legördülő listában.

A grafikus weblapszerkesztő programok igen népszerűek a kezdő weblapfejlesztők körében; használatuk kényelmes és könnyű velük egyszerű, statikus weboldalakat készíteni. A megfelelő menüpontok és/vagy ikonok kiválasztásával és az egyes paneleken a paraméterek beállításával gyakorlatilag felépíthető egy teljes weboldal a grafikus weblapfejlesztő programmal. Viszont ezen programok használatánál is nagy előny az, ha a weboldal készítője ismeri a HTML leírónyelvet, a CSS és a JavaScript kódokat alapszinten. Ez azért szükséges, mert minden grafikus weblapfejlesztő program más és más, ráadásul nem létezik olyan, ami teljesen kifogástalanul működik minden esetben, azaz minden utasításra pontosan megfelelő kódot generál a háttérben. Ezekre az esetekre például rendkívül jó, ha az ember bele tud nyúlni a HTML kódba és javítani tudja a program hibásan generált kódját, bár nem jellemző, hogy javítani kell a kódon, de azért előfordul.

A grafikus weblapfejlesztő programoknak tehát az a nagy előnyük a karakteres editorokkal szemben, hogy mind grafikus, mind kódnézetben egyaránt készíthető a weblap. A Dreamweaverben például könnyen válthatunk az ún. *Design* és *Code* nézetek között. Emellett létezik egy harmadik nézet is, a *Split* (osztott) nevű, amelynek köszönhetően mindkettő egyaránt látható és bármelyik nézetben kijelölünk egy részt, az automatikusan a másikban is kijelölve jelenik meg, így igazán könnyű akódban a hibakeresés. Az alábbi képen látható a Dreamweaver osztott nézetére egy jó példa.



5. ábra: Az Adobe Master Collection CS3 programcsomag Dreamweaver CS3 grafikus weblaptervezőjének felülete

Jó eszköz a Microsoft Share Point Designer 2007-es verziója is, mely teljesen ingyenesen letölthető formában található az interneten.[7] Általában mindkét weblapfejlesztő programtípus, azaz a karakteres és a grafikus egyaránt képes a site könyvtárának állományait beforgatni és a programon belül kezelni.

3. Médiaelemeket előállító programok

A weblapon megjelenő képek, hangok és videók jelentősen meghatározzák oldalunk megjelenését és összképét; ahhoz, hogy színvonalas weboldalak kerüljenek ki a kezünk alól elengedhetetlen, hogy igényes médiaelemeket helyezzünk el az oldalainkon. Számos program létezik, melyekkel mindezek előállíthatóak és megszerkeszthetőek.

[7] http://www.microsoft.com/hu-hu/download/details.aspx?id=21581

a. Képek szerkesztése

A képszerkesztő programokat két nagy kategóriába soroljuk: **vektor- és pixelgrafikus**[8] **szoftverek** csoportjába. Az alapvető különbség az, hogy míg a pixelgrafikus képek képpontokból épülnek fel, ahol a kép minden egyes képpontjának adott a színkódja; addig a vektorgrafikus képeket matematikailag leírható pontok, vonalak sokszögek és görbék építik fel, ezek függvényeit és képleteit tárolják a vektorgrafikus képek, továbbá felületük kitöltésével oldják meg a színezést és mintázatot.

Fontos tudni, hogy míg a pixelgrafikus képek nagyítása minőségromlással jár, addig a vektorgrafikus képeké nem. A weboldalakon megjelenő fotók biztosan pixelgrafikus képek, de számos kép vagy rajz lehet, ami vektoros. A weboldalak logói (emblémái) rendszerint vektorgrafikus képszerkesztőkkel készülnek, hiszen a logóknak nagyon különböző méretekben kell megjelenniük: ugyanolyan jól kell kinézniük egy tollra szitázva 0,5 cm x 0,5 cm-es méretben, mint egy multiposzterre nyomtatva 3 m x 3 m-esben. Az emblémákon túl még számos esetben találkozhatunk weblapfejlesztés során vektoros képekkel, így a pixelgrafikus programok mellett a vektorgrafikusok ismerete is fontos. Vannak programok, amelyek mindkét típusú képeket képesek kezelni.

Az **Adobe Photoshop** az egyik legnépszerűbb és legelterjedtebb pixelgrafikus képszerkesztő és képmanipuláló szoftver, amelynek óriási a támogatottsága az interneten: számos leírás, segédanyag és ingyenes tutorial érhető el hozzá. Érdemes az angol verziókat használni, ugyanis a segédletek zöme angol nyelvű, de a magyar nyelvű anyagokban is gyakran angol nyelvű programot mutatnak be. A program egyaránt fut Windows és OS X (korábban Mac OS X) operációs rendszerek alatt. A CS3-as (Creative Suite, 10.0, 2007) verzió óta létezik Standard és Extended változata, ez utóbbi egy bővített csomag, amelyben a 3D modellezéshez és a mozgóképek szerkesztéséhez tartalmaz kiegészítő eszközöket. Az internetről ingyenesen letölthetőek az egyes verziószámok harminc napos **trial**[9] verziója, ahhoz hogy ki tudjuk próbálni a szoftvert. Elérhetőek **portable** (hordozható) és interneten használható változatok egyaránt, ez utóbbi használatához csatlakozni kell az Adobe Creative Cloud-hoz és így az interneten keresztül érhető el a szoftver 30 napon keresztül, ebben az a jó, hogy nem a mi háttértárunkon foglalja a helyet a program és bárhonnan elérhető a felület,

[8] A pixelgrafikus képekre a raszteres képek kifejezést is szokás használni.

[9] Harminc napon keresztül ingyenesen korlátlan használat, a shareware programok esetében szokták ezeket általában harminc napon keresztül ingyenesen korlátlan használatra bocsátani kipróbálás céljából.

ahol van internet hozzáférésünk. Új verziók esetén gyakran elérhetővé teszik a **béta**[10] változatokat is.

A Photoshopon kívül számos más programot használhatunk a képeink előállítására, megrajzolására, módosítására vagy fotóink retusálására. Nézzünk meg két népszerű pixelgrafikus programot a képszerkesztés területén: **Paint.NET és Gimp** (GNU Image Manipulation Program) Bármelyik használata ajánlott, amennyiben nem rendelkezünk Photoshoppal, mindkettő **nyílt forráskódú, free sofrware**, ahol a „free" szó nem arra utal, hogy ingyenesek, hanem arra, hogy szabadon használható programok, magyarul szabad szoftvereknek[11] szokás ezeket nevezni.

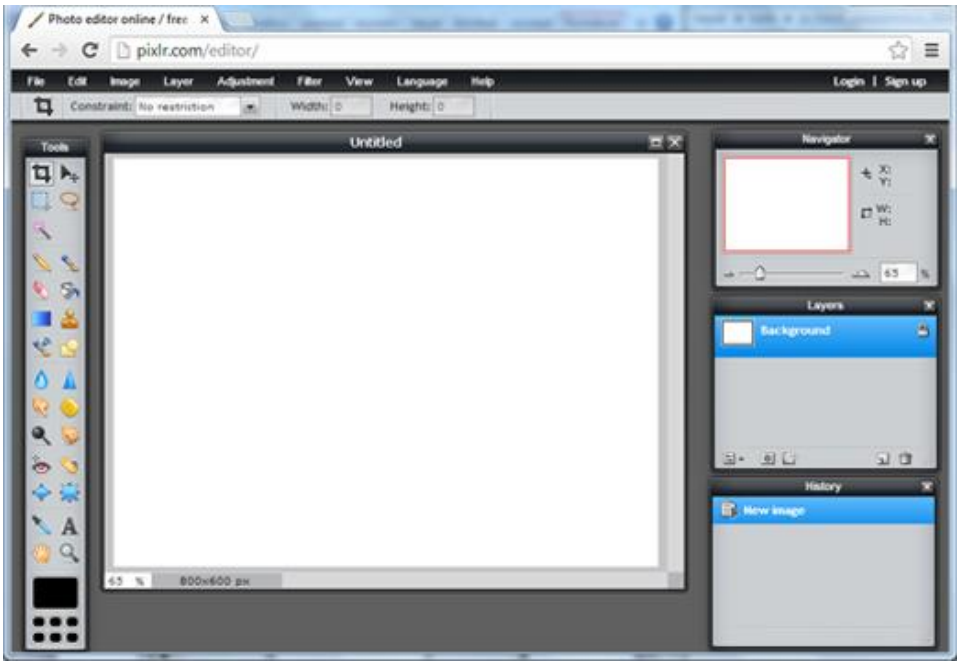
A Gimp előnye az, hogy platformfüggetlen és van némi vektorgrafikus támogatás beépítve, továbbá széles felhasználói körrel rendelkezik és több nyelven, ingyenesen elérhető dokumentáció tölthető le a hivatalos oldalról. A Paint.NET-et azoknak a Windowsos környezetben dolgozó felhasználóknak ajánlanám, akiknek gyorsan és egyszerűen telepíthető, kis helyet igénylő, ám széles tevékenységi kört ellátó képszerkesztő szoftverre van szükségük. Nagyon ügyes kis program, a letöltés mérete mindösszesen 3,5 MB. Támogatja a rétegek használatát és az átlátszóság tulajdonságot a képszerkesztés során.

A fotómanipulálás terén népszerű program a **Picasa,** amely fotók kezelésére, rendezésére, módosítására és retusálására használható **freeware**[12], azaz ingyen használható számítógépes program. Fotók esetében sokszor kívánatos a képeknek a fényerején, kontrasztján és színegyensúlyán változtatni. A Paint.NET és a Picasa együttesen elegendő eszközt jelent egyszerűbb webes felületekre szánt képek és fotók előmunkálatára, előállítására és módosítására. Van egy nagyon jó interneten elérhető képszerkesztő szolgáltatás: pixlr.com. Ezen belül a Pixlr Editort kell választani és máris használható a „felhőben" lévő képszerkesztő.

[10] A nagyközönség számára, tesztelés céljából ingyenesen elérhetővé tett még nem végleges verzió.

[11] A „szabad szoftver" elnevezés a felhasználók szabadságára utal. Azt jelenti, hogy a felhasználóknak szabad futtatni, másolni, közzétenni, tanulmányozni, megváltoztatni és tökéletesíteni a szoftvert. Pontosabban kifejtve a felhasználók négy különböző jogát jelöli: Szabad licencű szoftvereknek is szokás hívni. Bővebben: http://www.gnu.org/philosophy/free-sw.hu.html, 2012.

[12] Ingyen használható, tetszőlegesen hosszú ideig. Bővebben: http://hu.wikipedia.org/wiki/Freeware, 2012.



6. ábra: A pixlr.com felhőszolgáltatásképszerkesztő editorának felülete

Az **ACDSee** és a **Snagit** szoftverek nem képszerkesztő programok, ám hasznosak lehetnek a médiaelemek megtekintése, rendezése, módosítása és előállítása során. Az ACDSee elsősorban egy képnézegető szoftver, mely a képek kötegelt feldolgozására is alkalmas, emellett az alapvető képmanipulációs tevékenységek könnyen elvégezhetőek vele: a képeket át lehet méretezni, el lehet forgatni, lehet módosítani a kontraszton, fényerőn, élességen és többek között ebben a programban is található piros szemet eltávolító funkció.

A Snagit program elsősorban egy képernyőlopó eszköz, mellyel tetszés szerint leszedhető a teljes képernyő tartalma vagy annak egy részlete, vagy egy folyamatvégzés videója. Mindkettő harmincnapos trial változata letölthető az internetről.

A pixelgrafikus és vektorgrafikus szoftverek összefoglalásaként látható az alábbiakban egy táblázat.

Program neve	Program emblémája	Elérhetőség	Licenc	Op.rsz.
Adobe Photoshop		http://www.adobe.com/products/photoshop.html	Proprietary software	Windows/ Mac OS X
Paint.NET		http://www.getpaint.net/	Free software	Windows
GIMP (GNU Image Program)		http://www.gimp.org/	Free software	Plattform-független
Picasa		http://picasa.google.com	Freeware software	Windows / Linux / Mac OS X
Snagit		http://www.techsmith.com/snagit.html	Commercial software	Windows / Mac OS X
ACDSee Photo Manager		www.acdsee.com	Proprietary software	Windows / Mac OS X
CorelDRAW		http://www.corel.com/corel/product/	Proprietary software	Windows
Adobe Illustrator		http://www.adobe.com/products/illustrator.html	Proprietary software	Windows / Mac OS X
Inkscape		http://inkscape.org/	Free software	Windows / Mac OS X / Unix-like

7. ábra: A legelterjedtebb grafikus képnézegető, képszerkesztő és képmanipuláló szoftverek adatai

A pixelgrafikus szoftverek ismertetése után nézzük meg milyen lehetőségek állnak rendelkezésünkre a vektorgrafikus képek szerkesztése terén. A legnépszerűbb és legelterjedtebb vektorgrafikus szoftver a **CorelDRAW**, mely már 1987-től létezik a szoftverpiacon, igen megbízható, széles körben elterjedt. Szintén kedvelt vektorgrafikus szoftver az **Adobe Illustrator**, mely megtalálható a Master Collection csomagban. Mindkét termék **properietary software**[13]. Létezik egy **Inkscape** nevű, nyílt forráskódú, vektorgrafikus **free software**, amelyet 2003-ban fejlesztettek ki, jelenleg negyven nyelven elérhető, köztük magyarul is. Végül meg kell említenünk az **Adobe Flash**-t, amely elsősorban animációkészítő és nem képszerkesztő program, viszont vektorgrafikus képeket

[13] Zárt forráskódú szoftver. Nem összekeverendő a kereskedelmi forgalomban lévő szoftverekkel.

lehet létrehozni benne, egyébként nagy támogatást kapnak a pixelgrafikus képek is a szoftverben.

b. Hangok szerkesztése

Mindig gondoljuk át, hogy teszünk-e zenei aláfestést egy weboldalra, hiszen a háttérzenék és hangeffektek elvonhatják a figyelmet a szöveges tartalmakról. A reklám, a portfólió és a művészi jellegű weboldalakra szokás háttérzenét rakni, mert az erősíti az elérni kívánt hatást. Számos oldalon, például zenekarok oldalain elengedhetetlen, hogy néhány szám elérhető és meghallgatható legyen az oldalaikon. A hanganyagoknak is előmunkálaton kell átesniük, mielőtt weboldalra kerülnek. Például egy lejátszandó hangrészlet elején fel kell emelni a hangerőt a végén pedig el kell halkítani, nem kezdődhet és fejeződhet be egy hanganyag maximum hangerővel, szükséges az átmenet. Szükség lehet arra, hogy a hanganyagból kivágjunk részleteket. A háttérzenék esetében pedig úgy kell indítanunk és lezárnunk a hangfájl végét, hogy ciklikusan lejátszható legyen anélkül, hogy a felhasználóknak feltűnne az, hogy ismét elölről indult a zene. A felsorolt előmunkálatokat többféle szoftverrel el lehet végezni, a professzionális hangszerkesztő programok kategóriájába tartozik a Sony által fejlesztett **Sound Forge**[14] és az **Adobe Audition**[15], amely korábban Cool Edit néven futott. Mindkét program **proprietary** kategóriába sorolt termék, de kipróbálás céljából elérhető a harminc napos trial verziójuk az interneten. Létezik **szabad forráskódú**, magyar nyelven is elérhető zenevágásra alkalmas hangmanipuláló szoftver is, amelynek neve **Audacity**[16]. Mindhárom program futtatható Windows és Mac OS X alatt is.

c. Videók szerkesztése

Számos esetben szükség van arra, hogy egy weboldalon videofelvételek is megjelenjenek, nem kizárólag a videomegosztó portálokon szokás elhelyezni mozgókép anyagokat. Ezek előkészítésére és vágására szintén rengeteg szoftver létezik. Professzionális eszköz az Adobe System cég terméke a **Premier Pro**[17], a Master Collection csomagban található meg ez is a Photoshop mellett, amely sajnos egy drága szoftver. Kiváló program, a szintén zárt forráskódú és kereskedelmi forgalomban lévő **Vegas Pro**[18], mely a Sony cég terméke. Mindkét prog-

[14] Hivatalos oldal: http://www.sonycreativesoftware.com/soundforgesoftware, 2012.

[15] Elérhető trial verzió: http://www.adobe.com/cfusion/tdrc/index.cfm?product=audition, 2012.

[16] Hivatalos oldala: http://audacity.sourceforge.net/, 2012.

[17] Elérhetőség: www.adobe.com/products/premiere.html, 2012.

[18] Elérhetőség: http://www.sonycreativesoftware.com/vegaspro, 2012.

ram sajnálatos módon a Commercial proprietary software[19] kategóriába tartozik, és egyik sem olcsó. Amennyiben Windows operációs rendszerrel rendelkezünk, ingyenesen letölthetjük a freeware **Windows Movie Maker** nevű programot a Windows oldaláról[20], mellyel könnyedén megvághatóak a videoanyagok.

d. Animációk szerkesztése

Animációkészítés terén kétségtelenül az Adobe Flash a legjobb szoftver. A program tulajdonképpen egy rendkívül jó multimédiás platform, amely egyaránt jól kezeli a pixel és a vektorgrafikus képeket, amelybe a hanganyagok és a videók könnyedén beépíthetőek és könnyen kezelhetőek. Nagy előnye, hogy a grafikus felületen megjelenő elemekhez és velük kapcsolatos eseményekhez Action Script (egy objektumorientált nyelv) programkódokat rendelhetünk, így nagyon látványos, interaktív oldalakat és alkalmazásokat lehet vele készíteni. Legelterjedtebb alkalmazási területei a webes reklámok és az internetes játékok. Teljes weboldalakat nem érdemes Flashben készíteni, mert a telefonok és az Apple eszközök nem támogatják a megjelenésüket, azaz sajnos nem lehet minden eszközön megtekinteni az oldalakat. Flashből HTML5-be lehet fordítani, amely szabványt már minden böngésző, köztük a telefonokon lévők is támogatnak.

Álljon itt egy szemléletes példa Angelina Jolie színésznő munkásságáról, a teljes multimédiás anyag Flashben készült.[21]

ANIM

4. Fájlkezelők

Bár a weblapfejlesztő programok és editorok egyaránt képesek beforgatni és kezelni site-unk könyvtárszerkezetét, mégis számos esetben szükség van arra, hogy egy programtól független fájlkezelő segítségével rendezzük mappákba és nevezzük át fájljainkat. FTP kapcsolatot megvalósító fájlkezelőkre amiatt van szükség, hogy az elkészített website-unk teljes mappáját a benne lévő összes fájlal együtt fel tudjuk tölteni egy webszerverre. Az alábbi táblázatban lévő két eszköz remek megoldás erre a célra. A profi weblafejlesztők a FileZilla-t részesítik előnyben.

---

[19] Jelentése: üzleti forgalomban lévő, zárt forráskódú szoftver.

[20] Elérhetőség: http://download.live.com/moviemaker, 2012.

[21] Tóth Nikoletta, 2005-ben végzett médiatechnológus asszisztens szakos diák szakdolgozati munkája.

Program neve	Program emblémája	Elérhetőség	Licenc	Op.rsz.
Mozilla Filezilla		http://filezilla-project.org/	Free software	Multi-platform
GIMP (GNU Image Program)		http://www.ghisler.com/	Shareware software	Windows / / Windows Mobile / Android

8. ábra: Legismertebb, szerver oldali fájlkezelők

2.2.6 Webtárhely – hogyan biztosítsuk weboldalunk publikálását?

Már esett szó arról, hogy ha közzé szeretnénk tenni weboldalunkat az interneten, szükséges az, hogy egy webszerver egy bizonyos területére töltsük fel a teljes munkakönyvtárunk, más néven a site-mappánk tartalmát. A webszerverek ezen területeit – amelyek használata egyes szolgáltatók esetében reklámozási felületért cserében ingyenes, mások esetében fenntartásukért borsos árat kell fizetnünk – nevezzük webtárhelyeknek. Egy webszerveren általában több weboldal is található. A minőségi szolgáltatásokért, azaz nagyobb méretű tárhelyért, az adataink védelméért és a stabil, biztonságos működésért mindenképpen érdemes fizetnünk, az ingyenes webszolgáltató helyek általában az oldalunkra sok reklámot helyeznek el, s a működésük sokszor kiszámíthatatlan, sajnos nem folyamatosan biztosítják a weboldal kifogástalan működését. Ráadásul a szolgáltatások ingyenessége miatt mégcsak nem is reklamálhatunk. Gyakorlásra az ingyenes tárhelyek is megteszik, de a komolyabb, hosszútávú működtetésre és nagyobb látogatottságra szánt weboldalak esetében – főleg ha egyedi domain nevet is szeretnénk fenntartani – elengedhetetlen az, hogy minőségi webhoszt szolgáltatást igényeljünk. Mindkét esetben regisztrálás után egy accounttal (hozzáféréssel) fogunk rendelkezni, amivel kapunk egy megadott méretű webtárhelyet, kapunk egy FTP elérést és egy webcímet. Szolgáltatástól függően igényelhetünk adatbázis-hozzáférést is. A regisztráció során választott és kapott fontosabb adatokat mindenképpen jegyezzük fel annak érdekében, hogy bármikor elérhessük a kívánt területet. A feltétlenül feljegyzendő adatok: az admin-felülethez és az FTP hozzáféréshez tartozó felhasználói név, jelszó és a webcím. Ezen információk birtokában könnyedén közzétehetjük site-unkat az interneten.

2.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

2.3.1 Összefoglalás

A fejezetben a World Wide Web, azaz az internet webszolgáltatásával ismerkedtünk meg. Szó volt az URL címekről és az IP címzésről, a kliens-szerver modell lényegéről, és arról a folyamatról, hogy hogyan töltődnek le a böngészőnk ablakaiba az egyes weboldalak. Majd a statikus és dinamikus weboldalak közötti alapvető különbségekkel ismerkedhettünk meg. A fejezet második részében a weblapfejlesztés eszközeiről, azaz a böngészőkről, editorokról, grafikus weblapszerkesztőkről, médiaelemek előállítására szolgáló programokról és a fájlkezelőkről esett szó.

2.3.2 Önellenőrző kérdések

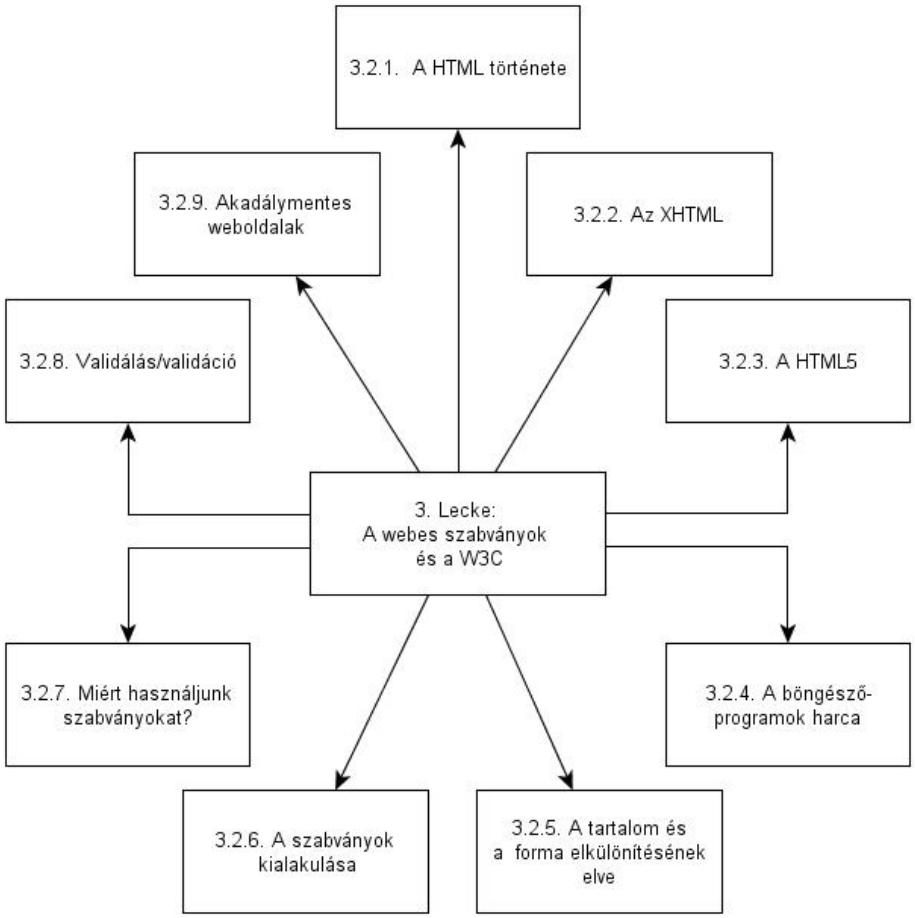
1. Kinek a nevéhez fűződik és mit jelent a WWW szolgáltatás?
2. Magyarázza el a kliens-szerver modell lényegét!
3. Mi az összefüggés az URL cím és az IP cím között?
4. Mi a különbség a statikus és a dinamikus weboldalak között?
5. Miért kell fájlkezelőt használnunk a weblapfejlesztés során?

3. A WEBES SZABVÁNYOK ÉS A W3C

3.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Cél, hogy az olvasó ismerje meg a weblapszerkesztés folyamatához kapcsolódó szabványokat és értse meg a szabványok használatának fontosságát. Továbbá ismerkedjen meg a HTML jelölőnyelv, a CSS és a böngészőprogramok fejlődésének rövid történetével.

3.2 TANANYAG



9. ábra: Fogalomtérkép

3.2.1 A HTML története

A **HTML** (Hypertext MarkUp Language) egy szöveges jelölőnyelv, más néven leírónyelv, amelyet weboldalak készítéséhez fejlesztettek ki, mára egyes változatai internetes szabvánnyá váltak a **W3C** (World Wide Web Consorcium) jóvoltából. A HTML leírónyelv egy általános leírónyelvből, az **SGML** (Standard Generalized MarkUp Language) dokumentumszabványból jött létre. Az SGML 1986-ban lett ISO szabvány, dokumentumok általános leírására szolgált, ennek elődje volt a **GML** (Generalized MarkUp Language), amelyet pedig már jóval korábban, az 1960-as években fejlesztettek ki az IBM-nél. A HTML kódokat a böngészőkön kívül még fel tudják dolgozni az ún. **aural böngészők**, amelyek a weboldalon lévő szövegeket és jelölőket olvassák fel, a **braille olvasók**, amelyek braille írássá alakítják a weboldal tartalmi részét, a levelezőprogramok, amelyeken keresztül HTML formátumban megírt hírleveleket szoktunk kapni és a mobileszközök.

A HTML-t 1990-től használják weblapok készítésére és az interneten a weboldalak megjelenítésére, öt jelentősebb állomást nevezhetünk meg a HTML fejlődésében.

1990-ben **a HTML 1.0**-val jöttek létre a dokumentum tartalmára vonatkozó címkék a fejléchez és a törzsrészhez. Ekkor alakultak ki a headingek, a hiperhivatkozások, a bekezdések és a listák jelölői − máig ezeket használjuk. Képek megjelenítésére ebben a verzióban még nem volt mód, de a dokumentum típusdeklarációját (DTD) már ekkor is meg lehetett adni a dokumentum legelején, még a <head> jelölő előtt.

1995-ben a **HTML 2.0** hozta el azt a változást, hogy már képeket is lehetett elhelyezni a weboldalakon és a szövegben félkövérrel vagy döntött stílussal kiemelni részeket. Ennek a verziónak a kiegészítésében megjelentek az űrlapok használata, ettől kezdve lehet többsoros szövegeket bevinni és több opció közül választani az űrlapokon. Az űrlapok elhozták azt a változást, hogy az alapjában véve egyirányú információközlést felváltotta a kétirányúság, azaz ettől fogva a felhasználó is küldhet adatokat a szerver felé.

Nagy újítást jelentett az 1996-ban elfogadott **HTML 3.0** és **HTML 3.2**-es verziók, hiszen ezekben már java appletek és scriptek (apró programocskák) beillesztésére is volt mód. Ettől kezdve le lehetett kérdezni, hogy milyen böngészőt és mekkora felbontást használ a felhasználó. Megjelent a <style> jelölő és megadhatóvá vált a karakterek betűtípusa; továbbá megjelent a táblázat készítés eleme is.

1997-ben a **HTML 4.0**-et a W3C ajánlássá tette és 1998 áprilisában hivatalos szabványnak mondta ki. Ez a verzió már megfelelt az ISO 8879 előírásnak és

az SGML szabványnak egyaránt. A négyes verzióban már megjelent a nemzetközi karakterkészletek és a balról jobbra olvasás támogatása. Hivatalossá tették a keretek (frame-k) használatát, további fejlesztésre kerültek a táblázatok és űrlapok használata. A fejlesztések alatt már szemmel tartották a csökkentett képességűek érdekeit. Hiába definiálta a W3C ebben az ajánlásban a használható kulcsszavakat, nem minden böngésző jelenítette meg a tartalmakat, a böngésző készítői nem vették komolyan az ajánlásokat.

A HTML számos változata közül az 1999 decemberében kijött **HTML 4.01**-es verzió volt az utolsó SGML-en alapuló verzió. A HTML 4.01-es weboldalak után egy ideig az XML alapú **XHTML** szabvány volt használatban, de mivel annak 2.0-ás verzójának fejlesztését felfüggesztették, így ma már az XHTML sem a legnépszerűbb leírónyelv, pedig nagyon sok reményt fűződött hozzá a szigorítások révén.

3.2.2 Az XHTML (eXtensible[22] HyperText Markup Language)

Az **XHTML** a HTML 4.0 újrafogalmazása **XML**[23] leíró nyelvben (eXtensible MarkUp Language). A HTML és XHTML nyelvek között az eltérés a szigorúbb formai követelményekben mutatkozott meg. A HTML 4.01-es verziót volt hivatott az XHTML felváltani, mely már az SGML-től eltérően XML leíró nyelven alapult, az XHTML a W3C által ajánlott szabvány. Az XHTML is több változatot élt meg:

AZ XHTML 1.0 2000 januárjában, az XHTML 1.1 pedig 2001 májusában vált hivatalos W3C ajánlássá.

A szigorítások könnyebb érthetőségének kedvéért az alábbi táblázatban megtalálhatók a legalapvetőbb különbségek a HTML és az XHTML között. A tag, a jelölő, a címke és az elem egymás szinonim fogalmai úgy; mint az attribútum és a jellemző szavak is egymásnak szinonim megfelelői.

HTML	XHTML
Az elemek és attribútomok tetszés szerint írhatóak kicsi, illetve nagy betűkkel.	A DOCYPE! elem kivételével minden elemet és attribútumot csupa kisbetűvel kell írni. Az attribútumok értékei tetszőlegesen írhatóak kis- illetve nagybetűkkel.
A weblap készítője dönti el, hogy az üres elemeket lezárja-e a kódban vagy	Minden elem lezárása szükséges. Az üres elemek lezárását önmagukban, egy

[22] A szó jelentése bővíthető.

[23] Az XML-t bővíthető leírónyelvnek és bővíthető jelölőszabványnak is nevezik.

HTML	XHTML
sem.	SPACET követő per jellel kell lezárni. Pl.:
Egyes attribútumok értékei idézőjelek („) nélkül is megadhatóak.	Minden attribútumértéket kötelező idézőjelek („) között megadni.
Nem feltétlen szükséges minden attribútumnak értéket adni.	Minden megnevezett attribútumnak kell, hogy legyen értéke.
A kiszolgálók a kliensnek a HTML-ek tartalmát a text/html médiatípussal küldi át.	A kiszolgáló a kliensnek az XHTML-ek tartalmát többféle médiatípussal is küldheti. Elsősorban: application/xhtml+xml. Használhatja a text/html és a text/xml-t is.

Napjainkban is sok 4.01 és XHTML verzióval készült weboldal létezik, hiszen nem várható el, hogy a régen 4.01-es vagy XHTML 1.0-ás vagy 1.1-es szabvány szerint helyesen megírt weboldalak mindegyikét átírják a legújabb, HTML5-ös szabványra. Igaz, hogy a DOCTYPE átírása HTML5-re igen egyszerű, s szerencsére a HTML5 kompatibilis a korábban használt nyelvekkel, azaz értelmezhetőek így is a korábbi kódok, így a teljes kód átírására nincsen szükség. Az új oldalakat viszont legtöbben már valóban HTML5 leírónyelven készítik.

3.2.3 A HTML5

A HTML tehát az ún. **SGML** jelölőnyelven (Standard Generalized Markup Language, magyarul Szabványosított, kiterjesztett jelölőnyelv) alapul, míg az XHTML alapja az **XML** (eXtensible Markup Language, magyarul Bővíthető jelölőnyelv), amely már webes adatbázisok közti kommunikációt biztosító, bővíthető nyelv. A HTML5 fejlesztésének a kezdete 2004-re tehető, amely 2014-re zárult le. A szabvány kompatibilis a korábbi HTML és XHTML verziókkal, a DOCTYPE átírásával – mint már fentebb szó volt − már elviekben HTML5-ös is a kódunk, bár, ha csak ennyit teszünk érte, akkor nyilvánvalóan egyrészt régebbi elavult elemeket is tartalmazni fog a dokumentumunk, másrészt nem használjuk ki az leírónyelv adta új lehetőségeket. Szerencsés, hogy a HTML5-ös technológiát támogatják a mobileszközök is.

A HTML korábbi verzióihoz képest a HTML5 egy erősen átdolgozott változat, kifejlesztésének célja az volt, hogy a weboldalak működéséhez ne legyen szükség pluginek telepítésére, úgy, mint például az Adobe Flash, vagy a Microsoft Silverlight esetében. 2004-ben alakult meg a **WHATWG** (Web Hypertext Application Technology Working Group) − függetlenül a W3C-től −, akik nem az XHTML-t, hanem a HTML-t kezdték továbbfejleszteni. 2007-ben a W3C hivatalosan kijelentette, hogy a következő HTML-szabvány a HTML5 lesz és nem dolgozott tovább az XHTML2-n.

A HTML5 gyakorlatilag a HTML4 és XHTML1 összegyúrt új verzióját jelenti, amelynek szintaxisa már nem SGML-en alapul, hanem az XML-t és a nem-XML-t egyaránt támogatja. Jó, hogy a HTML 4.01-ben érvénytelenített elemek már nem kerültek bele az új szabványba (pl.: font, center).

Az XHTML1-hez viszonyítva a HTML5 lazábban kezeli a címke lezárási szabályait. Ma sokan már a HTML5-ös szabvány szerint készítenek weboldalakat, nagy valószínűséggel a jövőben ez a szabvány fogja adni az alapot az interneten megjelenő weboldalak zöménél. Már most sok böngésző támogatja az új nyelv új elemeit, ebben mindig is élen jár az Opera, a Firefox és a Safari.

Számos korábban nem használt új elem is került a nyelvbe az **alkalmazások fejlesztése,** a **médiakezelés** és az **oldalelrendezést megvalósító elemek** terén. Az oldalelrendezéssel kapcsolatosan megjelentek a navigációra utaló <nav>, a fejlécre <header>, a láblécre <footer> utaló és a szakaszra vonatkozó <section> tagek. Továbbá a **multimédiás támogatottság** révén létrejöttek a <video> és az <audio> címkék, amelyek célja, hogy a böngészők bővítmények nélkül is képesek legyenek hangok és mozgóképek lejátszására.

3.2.4 A böngészőprogramok harca és fejetlensége

A HTML leírónyelvben a kezdetekkor sokféle módon megengedett volt a kód leírása, a HTML leírónyelv elemeinek szintaxisát igen tágan lehetett értelmezni az 1990-es évek végén, túlságosan nagy szabadságot adott a nyelv a weblapkészítőknek, ráadásul, ha egy kód pontatlanul készült el (például nem volt lezárva egy korábban megnyitott páros tag), akkor is értelmezték és megjelenítették őket a böngészők. A nagy szabadság és a böngészők általi hibás kódok elfogadásának eredményeképpen egyre inkább hozzászoktak a weblapok készítői ahhoz, hogy a kódot nem szükséges pontosan leírni, hiszen a hibás kódokat is képesek értelmezni a böngészők, sőt azokat ignorálva/kijavítva valahogyan meg is jelenítik a tartalmat, még ha nem is minden böngésző egyformán.

Az böngészők ráadásul sokáig önkényesen értelmezték a jól megírt kódokat is, s ez gerjesztette azt, hogy a kódokat össze-vissza írták meg a weblapkészítők. Így a weboldalak alig néhány százaléka felelt meg az SGML szabványnak.

A HTML nyelv szabályai tehát sokáig nem voltak teljesen egyértelműek, a történeti áttekintésből is látszik, hogy számos verziószámot élt meg a leírónyelv, amely fokozatos szigorításokon ment keresztül, mégis a HTML csak a 4.0-ás verziótól kezdődően tesz eleget az SGML szabványnak.

1990-es évek végén a böngészőpiacot az Internet Explorer 4, és a Netscape Navigator 4 határozta meg, majd később beszállt a versenybe az Internet Explorer 5 beta verzió. A felhasználók körében mindegyik elterjedt volt, sosem lehe-

tett tudni, hogy az általunk készített weboldalt melyik típusú és mely verzió-számú böngészővel fogják megtekinteni, így arra kellett törekedni, hogy az mindegyikben jól jelenjen meg. A hibás kódokat biztosan minden böngésző másképp értelmezve javította ki. Amennyiben azt szerettük volna elérni, hogy minden böngészőben a felhasználók ugyanazt a képet lássák kénytelenek voltunk nagyon sok többletmunkát végezni, azaz minden elterjedt böngészőre külön megírni a HTML kódokat és még a weboldal betöltődése előtt Javascripttel lekérni a kliens számítógépről az információt arról, hogy éppen melyik böngésző fut az adott gépen, majd az annak megfelelő kódot küldeni. Látható, hogy elég fejetlen és tarthatatlan megoldás volt mindez, de a böngészők eltérő értelmezései miatt valóban nem volt más lehetősége a weblapkészítőknek, képtelenség volt olyan kódot írni, amely minden böngészőben pontosan ugyanazt eredményezte volna. A weblapkészítőknek két választása volt: vagy több kódot írtak, vagy egy olyat, ami egészen közeli megjelenítést produkált az eltérő böngészőkben és verziószámokban. Az utóbbi volt a profibb, ám nehezebb megoldás.

Nem véletlen volt alapvető elvárás az, hogy a böngészők végre egy adott HTML kódot ugyanúgy értelmezzenek. A mai böngészők szerencsére a szabványnak kimondott HTML kódokat már közel ugyanúgy értelmezik, ami óriási előrelépés a korábbi helyzethez képest. Ez a W3C Organization, **WaSP** (Web Standards Project) projektjének és elsősorban Tim Berners Lee személyének köszönhetően van így.

3.2.5 A tartalom és a forma elkülönítésének elve

A HTML leírónyelvek a böngészőkkel együtt fejlődtek. A hivatalos HTML változatokat a W3C hagyta jóvá és készítette el a HTML 4.01-gyel kezdődően.

> Egy weblapkészítő azt szeretné, hogy minden böngészőben és minden számítógépen ugyanúgy jelenjen meg az egyetlen elkészített kód alapján a weboldal. Amennyiben a szabványokat egy fejlesztő pontosan használja úgy, a kívánt cél a mai böngészőkkel már megvalósul.

Ehhez nagyon fontos az is, hogy betartsuk azt az alapszabályt, miszerint **a tartalmat és a megjelenést külön kell választani egymástól**. Azaz a HTML nyelven írt kód a tartalmon és a szemantikai jelöléseken túl ne tartalmazzon semmiféle vizuális megjelenésre, (pl: karakterformázásra, elrendezésre) utaló kódot.

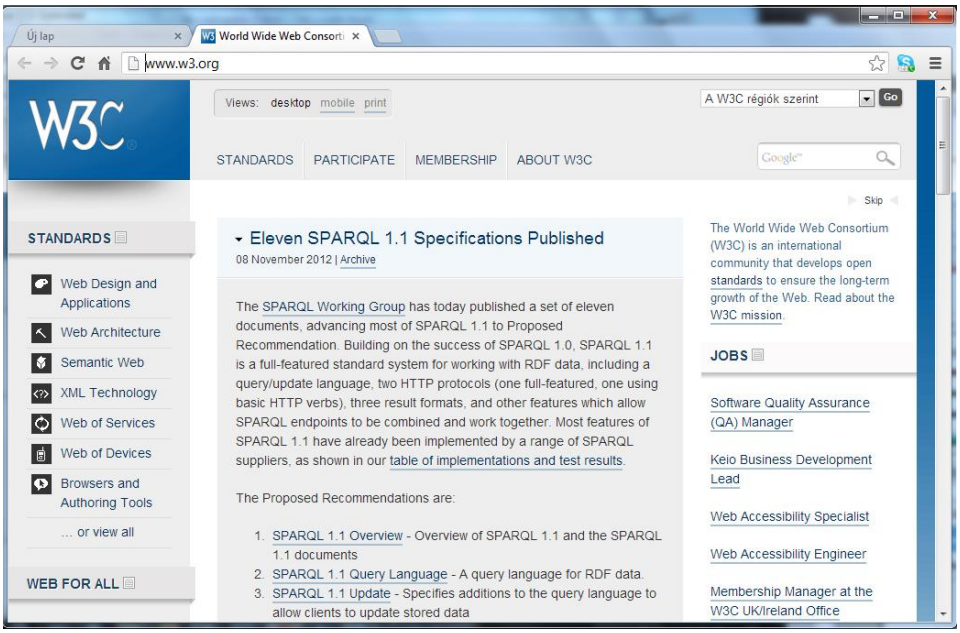
Fontos kiemelni azt a tényt is, miszerint a különböző böngészőkben eltérően lehetnek definiálva az alapértelmezett böngészőstílusok (pl.: háttérszínek, karakterek színe, mérete, font család, linkek színe stb.), melyből kifolyólag eltérő módon jelenhetnek meg oldalaink. Ezért ajánlott, hogy a weblapkészítő **saját stílust definiáljon** (amelyben maga adja meg az egyes HTML elemekhez (pl: body, p, h1, stb.) a megjelenést. Ennek használatával a kívánt tartalom minden böngészőben ugyanúgy jelenik meg, hiszen a saját weboldalstílus a weblapmegjelenítés tekintetében magasabb prioritású, mint a böngészőstílus. A saját stíluskészletet külső .css állományban érdemes létrehozni.

> A saját stílusok prioritásban előrébb vannak a böngészőkben definiált alapértelmezett stílusoknál, így minden böngésző a mi saját stílusleírásunkat veszi figyelembe, így egységesen, az általunk megadott stílusban jeleníti meg a tartalmat.

Ettől függetlenül a felhasználók – amennyiben szeretnék – kikapcsolhatják a weblapfejlesztő által készített stílusok használatát és így a böngészőstílus fog érvénybe lépni. Ezt akkor szokták tenni, ha például gyengénlátók és tudják magukról, hogy mekkora betűméretet és színösszeállítást tudnak könnyen olvasni, s arra módosították is maguknak a böngészőstílust. Ebben az esetben már az, hogy hogyan jelenik meg az általunk gyártott tartalom nem a mi felelősségünk, viszont arra fontos ügyelnünk, hogy a tartalom **szemantikailag helyesen jelenjen** meg, azaz jól strukturáljuk a HTML dokumentumban a tartalmat: használjuk például a címsorokhoz, a bekezdésekhez, a kiemelésekhez és a táblázatfejlécekhez a megfelelő HTML jelölőelemeket.

3.2.6 A szabványok kialakulása

Még 1994-ben alapította meg a **World Wide Web Consortium**-ot (**W3C**) Tim Berners Lee, amelynek, azóta is vezetője. A W3C szakemberei folyamatosan a web technológiai tökéletesítésén, és az újabb technikák kifejlesztésén munkálkodnak, továbbá a világhálón használható nyílt szoftverszabványokat, ajánlásokat tesznek. A cég hivatalos weboldala az alábbi címen érhető el: http://www.w3.org/.



10. ábra: A W3C hivatalos weboldalának képe

Sok specifikációt adtak ki, mint például a HTML 4.0, a CSS és a .png kiterjesztésű grafikus állományok szerkezetének a megfogalmazása, ezeket **ajánlásoknak** nevezték. A W3C jelentős munkát fektetett be és jelentős szerepet játszott a HTML nyelv szabványosításában a HTML 4.01 verziótól, az XHTML 1.0 szabvány létrejöttében, a CSS-ek használatának elterjedésében és az **akadálymentes weboldalak** megvalósításában egyaránt. Sajnos a megrendelőknek általában a szabványoknak való megfelelés és az akadálymentesség nem fontos szempont, így sok weblapkészítő nem figyel a szabványokban leírt szabályokra éz akadálymentességi elvek betartására. Pedig nagyon fontos, hogy a weboldalaink:

1. bizonyos szabvány, azaz HTML verzió szerint ún. **validok**, másrészt pedig
2. akadálymentesek legyenek.

3.2.7 Miért használjunk szabványokat?

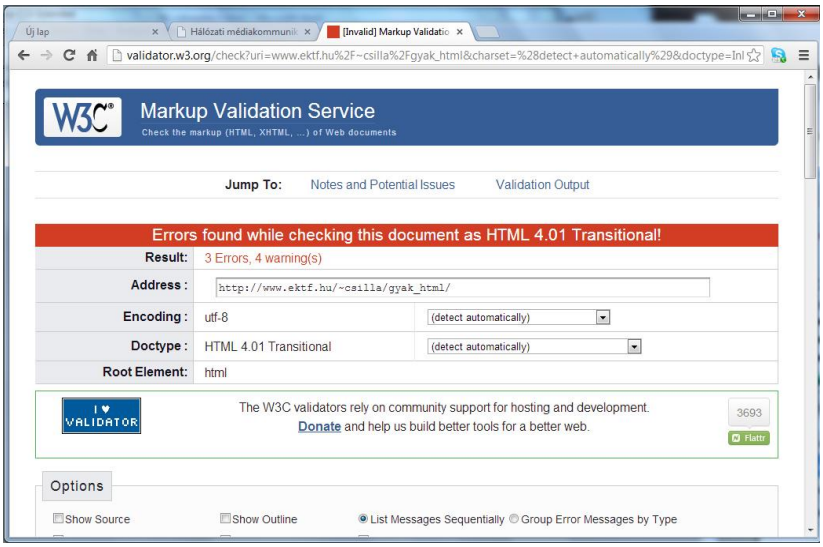
A legfrissebb előírások használatával olyan weblapokat készíthetünk, amelyek különböző eszközökön és böngészőkben a lehető legközelebbi végeredményt nyújtják. Az a helyes, ha a HTML-t (ami a tartalomért és a tartalmi jelölésért/struktúráért felel) és a CSS-t (amely a megjelenítést biztosítja) együttesen alkalmazzuk, ezekkel hatékony eredményt érhetünk el, továbbá a fejlesztési

folyamat is egyszerűbbé és átláthatóbbá válik, hiszen alapvető fontosságú, hogy a weboldal tartalma és megjelenése külön legyen választva. A legújabb és leghatékonyabb lehetőség az, ha a HTML5-öt használjuk együtt a CSS3-mal.

3.2.8 Validálás/validáció

A validálás folyamata a weboldal kódjának, azaz a választott szabvány helyes alkalmazásának az ellenőrzése. Szokás a weboldalakat HTML 4.0, HTML 4.1, XHTML vagy a HTML5 ajánlás illetve szabványok alapján validálni, továbbá létenek CSS validátorok is, amelyek a CSS állományunk kódjának a helyességét vizsgálják meg bizonyos verziók szempontjából. Sokféle validátor található a weben, a W3C validátora természetesen igen megbízható, hiszen ez a szervezet foglalkozik az ajánlásokkal és szabványosítással, ennek neve *W3C Markup Validation Service*, a címe pedig http://validator.w3.org.

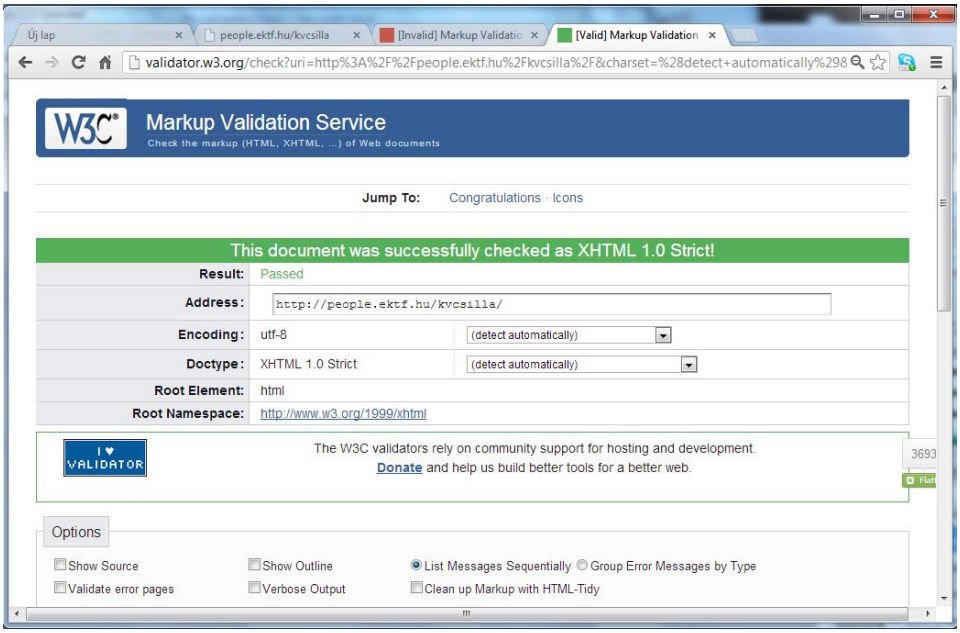
A validálás menete az, hogy a validátoroknak megadjuk a vizsgálandó weboldalunknak az URL címét, s az már ezek után a mi megadott címen lévő oldalunknak a kódját „átvilágítja”, s kiadja az adott szabvány szerinti hibák listáját, ezen tételekre egyesével klikkelve részletes információt kapunk az adott hibáról.



11. ábra: A W3C validátor képe, amikor egy oldal nem felel meg a HTML 4.01-es szabványnak[24]

[24] A vizsgált weboldal kódja tudatosan lett rosszul szerkesztve a példa kedvéért.

Az egyes hibákat pirossal kiemelve és azokat részletesen leírva jeleníti meg az értékelésben a szolgáltatás, továbbá a rossz kód helytelenségének indoklását is elolvashatjuk.



12. ábra: a W3C validátor képe, miután helyesnek találta egy Drupállal készült weboldal kódját

3.2.9 Akadálymentes weboldalak

Számos fogyatékkal élő személy böngészi a weboldalakat, vannak köztük vakok, gyengénlátók, színtévesztők, hallássérültek, mozgássérült felhasználók és értelmi akadályozottak egyaránt. Böngésznek a weboldalakon gyerekek, idősek, alacsony informatikai képzettségű személyek, valamint gyakran rossz hardvereszközzel vagy lassú internet-hozzáféréssel rendelkező felhasználók. Kirándulások, utazások esetén bárki juthat olyan helyzetbe, hogy az internetezéshez csak egy kisképernyős mobiltelefont tud használni és/vagy nagyon lassú az internethez való hozzáférése, vagy egy sokáig tartó nap után a fáradságtól rosszul lát és nehezen kezeli az egere, vagy eltört a keze, s emiatt nem tud egeret használni átmenetileg. Tehát bárki fogyatékkal nem élő is könnyen kerülhet ideiglenesen akadályozott illetve hátrányos helyzetbe.

Az akadálymentes weboldalak készítésének a célja az, hogy **mindenki egyenlő eséllyel férjen hozzá a weben megjelenő tartalmakhoz**, azaz a weboldalak lehetőleg a legszélesebb felhasználói kört szolgálják ki, azokon bárki köny-

nyedén megtalálja a keresett információkat és bárki, különösebb erőfeszítés nélkül könnyeden képes legyen azt használni.

A W3C a weblapok akadálymentesítésének irányelveit fekteti le a **WCAG 1.0**, illetve a **WCAG 2.0** ajánlásaiban[25]. A WCAG a **Web Content Accessibility Guidelines** angol kifejezés rövidítése, mely magyarra úgy fordítható szó szerint, hogy Tartalom Megközelíthetőségi Irányelvek, amely egy, a webes tartalmak egyenjogú hozzáférhetőségét érintő kezdeményezés. Azért esik szó erről ebben a fejezetben, mert ezek az ajánlások feltételezik a webes szabványok használatát is.

Az ajánlások szerint a szövegesen megfogalmazott pontok segítségével tudjuk elkészíteni az oldalainkat. A különböző szintű elvárásoknak megfelelően több szintű WCAG besorolás létezik: A, AA és a legkomolyabb követelményeket megfogalmazó AAA. Ezek az ajánlások tulajdonképpen hasznos tanácsok, amelyeket, ha betartunk, akkor megkönnyítjük a fogyatékkal élők helyzetét úgy, hogy az oldalon látszólag kívülről semmi sem változik, az átlagos felhasználók semmit sem vesznek észre belőle, viszont a weboldal koncepcionális felépítésének, úgynevezett **egyetemes tervezésének** eredményeképpen segítjük a fogyatékkal élők böngészői tevékenységét.

Az ajánlás egy weboldal elkészítésének az egészére vonatkozik, tehát ha valaki ilyen oldalt szeretne készíteni, akkor már a kezdetektől fogva eszerint kell felépíteni az oldalát. Ennek érdekében igen fontos az, hogy a **weboldalunkat a tartalomra koncentrálva készítsük** és **a tartalmi jelentés jelölésére is törekedjünk**. Ez azt jelenti, hogy a kódban értelemtükröző elemeket használjunk, mint például címsorok illetve kiemelések, listák használata, stb. Ezeket eleve a képernyő-olvasó szoftverek is felismerik és bemondják a szövegek olvasása közben, hogy használójuk maga előtt „lássa” a weboldal szerkezetét is. Tehát a képernyőolvasók használatával megtudhatjuk, hogy épp milyen jelölésű tartalmi elem következik a szövegben.

Az akadálymentes weboldalaknak számos feltételnek kell megfelelniük, többek között az egyik legfontosabb elv az, hogy **minden információt hordozó tartalom bárki számára szövegesen is elérhető legyen**, azaz szövegesen is megjelenjen az oldalon, hogy az a képernyőolvasó programok számára felolvasható legyen: fontos információkat tehát például nem szabad képként vagy képen megjeleníteni, mert az azokon lévő szövegekhez nem férnek hozzá a felolvasó szoftverek.

Fontos az is, hogy a weboldalon megjelenő képekhez mellékeljünk úgynevezett alternatív szöveget és a videókhoz rövid és hosszabb (teljes) szöveges leírást a vakok és gyengénlátók számára illetve azoknak, akiknek a kis sávszélesség miatt ezek az információk nem töltődnek le. Ezen esetben a kép illetve vi-

[25] Bővebben olvasható az ajánlások tartalma: http://www.w3.org/WAI/intro/wcag, 2012.

deó adott kontextusban érvényes tartalmát szokás leírni. A könnyed olvashatóságot például úgy segíthetjük, hogy színeiben erősen kontrasztos oldalakat hozunk létre, azaz sötét háttérre nagyon világos betűket helyezünk, vagy fordítva. Ezeken kívül számos apró szabálynak kell még eleget tenni ahhoz, hogy megkönnyítsük a fogyatékkal élők böngészőtevékenységeit.

Léteznek jó WCAG validátorok, amelyekkel a weboldalak akadálymentessége tesztelhető és például a Firefox alá telepíthető az ún. **Wave** kiterjesztés, amely a weboldal akadálymentességét ellenőrzi.

3.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

3.3.1 Összefoglalás

A fejezetben a tanuló teljes körű képet kapott a HTML leírónyelv fejlődésének történetéről, az XHTML és a HTML5 szabványok kialakulásáról, s a legújabb betartandó elvekről. Megtudhatta azt, hogy miért érdemes a szabványokat használni weboldalaink felépítéséhez és hogyan ellenőrizhető le az, hogy a kódjaink megfelelnek-e a szabványoknak, azaz validok-e.

3.3.2 Önellenőrző kérdések

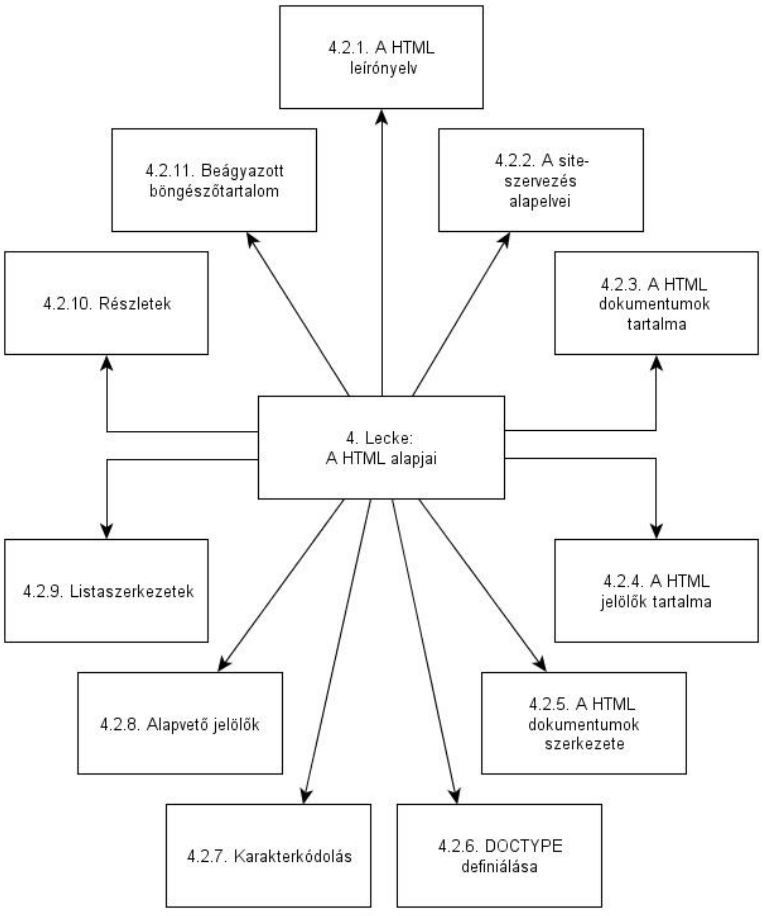
1. Mesélje el a HTML kialakulásának a történetét!
2. Miért volt szükség az XHTML létrehozására és mi a cél a HTML5 bevezetésével?
3. Sorolja fel a különbségeket a HTML és az XHTML leírónyelvek között.
4. Mit jelent és mire szolgál a validátor?
5. Miért érdemes szabványokat használni?

4. A HTML ALAPJAI

4.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Cél, hogy megismerkedjen az olvasó az XHTML és a HTML5 szerkezeti felépítésével és az alapvető jelölőkkel, azok attribútumaival. A fejezet a weboldalak fájlszervezésének szabályairól és a HTML jelölők szintaxisának bemutatásával kezdődik, majd a weboldalak szerkezeti felépítésén végighaladva az leghasználatosabb általános jelölők bemutatásával zárul.

4.2 TANANYAG



13. ábra: Fogalomtérkép

4.2.1 A HTML leírónyelv

A **HTML** leírónyelvvel (**Hypertext Markup Language**, magyarul Hiperszöveg leírónyelv vagy jelölőnyelv) határozzuk meg a weboldalak tartalmát és struktúráját. A weboldalak megjelenítéséért a **CSS** állományok (**Cascading Style Sheet**, magyarul Lépcsőzetes vagy egymásbaágyazott vagy rangsorolt stíluslap[26]) felelnek.

Az a helyes, ha a tartalom és a forma egymástól élesen elkülönül, a HTML (strukturált tartalom) és a CSS (formázás, megjelenés, elrendezés) állományokkal ez professzionálisan megvalósítható, ez azért fontos, mert így a tartalom és a megjelenés egymástól függetlenül is változtatható. A módszernek köszönhetően ugyanazon tartalomhoz többféle megjelenés is adható más-más CSS állományok csatolásával. Így például elérhető az, hogy a felhasználók maguk választhassanak többféle megjelenítés (szokás nevezni: témának is) közül. A CSS állományok HTML-ekhez csatolásának köszönhető az is, hogy egy website minden egyes oldala egységesen ugyanolyan megjelenésű, ez nem véletlen, hiszen a site minden egyes HTML oldalához ugyanaz a CSS állomány van csatolva. Ha valamit a megjelenésben változtatni szeretnénk, például más színűre módosítanánk a hátteret vagy más karakterméretet használnánk a címekben vagy más fejlécképet szeretnénk megjeleníteni az egyes oldalakon, akkor ezeken a paramétereken csak egy-egy helyen a CSS megfelelő sorában kell módosítanunk, s ezáltal a site összes HTML dokumentumán már látszik is a kívánt változás, így könnyen megvalósítható a weblapok gyors arculatváltása.

Például az említett módszer miatt vált feleslegessé és elavulttá az XHTML-ben és a HTML5-ben már nem létező <font> elem, hiszen ezt a szöveges bekezdések formázására használtuk a HTML kódban régebben. Ha a régi, elavult elvek szerint felépített weboldalakon szerettünk volna arculatváltást eszközölni – amikor is még nem különült el a tartalom és forma, hanem a karakterformázások is a HTML dokumentum részei voltak –, akkor minden egyes HTML dokumentumban lévő bekezdésszöveghez tartozó formázást át kellett volna írjunk, ahhoz, hogy egységesen megtörténhessen az arculatváltás. Nos, ez több száz HTML dokumentumból álló site esetében, amelyekben számos szöveges bekezdés van, akár hónapokat is igénybe vehetne, s ezen felül biztosan nem menne az egységesítés tévesztés nélkül.

Fontos hangsúlyozni, hogy a HTML nem programozási nyelv, azaz nem valósíthatunk meg vele szelekciót (feltételektől függő elágazást), sem iterációt

[26] A „cascading” szó mindhárom említett fordítása helyes és használatban van.

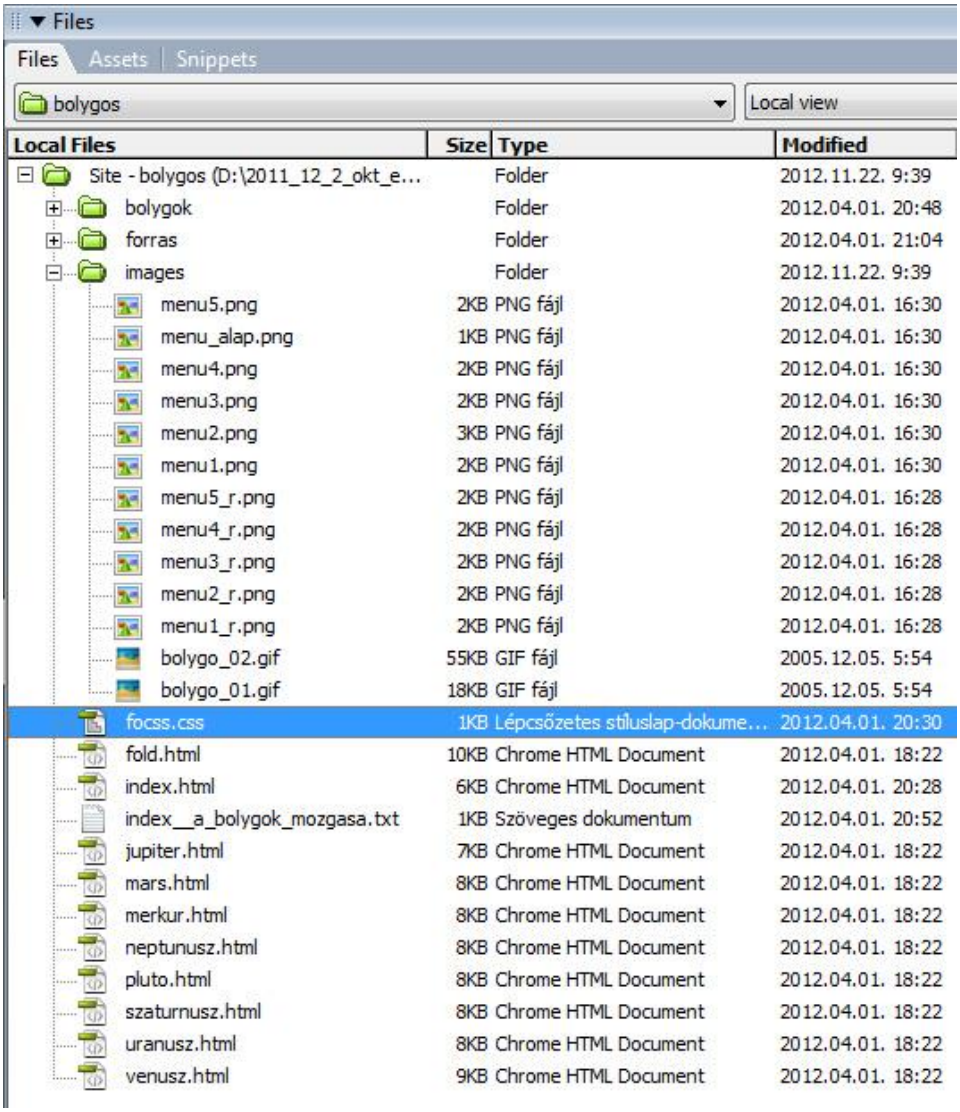
(ismétlődést, ciklusokat). A három vezérlési szerkezet közül csakis a szekvencia (parancsok egymásutáni végrehajtása) valósítható meg vele. Emiatt nem is mondhatjuk, hogy HTML nyelven programozunk (pedig sajnos sokan használják ezt a kifejezést), hanem csak az igaz, hogy HTML leírónyelven kódolunk.

4.2.2 A site-szervezés alapelvei – könyvtárszerkezet és névadási szabályok

1. Munkakönyvtár

Egyetlen gördíthető oldal egyetlen HTML oldalnak felel meg a böngészőben. A website-ok viszont számos HTML oldalt és az azokhoz tartozó médiaelemet tartalmaznak.

Nagyon fontos, hogy a weblapjaink készítése előtt mindig hozzunk létre egy ún. **munkakönyvtárat** (ezt site-könyvtárnak is szokás nevezni), ezen belül helyezzük el a website-hoz tartozó összes állományt. A munkakönyvtáron belül természetesen további alkönyvtárakat érdemes létrehozni és ezekbe osztályozni a site-hoz tartozó fájlokat. A site-mappa gyökerében foglal helyet az **index.html** nevű állomány, ennek a fájlnak a kiterjesztése dinamikus weblapok esetében általában .php, de lehet ettől eltérő is, a neve nem változik. Egy weblap kliens általi lekérése esetében a site-mappából automatikusan töltődik be az index nevű kezdőoldal, erre a felhasználónak hivatkoznia sem kell, a név azonosítja. Sok esetben azonban az index név rejtve marad, a böngésző címsorában nem jelenik meg, csak a tartalom töltődik le. Az index.html tehát a weboldal kezdőlapja, angolul homepage-nek, magyarul honlapnak szokás nevezni.



14. ábra: Példa egy site könyvtárszerkezetére

Tehát minden weboldal kezdőoldalának neve index, a böngészők mindig ezt a nevű állományt tekintik a kezdőlapnak. Érdemes a többi weboldal nevének utalnia a benne lévő tartalomra, ezek kiterjesztése .html[27] A 9. ábra példájában jól látható, hogy milyen típusú állományok szerepelhetnek egy website könyvtárszerkezetében: a legtöbb állomány .html kiterjesztésű, amennyiben

[27] Egy időben .htm kiterjesztést adtak a weblapkészítő programok alapértelmezetten, az is ugyanolyan megfelelő volt, ám ma ismét html kiterjesztés használata a szokás.

dinamikus tartalmakat megvalósító weblapot készítenénk, úgy .php kiterjesztésű állományok is szerepelnének. Látható, hogy az images nevű mappában .png és .gif kiterjesztésű képfájlok szerepelnek, de lehetnének köztük .jpg állományok is. A W3C a .png kiterjesztésű képállományokat javasolja a weblapokon használni. Látható, hogy az éppen kijelölt állomány kiterjesztése .css, ez a sitehoz kapcsolódó stílusállomány, egy weboldalhoz több .css állomány is tartozhat. Továbbá a weboldalak könyvtárstruktúrájában szerepelhetnek hang-, videoállományok (ezekről részletesebben a 6. fejezetben olvashat) .pdf-ekben hosszabb szövegek, s Power Point prezentációk, vagy egyéb bármilyen kiterjesztésű állomány, amelyre a HTML dokumentumokból linkelünk.

2. Hyperlinkek és elérési út

A website hiperlinkekkel összekötött rendszer, azaz a HTML dokumentumokat is linkekkel kell összekötnünk. Fontos, hogy a HTML állományokban lévő elérési utak megadása esetében – legyen az egy médiaelemre vagy egy másik HTML dokumentumra való hivatkozás a munkakönyvtáron belül -– mindig **relatív elérési utat** (relatív hivatkozást) használjunk, azaz mindig az adott, szerkesztés alatt lévő HTML állomány helyéhez viszonyítva adjuk meg a hivatkozott állomány elérését.

> Sose használjunk **abszolút hivatkozást** website-unkban, azaz ne adjuk meg egy fájl elérését oly módon, hogy az elérésben először a meghajtót nevezzük meg, aztán további mappák és almappák után a munkakönyvtárunkat és azon belül az adott fájlt. Ez amiatt volna nagy hiba, mert a weblap elkészülése után a munkakönyvtárunkat fel fogjuk másolni egy webszerverre, és azon, biztosan nem lesz jelen az elérésben leírt meghajtó, így nem fognak megjelenni az állományok sem.

3. Névadási szabályok

A **könyvtáraknak és** a bennük szereplő **bármely fájlnak a nevében** csak a következőket használjuk:

- **az angol ábécé kisbetűi;**
- **pozitív egész számok;**
- **alulvonás: _ .**

Fontos, hogy a fent megadott karaktereken kívül semmi más ne szerepeljen a fájlok és könyvtárak neveiben, azaz ne használjunk nagybetűket, ne hasz-

náljunk magyar ékezetes betűket, SPACE-t és egyéb speciális karaktereket (pl.: /, ?, ., :, ~, @) sem; a SPACE helyett szokás az alulvonást alkalmazni.

4.2.3 A HTML dokumentumok tartalma

A HTML, XHTML és HTML5 dokumentumok alapvetően formázatlan szövegeket tartalmaznak, szerepelnek bennük:

3. *a weboldalon megjelenő szövegrészek,*
4. *objektumhivatkozások (más állományokra való hivatkozások, elérési utak),*
5. *szöveges utasítások.*
6.

A szöveges utasításokat a HTML-ben kisebb és nagyobb jelek közé tesszük és **tageknek, magyarul jelölő**knek **vagy HTML elem**eknek nevezzünk (pl: <img>, <table>). Továbbá szokás **címké**knek is nevezni őket, mert a jelentéstartalmukon túl egyben címkét is jelentenek, amellyel a HTML egyes részeit jelöljük meg. Az egyes elemekhez meg lehet adni tulajdonságokat/jellemzőket, ezeket **attribútum**oknak nevezzük. Legtöbb attribútumnak a megadása opcionális, azaz nem kötelező.

4.2.4 A HTML jelölők szerkezete

A tagek legnagyobb része ún. **páros tag**, azaz létezik egy nyitó és egy záró eleme. Ezek ugyanazok a kulcsszavak kisebb-nagyobb jelek között írva, csak a záró tag különbözik a nyitótól abban, hogy a záró kisebbjel után, közvetlen a kulcsszó előtt van egy perjel (angolul slash: „/”).Egy tartalom elejét és végét jelölheti a páros tag, például egy címet, vagy bekezdést a szövegben, vagy egy táblázat elejét és végét stb.

Szintaxisa általános leírásban és egy példán keresztül látható az alábbiakban, a jelölők felépítését általánosan leíró keretezett részben szereplő „␣” jel a space-nek, magyarul a szóköznek felel meg.

```
<jelölő␣attribútumnév1=„érték1”␣attribútumnév2=„érték2”…>
  …
</jelölő>
```

Példa bekezdés jelölésére:
<p>Részt vettem egy gyorsolvasó-tanfolyamon. A Háború és békét nem egészen 20 perc alatt olvastam el. Az oroszok-

ról szól. (Woody Allen)
* .*

A példában lévő tagpár a bekezdés elejét és végét jelöli, az pár pedig a kiemelés kezdetét és végét. A példában az is látható, hogy a HTML elemek egymásba ágyazhatóak. Amire ügyelni kell a tagek egymásbaágyazása esetén az az, hogy mindig az újonnan megnyitott tageket zárjuk le először, s csak azután a korábban megnyitottakat.

Példa páros tagek egymásbaágyazására:
Jó példa: Alma ,
Rossz példa: Alma .

A párosak mellett néhány **páratlan tag**, más nevén üres tag is létezik. Az XHTML leírónyelv megköveteli az **üres jelölő**k használata esetén, hogy azokat önmagukban lezárjuk, korábbi HTML verzióknál erre nem volt szükség, és a HTM5 sem követeli meg ezt a szigorítást. Nézzünk példát üres jelölőkre! Páratlan tag a HTML5-ben például a
 (XHTML-ben: <br␣/>), amely soremelést és a <hr> (XHTML-ben: <hr␣/>), amely vízszintes vonalhúzást eredményez. Látható, hogy ezen tagek értelmetlenek is volnának párosan, hiszen nincs milyen tartalmat közrefogniuk. Az XHTML-es verziókban a space („␣”, magyarul szóköz) használata kötelező a perjel előtt. Tipikus üres elem a képek beszúrására használt jelölő, lássunk ennek szintaxisát általános leírással, majd egy példával.

HTML5:
<jelölő␣attribútumnév1=„érték1”␣attribútumnév2=„érték2”**…>**
XHTML:
<jelölő␣attribútumnév1=„érték1”␣attribútumnév2=„érték2”**…␣/>**

Példa HTML5-ben:
<img src=„/portrek/woody_allen.jpg” alt=„Woody Allen portréja”> és példa XHTML-ben:
.

A példáról jól leolvasható, hogy az elemekhez tartozó attribútumokat a címkenévtől és a többi attribútumtól szóközzel elválasztva kell megadni, az attribútumok nevei kisbetűsek, s minden attribútumnevet egy egyenlőségjel („=„) követ, amely után közvetlenül következik, feső idézőjelek között („ „) az attribútum értékének a megadása! A példában az <img> jelölőnek láthatóan két attribútuma van, az egyik az „src”, melynek jelentése source (magyarul forrás), ennek értéke a weboldalon megjelenítendő kép elérési útja (angolul PATH-ja) relatívan megadva az aktuális HTML-hez. Az „alt” attribútum pedig az alternatív

szöveget jelöli (alternative text), melyet akadálymentességi szempontból fontos, hogy minden esetben kitöltsünk.

Páratlan/üres tagek is párosak közé ágyazhatóak, tipikus példa erre, mikor egy képre klikkelve töltődik be egy weboldal.

Példa egymásbaágyazásra:
**
<img src=„ekf.png” alt=„Az Eszeterházy Károly Főiskola logója”>
**

A példában lévő kódban <a> jelölőpárok fognak közre egy <img> taget, ennek eredményeképpen a weboldalon egy kép jelenik meg, amelyre rákattintva betöltődik az Eszterházy Károly Főiskola weboldala. A példában jól látható a kódrész strukturáltsága. Helyes, ha a kódunkat úgy rendezzük, hogy az alacsonyabb hierarchia szinten lévő elemeket függőlegesen két szóközzel beljebb kezdjük az előző szinthez viszonyítva. A beágyazott elemeket is beljebb szokás írni két szóközzel, mint az azt közrefogó elemeket, ennek köszönhetően lesz a kód jól átlátható. Az átláthatóságnak köszönhetően pedig könnyebben érthető és javítható is, ezért ügyeljünk, hogy HTML kódjainkat ennek megfelelően strukturáljuk. A strukturáltságot jól szemlélteti a HTML dokumentumok szerkezetét bemutató 16. ábra is.

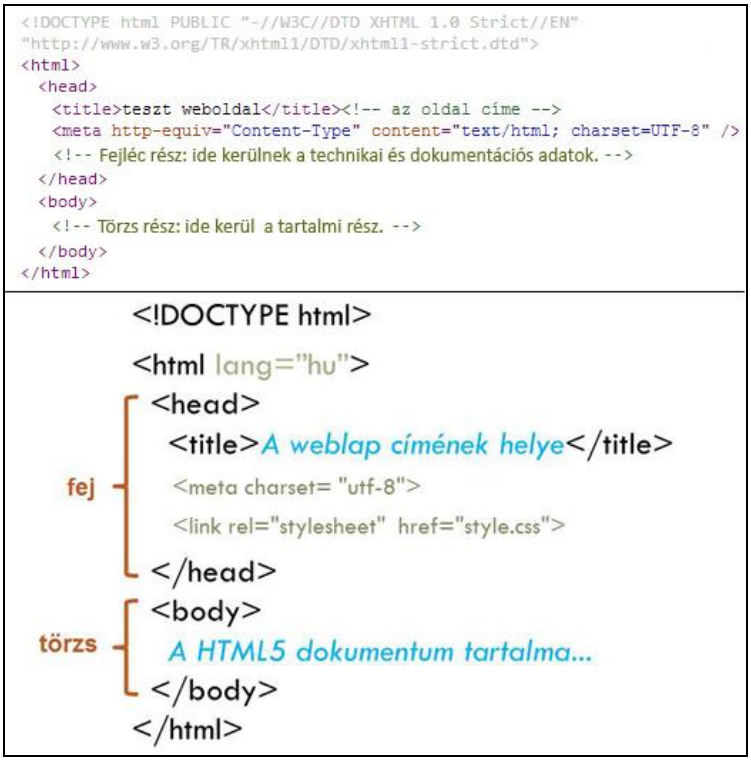
A weblapon megjelenő szövegeken, tageken, attribútumokon és attribútum-értékeken felül még néhány általános szöveges információ szerepel a dokumentum első részében, a fejrészben, s még az előtt, az ún. DOCTYPE bejegyzésben információ található a dokumentum típusáról. Fontos kiemelni, hogy a HTML dokumentumok nem tartalmaznak képeket, hang-, vagy videoállományokat, vagy bármi féle egyéb kiterjesztésű fájlt nem foglalnak magukban, csakis a megjelenítendő kép vagy egyéb állomány linkjét, másnéven elérési útját a munkakönyvtáron belül, s azt relatívan megadva. Tehát ha egy sok képet, videót és hangot tartalmazó weboldalt látunk, akkor nem azt kell elképzelni, hogy az egy nagyméretű .html kiterjesztésű állomány, mely mindent magában foglal, hanem tudnunk kell, hogy az ilyen weblapok is csak szövegeket tartalmaznak, s a rajtuk megjelenő kép hang vagy videoállomány külön fájlként valahol a .html dokumentumunk mellett helyezkedik el a munkakönyvtárban. A böngészők azok a programok, amelyek képesek a képeket jelölő tagek helyén a source attribútumban meghatározott elérésen lévő képet betölteni. Tehát csak a böngészőkben jelennek meg a képek hangok videók úgy, mintha azok a HTML dokumentumunk részei lennének.

4.2.5 A HTML dokumentumok szerkezete

Az XHTML és HTML5 állományok három fő részre bonthatóak.

1. A DOCTYPE! definíció az állomány legelején található, ami a dokumentumban használt szabványra utal.
2. A második rész a HTML fejléce, amelyet <head> és </head> elemek közé írunk, ide kerülnak a technikai és dokumentációs adatok leírása.
3. A harmadik egység a HTML törzsrésze <body> és </body> elemek között megadva, amely a böngészőben megjelenítendő információkat és azok szerkezetét tartalmazza.

Az alábbi képen látható egy XHTML és egy HTML5 dokumentumnak a szerkezete.



15. ábra: Az XHTML és HTML5 dokumentumok szerkezete

Mindkét szabvány esetében jól látható, hogy mindent megelőz a **DOCTYPE bejegyzés**, ennek kell a dokumentum elején szerepelnie, ez nem egy HTML tag, hanem egy iránymutatás a böngészőknek. A típusdefiníció után a dokumentumot a **<html> </html>** páros jelölők fogják közre, ezeken belül először a **fejléc** megadása történik a **<head>** és **</head>** jelölők között, majd **<body>** és **</body>** elemek között írjuk meg a dokumentum törzsrészét.

A <head> nyitó és záró elemek közé írt adatokat a böngészők nem jelenítik meg, ezek fontos háttérinformációkat szolgáltatnak a dokumentummal kapcsolatosan. A head részben szereplő **<title> </title>** elemek jelölik a weboldal címét, az ezen elemek közé írt szöveg jelenik meg a böngésző TAB-jain. Fontos az, hogy beszédes, a tartalomra utaló és kulcsszavakat magában foglaló címet adjunk az oldalunknak, mert a cím a felhasználók tájékoztatására és a keresőrobotok találati listájára egyaránt nagy befolyással van. A fejlécen belül a **<meta>** tagpárban szoktuk megadni az oldal karakterkódolását. Szintén meta tagben szokás megadni még a weboldal készítőjének nevét és a weboldalt jellemző kulcsszavakat is.

A fejléc után a törzsrészt a **<body>** és **</body>** jelölők határolják. Itt helyezkedik el az oldal tartalma, az ezen elemek közé írt tartalom látható a weboldalon a böngészőkben.

4.2.6 A DOCTYPE (dokumentumtípus) definiálása

Az XHTML ereje az egyszerű szabályokban és a könnyen követhető irányelvekben rejlik. Mint már fentebb is említésre került a korábbi HTML verzióknál jóval szigorúbbak ezen leírónyelv szabályai. Azáltal, hogy a szabályok szigorúbbak, sokkal egyértelműbb a kódszerkesztés és sokkal egyértelműbb azok értelmezése is a böngészők számára.

A HTML 4.01, az XHTML 1.0 és a HTML5 nyelvű dokumentumok mindig a DOCTYPE elemmel kezdődnek, amelyben a dokumentum típusa (kódolási módja) adandó meg, ez a böngésző számára leírja, hogy az miként értelmezze az adott dokumentumot. Ebből a kódrészletből tudják meg a dokumentumot megjelenítő böngészők, hogy milyen érvényességellenőrző szolgáltatásokkal ellenőrizendő a kód, és hogy az adott dokumentum valóban megfelel-e a megadott dokumentumtípus szabályainak. A HTML 4.01 és az XHTML 1.0 esetében három különböző DOCTYPE definíció létezik, mindhármat más DTD (Dokumentum Típus Definíció) írja le. Az XHTML 1.0 esetében erről részletes információk találhatóak az alábbiakban. A HTML5 szabványban egyféle létezik, amely a következő: **<!DOCTYPE html>**. A DOCTYPE kifejezést mindig csupa nagybetűvel írjuk.

A DTD dokumentumtípus megadása XHTML 1.0 leírónyelvben

Az XHTML különféle típusú dokumentumok megírását teszi lehetővé. Minden típus külön szabályrendszerrel rendelkezik. Az egyes típusokhoz tartozó szabályokat ún. DTD-kben (Documentum Type Definition), magyarul dokumentumtípus meghatározásokban írják le. Az XHTML-ek DOCTYPE bejegyzéseiben tehát azt is közöljük, hogy a dokumentumban melyik DTD előírást követtük a kód megírásakor. A weboldalak kódjai a helyes DOCTYPE típusú deklaráció nélkül nem érvényesíthetőek.

Az XHTML 1.0 három DTD-t kínál, azaz három féle DOCTYPE bejegyzés létezik a leírónyelv esetében, ezek alatt röviden megfogalmazva megtalálható, hogy mikor melyiket szerencsés választani:

7. *XHTML 1.0 Transitional (Átmeneti XHTML 1.0):*

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

Ez akkor használatos, amikor webszabványra térünk át és néhány elavult HTML tag még található a kódban elvétve, vagy amikor mások is hozzáférhetnek és módosíthatják a kódunkat, azaz beleírhatnak nem a szabványnak megfelelő részleteket. Amiatt kell ez esetben megadni, hogy a régebbi HTML parancsok is értelmezve legyenek.

8. *XHTML 1.0 Strict (Szigorú XHTML 1.0):*

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

Akkor érdemes használni, amikor tiszta, jelentésjelölő kódot készítünk olyan tárhelyen, ahol nem ír bele senki és semmilyen program kódrészeket.

9. *XHTML 1.0 Frameset (Keretvázas XHTML 1.0):*

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

Abban az esetben lehetne ezt használni, amikor a frameset szó szerepel a kódban, de nehéz elképzelni, hogy ma bárki is készít még frames szerkezetű weboldalt.

Mikor, melyik DOCTYPE definíciót érdemes használni XHTML 1.0 esetén?

A kérdés tehát pontosabban az, hogy mikor használjuk az **átmeneti (tranzisional)** és mikor a **szigorú (strict)** típust. A harmadik, **frameset (keretvázas)** típus nagyon nem használatos forma már napjainkban.

Strict

Mindenképpen a szigorú változatot érdemes használni, amennyiben tudjuk követni a szigorú szabályokat a teljes munkafolyamat során, ennek feltétele az, hogy csak mi férhessünk hozzá és felügyeljük a kódokat, azaz biztosak legyünk abban, hogy más szerkesztők vagy szabványokat nem követő tartalomkezelő rendszerek nem módosítanak a kódokon. Jó, ha a kódok csak **felépítéstükröző (szerkezeti)** és **jelentésleíró (szemantikus)** elemeket tartalmaznak, olyat nem, ami a külalakkal/külső megjelenéssel vagy a weboldal működésével foglalkozik. Ez utóbbi azt jelenti, hogy a külalak és megjelenítés megvalósítását kizárólag CSS-sel, a működést pedig kizárólag a háttérben futó parancsfájlokkal oldjuk meg.

Tehát ne szerepeljen a kódunkban például <font> tag a karakterek formázására, ne szerepeljenek a kódban elavult elemek, mint például *target* attribútum vagy a <body> tagnek a *bgcolor* és *textcolor* attribútuma, stb. Ne táblázattal legyen megvalósítva az oldalnak a felépítése/elrendezése és ne szerepeljenek a dokumentum szövegeiben JavaScript parancsok, se JavaScript hivatkozások. Főképp, véletlenül se használjuk a már igen régen elavult <frameset> elemet.

Transitional

Sajnos olykor arra kényszerülünk, hogy az XHTML 1.0 szabvány dokumentumtípus megadása esetén a Transitional, azaz az átmeneti változatot használjuk. Minden esetben ezt érdemes használni, ha az előzőekben felsorolt dolgoknak nem tudunk eleget tenni, azaz a felsorolt elemek bármelyike is szerepel a HTML dokumentum kódjában, vagy másoknak is van hozzáférésük a kódjaink szerkesztéséhez, még ha az egy tartalomkezelő rendszer is, nem egy élő személy. Ez a DPD elnéző a korábbi HTML tag-ekkel és a korábbi, ma már elavultnak minősített attribútumokkal szemben.

Frameset

A harmadik, Frameset ajánlás használatát mindenképpen kerüljük, nagyon nagy öngól ma frames szerkezetű oldalakat készíteni. A frames szerkezetű weboldalakkal elsősorban az a probléma, hogy a keresők nem az egész weboldalt (azaz nem a framesetet) találják meg, hanem csak annak egyik framejében lévő HTML-t, azt, amin a keresett adat áll; senki sem szeretné, ha az oldalainak csak egy területét dobná ki a kereső, például a jobb alsó fram-ben lévő információkat; anélkül, hogy megjelenne a menüsor, vagy a kezdőlapra mutató logó. A frame-s szerkezetű weblapok esetében hozzáférhetetlenek (nehézkesen hozzáférhetőek) az adatbázis adatok és nagyon nehézkes (működésképtelenek) a dinamikus oldalak működtetése. A Szabványkövető weblaptervezés című könyv nemes egyszerűséggel a következőképpen vélekedik a framek használatából adódóan megkövetelt XHTML 1.0 frameset típusú DOCTYPE definícióról: „Akkor érdemes használnunk, amikor valaki egy pisztolyt a fejünkhöz tartva keretvázas oldal írására kényszerít bennünket.”.

A DOCTYPE típusa hatással lehet az oldalak megjelenítésére is.

Nem mindegy, hogy melyik DOCTYPE-ot adjuk meg, mert a megadástól függően változhat ugyanannak a kódú dokumentumnak a megjelenítése ugyanabban a böngészőben is akár. Példaként nézzük meg a <body> tagen belül régen megadhatott bgcolor attribútumot! Tegyük fel, hogy a dokumentumban ez szerepel. Ha az XHTML 1.0 Strict-et adjuk meg a DOCTYPE-ban, akkor az érvényességellenőrzők hibát fognak jelezni és a szabványkövető böngészők figyelmen kívül hagyják az attribútumot, így a *bgcolor* attribútumnak megadott érték, azaz a megadott háttérszín nem fog megjelenni az oldalon. HA ezzel ellentétben az XHTML 1.0 Transition-t adjuk meg a DOCTYPE bejegyzésben, akkor a *bgcolor* attribútum, amely a régebbi HTML nyelvből jött minden további nélkül értelmezésre kerül és háttérszín formájában megjelenik az atribútumnak megadott érték az oldalon.

Hasonlóan szemléletes a *target* attribútummal kapcsolatos példa, amikor is egy külső weboldalt új ablakban szeretnénk megnyitni. A *target* tulajdonság az elavultnak nyilvánított elemek közé tartozik, azaz ilyen már nem szerepelhet egy szigorú XHTML kódban. Amennyiben a dokumentumunk értelmezését a Transition DTD ajánlás szerint szeretnénk értelmeztetni, minden további nélkül ez a

tulajdonság az <a href> tagen belül értelmezve lesz és a megnyitni kívánt oldal valóban új ablakban fog megjelenni. De ha a szigorú, Strict DTD ajánlás szerint szeretnénk a böngészőkkel értelmeztetni a kódunkat, akkor ezzel az attribútummal semmit sem fogunk elérni, helyette erre a célra JavaScript kódot lesz szükséges használni.

A DOCTYPE bejegyzés után az XHTML névtér megadása következik: http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd.

4.2.7 Karakterkódolás

A <head> tag első bejegyzéseként, meta tag-ben érdemes megadni a karakterkódolást. A magyar ékezetes betűk helyes megjelenítése érdekében használjuk az **UTF-8 kódolás**t, ennek megadása a következő HTML5-ben:

```
1  <meta http-equiv=„Content-Type”
   content=„text/html; charset=utf-8”>.
```

XHTML-ben ugyanez a kód a helyes, csak abban az esetben a záró hegyes zárójel elé ki kell tegyük a space és perjel párost is („␣/”), amely eleget tesz a .Figyelem! Néhány editorban (például a Notepadben) a dokumentumok mentése alapértelmezetten ANSI karakterkódolással történik, még akkor is, ha a formátumot .html-re állítottuk. Ez esetben feltétlenül ügyeljünk arra, hogy a dokumentumunk kódolása is UTF-8-ban legyen, ezt mentéskor be lehet állítani. Ha a meta tagben nem helyesn van megadva az UTF-8 kódtábla illetve a dokumentum nem UTF-8 kódolásban van mentve a magyar ékezetes betűk nem fognak helyesen megjelenni az oldalunkon. Ha egy nem csak angolnyelvű oldalon dolgozunk és az az ASCII táblában nem létező karaktereket használ, akkor is ezt a karakterkódolást érdemes használni.

Amennyiben magyar HTML5-ös weboldalt készítünk a <html> tagben a „lang” attribútum értékeként szokás megadni nyelvként a „hu” értéket. Ez az attribútum nem befolyásolja a magyar ékezetes karakterek megjelenítését a weboldalon, megadása a keresőrobotok számára hasznos, hiszen ennek megadásával működésük hatékonysága növelhető.

4.2.8 Alapvető jelölők

A HTML értelmezője nem veszi figyelembe az editorban végrehajtott formázásokat, így nem veszi figyelembe a sortöréseket, a tabulátorokkal beljebb

kezdett sorokat, sem pedig több szóköz egymásután történő gépelését. Ha egy HTML dokumentum szövegét az editorban megformázzuk, elmentjük, majd a változásokat megnézzük a böngészőkben látható, hogy semmilyen formázás nem maradt meg, a sorokat bal oldalt felül kezdi, a sorokat akkor vált, ha már egy adott sorba nem fér el több szó, több szóköz esetén pedig csak egyet rak ki.

A tartalom szerkezetének a kialakítását HTML elemekkel tudjuk csak elvégezni.

1. Sortörés

Amennyiben azt szeretnénk, hogy egy szövegrész új sorban kezdődjön az előző szövegrészhez képest, úgy **
** taget kell tenünk közéjük, ez soremelést eredményez a böngészőben. A címke a **break** (magyarul törés) angol szóból eredően kapta a nevét. A HTML dokumentumunkban lévő szövegrészek közé tehát
 tageket írva lehet tördelni a szöveget sorokra. A sortörés páratlan tag, azaz belőle egy gépelendő, amennyiben ki szeretnénk hagyni egy sort, kettőt kell belőle egymásután gépelni. Az XHTML nyelv megköveteli, hogy minden tag le legyen zárva, ezért a szabály szerint a tagen belül kell önmagát lezárni olyan formán, hogy a „br” kulcsszó után egy szóközt és egy „/” jelet teszünk (
).

```
1  HTML5-ben a sortörés: <br>
```

Nagyon fontos tudni, hogy címeket, alcímeket, bekezdéseket, felsorolást sorszámozást és egyéb más formázásokat nem helyes
 taggel szerkeszteni, sőt a bekezdések közötti távolságokat és sorközöket sem szabad ezzel szabályozni, ez utóbbiak megoldására CSS-sel való formázás alkalmazandó. A sortörést szokás versek rövidebb sorokratörésére használni.

2. Címsorok

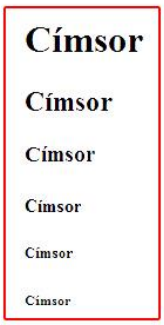
A címsor páros tagek segítségével átláthatóbbá és tagoltabbá tehetőek az oldalak. Segítségükkel hierarchia alakítható ki a címek között. Több szintjük is létezik, ez segít a felhasználóknak és a keresőmotoroknak eligazodni a weboldalon egyaránt. A HTML-ben hat szintje van a címsoroknak, ezeket rendre **<h1> … </h1>,** <h2> … </h2> és további tagpárokkal jelöljük egészen **<h6> … </h6>**-ig bezáróan. Minél nagyobb a „h” utáni szám, annál alacsonyabb szintű és egyben kisebb méretű az adott címsor. A címsor stílus blokkszintű elem, azaz az ebbe rakott szövegek mindig új sorban kezdődnek, emellett a címsorok alapértelmezetten félkövérrel kiemeltek. A „h” tag a **heading** (magyarul címsor) angol szóból ered.

Amikor egy oldalon címsorokat hozunk létre fontos az, hogy a legelső cím mindig <h1>-gyes címsor legyen, majd azt kövesse a <h2> az alcímek megadása esetén, majd a <h3> és így tovább. Sok esetben azért választják néhányan az első cím stílusaként a <h3>-at, mert a <h1> címsor alapértelmezett formázása túlságosan nagyméretű. Ez a megoldás helytelen! Első címnek mindig a <h1>-et kell választanunk és amennyiben túlságosan nagyméretűnek találjuk, úgy a CSS állományban kisebb méretűre kell állítani. Soha ne a megjelenítés, hanem a tartalmi fontosság alapján készítsük a kódunkat.

Az említett rossz megoldás a szövegelemek hierarchiájának szemantikai jelölése szempontjából nagyon helytelen! A képernyőolvasókat használók számára pedig értelmezhetetlen. A képernyőolvasót használó vak vagy gyengénlátó személyek is úgy olvassák a weboldalakat, ahogyan a hátrányokkal nem rendelkezők, azaz ők is először csak pásztázzák a tartalmat, majd a számukra lényegesnek ítélt információkat olvastatják csak fel.

> A képernyőolvasók forróbillentyűk lenyomásának hatására képesek csak az oldalon lévő címeket, vagy linkek felolvastatni, emiatt válik teljesen értelmetlenné az ha harmadikszintű címekkel kezdünk.

```
1  <html>
2  <head>
3    <title>teszt weboldal</title><!-- az oldal címe -->
4    <meta http-equiv="Content-Type" content="text/html;
     charset=UTF-8" /><!-- ez a sor a magyar karakterkészletért
     felelős -->
5  </head>
6  <body><!-- háttérszín szövegszín -->
7    <h1>Címsor</h1>
8    <h2>Címsor</h2>
9    <h3>Címsor</h3>
10   <h4>Címsor</h4>
11   <h5>Címsor</h5>
12   <h6>Címsor</h6>
13 </body>
14 </html>
```



16. ábra: A címsorok megjelenése a böngészőkben kóddal

Hiba, ha nem követjük a weboldal címeinek logikai hierarchiáját a jelölőnyelvvel, rossz például, ha minden soron következő cím rendre eggyel az előzőnél nagyobb számot kap. Nem az a cél, hogy sorszámozzuk a címeket, hanem az, hogy az egyes címekhez és alcímekhez hierarchiaszintet társítsunk. Így meglehet, hogy egy weboldalon csak a <h1>-gyes és a <h2>-es címsor-jelölőket használjuk.

Például Márai Sándorral foglalkozó weboldalon a címek hierarchiájának jó példája:
<h1>Márai Sándor életútja</h1>
<h2>Családi háttere</h2>
<h2>Egy polgár világpolgár lesz</h2>
<h2>Elismert íróként</h2>
<h2>Európai emigráció</h2>
<h2>Amerikai emigráció</h2>
<h2>Halála</h2>
<h1>Művei<h1>
<h2>Itthon megjelent művei </h2>
<h2>Az emigráció idején megjelent könyvei</h2>
<h2>Halála után megjelent kiadványok</h2>.

3. Bekezdés

A ** és ** tagek jelölnek és zárnak közre egy bekezdést. A címke neve a **paragraph** (magyarul bekezdés) angol kifejezésből ered. A címsorok jelölőihez hasonlóan ez a tagpár is blokkszintű elem, azaz minden bekezdés-jelölőpár közé tett szöveg új sorban kezdődik és a rá következő elem előtt is kimarad egy sor. A HTML bekezdésjelölőbe tett szövegrészek tehát térközzel különülnek el az oldal többi elemétől. Egymást követő két bekezdés azt az eredményt adja, hogy a két bekezdésszöveg között kimarad egy sor.

A és tagpárok közé a szövegben elhelyezhetőek
 tagek, amelyekkel ott törjük a sorainkat, ahol szeretnénk. Ha nem teszünk a szövegekbe sortörés címkéket, akkor a szöveg folytonosan fog kiíródni a böngészőben, hiába van az editorban tördelve.

```
1  <p>
2    Ez egy bekezdés, ami térközzel elkülönül
3    a többi elemtől az oldalon. Hiába van több
4    sorba írva a szöveg az editorban, a böngésző-
5    ben ez egy sorban fog megjelenni
6  </p>
```

Ha egy szöveget írunk a HTML dokumentumba, akkor azt mindig érdemes bekezdés tagek közé rakni, hogy a CSS-ben tudjunk ahhoz formázásokat rendelni.

4. Vízszintes vonal

A vízszintes vonalat produkáló **<hr>** címke segít vizuálisan részekre osztani a képernyő tartalmát, a sortöréshez hasonlóan nincsen záró párja, ez is páratlan tag, tehát XHTML-ben helyesen írva a záró nagyobb jelet megelőzi egy szóköz és egy perjel („/”). Akkor használjuk, ha szakaszokat szeretnénk elkülöníteni a képernyőn. A HTML5-ben ez az elem szemantikai értelemben már egy tematikus elválasztó is a grafikai megjelenésén felül. A címke neve a **horizontal line** angol kifejezésből ered.

```
1  HTML5-ben: <hr>
```

5. Megjegyzés

Amennyiben a forráskódhoz szeretnénk valamilyen magyarázatot fűzni, annak érdekében, hogy mások, vagy később a magunk számára is érthető legyen a kód, használjunk megjegyzés, angolul **comment** taget, jelölése: **<!-- -->**. A két jel közé lehet beírni a kommenteket, ezek semmilyen hatással nem lesznek a kódra és nem jelennek meg a böngészőben sem, egyszerűen könnyebben értelmezhetővé és átláthatóbbá válik tőlük a kód, ha velük azt magyarázzuk. A tag-ek több sort is magukban foglalhatnak, kódrészleteket is szokás kikommentezni, amennyiben azok nélkül szeretnénk tesztelni az oldalt. Azért bármit ide sem írhatunk, hiszen minden böngészőben könnyedén megjeleníthető a weboldal HTML kódja, másnéven forrása, tehát amit kommentben írunk az a forrás megjelenítésével láthatóvá válik bárki számára.

```
1  <!-- A komment szövege írandó ide.
2  Nem jelenik meg a böngészőben az a szöveg,
3  amit a comment tagek közé írunk. -->
```

4.2.9 Listaszerkezetek

1. Felsorolás

A felsorolások készítését kétféle módon oldhatjuk meg. Az első változat a számozatlan lista, ennek használatakor nincsenek sorszámozva az elemek, csak szimbólumok vagy képek (alapértelmezetten pontok) helyezkednek el minden elem előtt, ilyenkor használjuk az ** … ** jelölőpárt a teljes lista közrefogására, neve az **unordered list** angol kifejezésből ered (magyarul a jelentése: nem rendezett lista). A listában szereplő egyes listaelemeket **
** … **
** jelölőpárok közé helyezzük, az elem neve a **list item** angol kifejezésből ered. Ezek blokkszintű elemek, azaz minden listaelem külön sorba kerül. Szemantikailag

tehát ezeknek a jelölőknek a használata helyes listák készítésekor, semmi esetre se használjuk erre a célra a sortörést vagy bekezdés formátumot. A képernyőolvasók és a keresőrobotok is ezen elemek használatával képesek a listákat beazonosítani. A weboldalak menüjét is ilyen listaelemekkel kell megadni, viszont annak minden egyes listaeleme egy-egy hivatkozás lesz.

```
1  <ul>
2      <li>Mozilla Firefox</li>
3      <li>Opera</li>
4      <li>Internet Explorer</li>
5  </ul>
```

```
1  <html>
2  <head>
3    <title>teszt weboldal</title><!-- az oldal címe -->
4    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
     /><!-- ez a sor a magyar karakterkészletért felelős -->
5  </head>
6  <body><!-- háttérszín szövegszín -->
7    <ul>
8      <li>Mozilla Firefox</li>
9      <li>Opera</li>
10     <li>Internet Explorer</li>
11   </ul>
12 </body>
13 </html>
```

- Mozilla Firefox
- Opera
- Internet Explorer

17. ábra: A lista megvalósításának a kódja a megjelenő weboldaltartalommal

2. Számozott lista

A másik változatban számozott listát készítünk. Ez esetben a listaelemeknek fontos a sorrendje így azok előtt számok vagy betűk jelennek meg, a böngésző alapértelmezetten automatikusan számozza az elemeket. Ilyenkor az ** … ** jelölőpárt használjuk, melynek elnevezése az angol **ordered list** kifejezésből ered (magyarul: rendezett lista). Ebben a típusú listában is ** … ** jelölők közé írandók a listaelemek. Fontos megjegyezni, hogy mindennek, ami egy listába kerül, annak egy listaelem-páron belül kell lennie.

```
1  <ol>
2      <li>Mozilla Firefox</li>
3      <li>Opera</li>
4      <li>Internet Explorer</li>
5  </ol>
```

```
<html>
<head>
  <title>teszt weboldal</title><!-- az oldal címe -->
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
  /><!-- ez a sor a magyar karakterkészletért felelős -->
</head>
<body><!-- háttérszín szövegszín -->
  <ol>
    <li>Mozilla Firefox</li>
    <li>Opera</li>
    <li>Internet Explorer</li>
  </ol>
</body>
</html>
```

1. Mozilla Firefox
2. Opera
3. Internet Explorer

18. ábra: A számozott lista megvalósításának a kódja a megjelenő weboldaltartalommal

A felsorolást és számozott listát sokan úgy oldják meg, hogy egyszerűen felsorolnak egymás alá szavakat vagy kifejezéseket a végükre
 elemet téve, számozás esetén pedig manuálisan beszámozzák őket. Ez helytelen megoldás! Fontos, hogy a képernyőolvasók tudják, hogy mikor következik lista vagy felsorolás, hiszen akkor fel tudják olvasni, hogy „lista következik”, a vak vagy gyengénlátó felhasználók számára, akik képernyőolvasót használnak ez nagy segítség, hogy általa könnyen strukturálni tudják a fejükben a szöveget. Emiatt kell a felsorolás elemeket alkalmazni, és azért mert a HTML nyelv az szemantikus jelölő nyelv, azaz értelemjelölő nyelv.

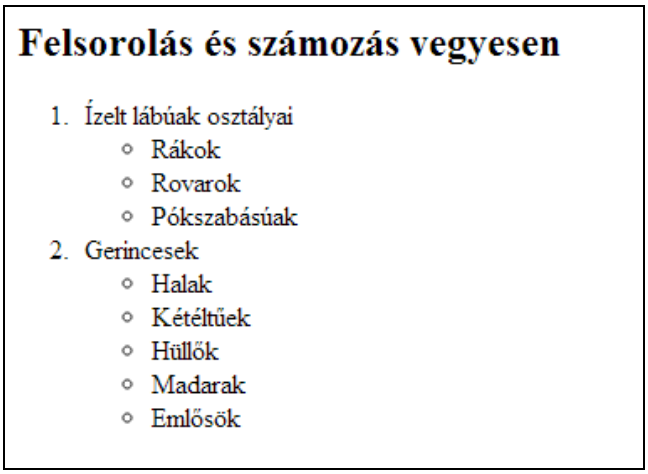
3. Egymásbaágyazott listák

A listák egymásba-ágyazhatóak, azaz az egyes listaelemekbe újabb listák (ún. al-listák) helyezhetőek. Lehet számozott listán belülre további számozott al-listát illeszteni, vagy felsorolás alá további al-felsorolásokat helyezni. A legcélszerűbb, ha ezeket vegyesen használjuk, s akkor igazán szemléletes, átlátható listákat kapunk.

Nézzünk meg egy számozott listába helyezett felsorolásra példát az alábbiakban! Az átláthatóság érdekében ez esetben különösen kell figyelni a kód strukturálására, azaz az editorban javasolt két sorközzel beljebb kezdeni a hierarchiában alacsonyabb szinten lévő elemeket (bár ez a strukturáltság nem jelenik meg a weboldalon, mégis csakis így vehetőek észre az esetleges hibák a kódban).

```
1  <ol>
2    <li>Ízelt lábúak osztályai
3      <ul>
```

```
4         <li>Rákok</li>
5         <li>Rovarok</li>
6         <li>Pókszabásúak</li>
7       </ul>
8     </li>
9     <li>Gerincesek
10      <ul>
11        <li>Halak</li>
12        <li>Kétéltűek</li>
13        <li>Hüllők</li>
14        <li>Madarak</li>
15        <li>Emlősök</li>
16      </ul>
17    </li>
18 </ol>
```



19. ábra: A felsorolás és a sorszámozás vegyes lista megjelenése a böngészőben

A böngészőben való megjelenésen látható, hogy a beágyazott listák alapértelmezetten behúzásra kerülnek.

4. Meghatározás lista, fogalom lista

A meghatározás listát szokás **definíciós listának** is nevezni, Lényege az, hogy szavakat, kifejezéseket vagy fogalmakat tartalmaz, amelyekhez magyarázatok vagy meghatározások tartoznak. A **<dl> … </dl>** jelölőpár fogja közre a definíciós lista egészét, melyen belül a kifejezéseket a **<dt> … <dt>** párok közé,

míg a meghatározásokat vagy más néven a magyarázatokat a **<dd> … </dd>** jelölők közé kell írni. A tagek a következő angol kifejezésekből erednek:

10. dl: **definition list**, magyarul definíciós lista;
11. dt: definition term, magyarul definíciós kifejezés;
12. dd: definition data, magyarul definíciós adat.

Lássunk egy konkrét példát!

```
1  <dl>
2    <dt>Halak</dt>
3      <dd>Fajai vízben élő gerincesek.</dd>
4    <dt>Kétéltűek</dt>
5      <dd>A kétéltűek átalakulással fejlőd
       nek.</dd>
6    <dt>Hüllők</dt>
7      <dd>A megtermékenyítés az anyaállat testé
       ben megy végbe.</dd>
8    <dt>Madarak</dt>
9      <dd>A madarak testét módosult szarupikke
       lyek, a tollak fedik. </dd>
10   <dt>Emlősök</dt>
11     <dd>Testüket szőr borítja, tüdővel lélegez
       nek. Elevenszülők.</dd>
12 </dl>
```

A leírt példa úgy jelenik meg a böngészőben, hogy a <dt> tagpárok közé írt tartalmak rögtön a sor elején kezdődnek balra igazítva, a <dd> tagpárok közé írt tartalmak pedig kicsit balról behúzva, balra igazítottan jelennek meg.

A példában egy fogalomhoz pontosan egy meghatározás társult. Viszont lehet egy fogalomhoz több meghatározást is megadni, s mindez fordítva is igaz, azaz egy meghatározás is megadható több kifejezéshez. Ez oly módon szabályozható, hogy például ha egy kifejezéshez szeretnénk megadni például több meghatározást, akkor a <dt> párból egyet írunk, amelyet több <dd> pár követ, s mindegyik párban más-más a meghatározás. Vagy a fordítottja is használható (több <dt>-t írunk és egy <dd>-t) például abban az esetben, mikor szinonim fogalmakhoz ugyanaz a meghatározás tartozhat.

4.2.10 Részletek

A **<details> … </details>** elempár segítségével megoldható az a HTML oldalakon, hogy elrejtett részleteket jeleníthessünk meg. Ezek olyan formában jelennek meg az oldalakon, hogy például egy „Részletek” felirat előtt megjelenik egy jobbra mutató háromszög, amely annak hatására, hogy a felhasználó a háromszögre kattint, lenyílik (lefelé mutat), s a lenyíló tartalomban további szövegrészek, például bekezdések olvashatóak. Nézzük meg az alábbiakban ennek a szerkezetnek a kódját egy akvarisztikával kapcsolatos feladatkiírás részletezésével. A **<summary>** tagpárban adható meg a jobbra mutató háromszög mögött olvasható rész, a summary angol szó magyarul összegzést jelent..

```
1  <details>
2    <summary>A feladat részletei</summary>
3      <p>Készítsen sósvizű akváriumot!</p>
4      <p>Várja meg, míg az „élő köveken” található
5      a microorganizmusokból látható élőlények
6      alakulnak ki pár hét elteltével!</p>
7  </details>
```

4.2.11 Beágyazott böngészőtartalom

Egy weboldalra **<iframe> …. </iframe>** jelölőpár használatával lehet beágyazni másik weboldalon (.html dokumentumban) lévő tartalmakat. Amennyiben nem fér el a beágyazott tartalom a számára szánt helyre automatikusan megjelennek a vízszintes és/vagy a függőleges görgetősávok. A címke elnevezése az **inline frame** angol kifejezésből ered.

A jelölőnek van egy kötelezőként megadandó attribútuma, amely nem más, mint magának a beágyazandó tartalomnak a címe, ezt az ún **src** (**source**) attribútumban kell megadnunk a kép (<img>) jelölőjéhez hasonló módon az adott tartalom elérési útját vagy URL hivatkozását kell megadnunk idézőjelek között. Lássunk egy példát, arra, amikor egy másik weboldal tartalmát szeretnénk megjeleníteni a saját oldalunkon. Az alábbiakban egymás után három példa látható, az elsőben csak egy .html kiterjesztésű oldal lesz beágyazva a weboldalunkra, a másodikban egy másik weboldal (ez esetben URL cím lesz megadva), a harmadik példában pedig YouTube videó.

```
1  <iframe src=„terkep.html”></iframe>
```

```
2  <iframe src=http://www.ektf.hu width=„35%”
   height=„40%”></iframe>
```

```
3  <iframe width=„560” height=„315” src=http://
   http://www.youtube.com/watch?v=DvVmqnNBo9w
   border=„0”></iframe>
```

Sok más esetben használatos még az <iframe> elem, például, amikor Google-térkép- részletet szeretnénk beszúrni egy „Elérhetőség” menüpont alá és az oldalunkra illesztjük a maps.google.com weboldalon felkínált kódrészletet. Figyeljük meg, hogy azt a kódrészletet is <iframek> fogják közre. Továbbá Flash animációk oldalba illesztését is <iframe>-mel oldjuk meg, bár a Flash animációk már egyre ritkábban kapnak helyet a weboldalakon, leginkább már csak applikációk illetve bannerek készülnek Flash-sel.

4.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

4.3.1 Összefoglalás

A fejezet első részében a weboldalak fájlszervezésének szabályairól, az XHTML és HTML5-ös dokumentumok szerkezetéről és az XHTML DTD-inek deklarálási módjáró volt szó. A második részben pedig a leghasználatosabb általános HTML5-ös jelölőkről esett szó, amelyek egy-egy példa keretén belül kerültek bemutatásra a szemléletesség és érthetőség érdekében.

4.3.2 Önellenőrző kérdések

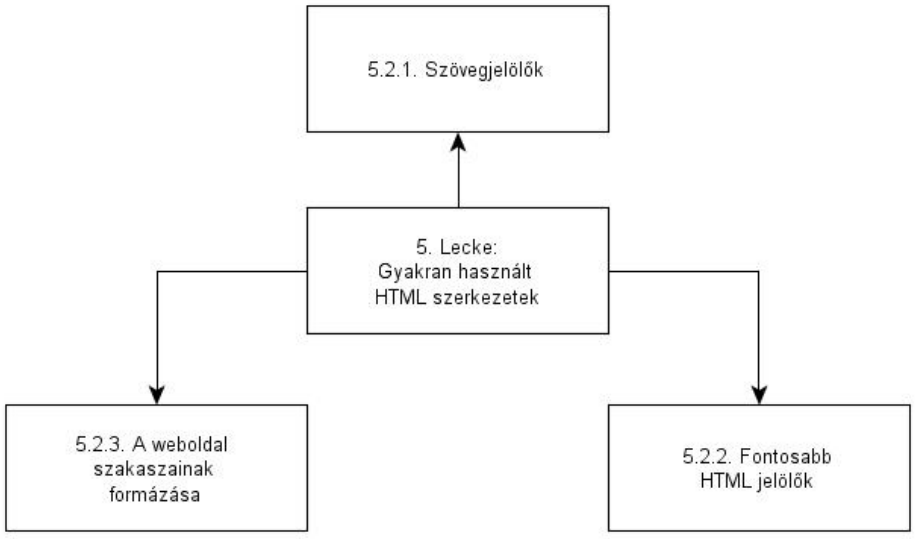
1. Mi a DTD és a DOCTYPE?
2. Milyen DTD ajánlások léteznek az XHTML 1.0 esetében, és melyiket mikor érdemes használni?
3. Meséljen a HTML5-ös dokumentumok szerkezeti felépítéséről!
4. Mi a különbség a páros és a páratlan tag-ek között?
5. Mondja el, hogy milyen típusú listák léteznek és azokat hogyan szokás létrehozni!

5. GYAKRAN HASZNÁLT HTML SZERKEZETEK

5.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A lecke további HTML5-ös elemek megismerését biztosítja. Először a szövegjelölő tagekkel ismerkedhet meg a tanuló, majd a gyakran használt képek, táblázatok, linkek beillesztésére szolgáló HTML5-ös elemek kerülnek bemutatásra. Végül szó esik két széles felhasználási lehetőségeket kiaknázó címke bemutatására.

5.2 TANANYAG



20. ábra: Fogalomtérkép

5.2.1 Szövegjelölők

A HTML a tartalom leírására és jelentésének megadására szolgálnak. A folyó szövegrészekben vagy a bekezdéseken belül előfordulhatnak különböző jelentéssel bíró részek, mint például az **idézet**, **kiemelés**, **rövidítés**, stb. A leíró-nyelv lehetőséget ad ezen jelentések megadására és ha valóban szabályos dokumentumot akarunk létrehozni, akkor ezeket alkalmaznunk kell. Ezek a címkék

zömmel hatnak a megjelenítésre is, de a tartalmi jelölés miatt fontos használatuk, hiszen hatékonyabbá teszik a keresők működését: segítik, hogy a találati listák megfelelő helyein legyenek a keresett szövegrészek.

1. Idézetek

Az idézetek jelölésére három lehetőség van. Hosszabb idézetekhez, amelyek számára blokkot különítünk el a **<blockquote> … </blockquote>** jelölőket használjuk. A böngésző nem teszi ki az idézőjeleket ebben a tagben megjelenített szövegrészek elé és mögé, viszont az idézet minden sora behúzva jelenik meg. Amennyiben verseket idézünk
 tagekkel tudjuk a szöveget sorokra tördelni. A <blockquote> jelölőhöz tartozik egy **cite** nevű attribútum, amelyben az URL forrás adható meg.

```
1  <blockquote
     cite=„http://www.citatum.hu/idezet/4286”>
     „Mily boldog az, ki feddhetetlen.<br>
     A világ is feledi, ki elfeledte már.<br>
     Ez a makulátlan elme örök ragyogása!<br>
     Az ima meghallgatásra lel, s a kívánság
     lemondásra.” (Alexander Pope)
2  </blockquote>
```

A **<q> … </q>** címkéket a soron belüli idézetek esetében használjuk, s a címkék közé írt szöveg automatikusan idézőjelek közé kerül a böngészőben, lássunk egy kódot példaként. A címke neve a quotation szóból ered.

```
1  <p>
     Shakespeare adta a IV. Henrik második részében
     Suffolk szájába a következő szavakat: <q>Hol
     te vagy, ott a világ maga</q>. Sok helyen ol-
     vashatjuk ezt az idézetet.
2  </p>
```

A **<cite> … </cite>** tagpár között műveknek a címét szokás megadni, nevezzük ezt a jelölőt **címhivatkozás**nak is. Mű lehet könyv, festmény, szobor, film, tv-műsor stb. A személyek neve nem ide tartozik! A HTML 4.01-ben a <cite> tag még csak idézet meghatározására volt alkalmas, ma a HTML5-ben már a műveket jelöljük ezzel. A címke neve a **citation** szóból ered.

```
1  <p><cite>Over the town</cite> by Marc Chagall.
     Painted in 1918.
2  </p>
```

2. Kiemelések

A szövegekben megjelenő tartalmi kiemelésekből három félét különböztetünk meg. **Kihangsúlyozás**t ** … ** jelölőpárral adhatunk meg a szövegben, ez az angol **emphasis** szóból ered. **Erős kiemelés**re a ** … ** jelölőpárat használjuk, s a harmadik szövegből való kiemelésre használt tagpár a **<mark>…</mark>**, amely szó magyarul **jelölés**t jelent. A <mark> a leghangsúlyosabb a három közül, sárga hátteret ad a közrefogott karaktereknek, ez a HTML5-ös nyelvben új elemként jelenik meg. A másik két elem is különbözik megjelenésében az egyszerű szövegtől, az jelölők közé helyezett szövegek dőlten, míg a jelölők közötti szövegek félkövéren jelennek meg, de ennek ellenére nem a megjelenés miatt használjuk ezeket, hanem a jelentéstartalom kiemelése miatt.

```
1  <p>
     Ez nem <em>szép</em>, hanem
     <strong>csodálatos</strong>!
     Sőt kiemelkedően <mark>fantasztikus</mark>!
2  </p>
```

Biztosan sokan használják a ** … ** tagpárt arra, hogy a szövegeket félkövéren (angolul: bold), az ** … ** tagpárt, pedig arra, hogy a kívánt karaktereket kurzívan szedve (dőlt betűvel írva, angolul: italic) jelenítsék meg a böngészők. Ez a két karakterformázó tag a HTML5-ben is értelmezve van, ám amennyiben lehetséges ajánlott ezek helyett a HTML5-ben már jelentéstartalommal is bíró   iletve <mark> tageket használni, hiszen ezek tartalmi kiemeléseket is jelölnek, mellékesen ezen tagek használata is eredményezi a félkövér illetve a kurzív megjelenést. A w3schools.com, a W3C Organization on-line HTML iskolája azt javasolja, hogy amennyiben a tag használata helyett az éppen aktuális rendszerben megengedett ezen célokra jobban megfelelő tag használata, akkor ezt ne alkalmazzuk.[28] Akadálymentességi szempontból is az újabb tagek (, és <mark>) használata ajánlott kiemelések megvalósítására.

[28] http://www.w3schools.com/tags/tag_b.asp, 2014.

Az <i> tag a HTML5-ben annyi jelentéstartalmat kapott, hogy ezt ajánlott használni műszaki fogalmak és kifejezések kiemelésére. Az <i> tag helyett ajánlottabb az tag használata, amely ugyanúgy kurzív megjelenést biztosít.[29]

Szokás még az egyes szövegrészeket az **<u> … </u>** (angolul: underline) páros közé írni annak érdekében, hogy a közrefogott karakterek a böngészőben aláhúzva jelenjenek meg. Az <u> taget HTML5-ben azon szövegrészek kiemelésére használják, amelyek stilárisan (stilisztikailag) különböznek a normál szövegtől, például a helytelenül írt szavak, vagy a kínai tulajdonnevek. Akadálymentességi szempontból nagyon fontos, hogy lehetőleg ne használjunk <u> taget olyan oldalakon, ahol azt lehet hinni az aláhúzott szövegekre, hogy azok linkeket jelentenek.[30] Emiatt csak igen indokolt esetben ajánlatos a használatuk, hiszen az aláhúzott szövegek a weboldalakon linkeket szokta jelölni.

3. Kisbetűs szövegblokkok

Kisbetűsen írja ki a böngésző a **<small>… </small>** elemek közé írt szövegeket. Ilyen elemek közé szokás írni a megjegyzéseket vagy a Copyright (szerzői jogi korlátozásokat) információkat, ez blokkszintű elem, azaz a jelölők közé írt szövegek új sorban jelennek meg.

4. Alsó- és felsőindex

Sok esetben van szükség bizonyos karakterek alsó- és felsőindexbe helyezésére a weboldalakon. A kémiai elemek vegyjeleinek leírásához például az alsóindexek használata, míg a terület és űrtartalom meghatározások esetében a felsőindexek használata elengedhetetlen. Míg az alsóindexbe helyezendő karaktereket **_…** címkékkel foghatjuk közre, addig a felsőindexbe megjeleníteni kívánt karaktereket **[…]** jelölők közé írandók. A <sub> tag az angol **subscript** szóból (magyarul: alsóindex), a <sup> pedig az angol **supersript** (magyarul felsőindex) szóból ered. A következő mondat kódja látható az alábbiaban: Izolda 1 km^3-es hordóba pontosan 1000 liter H_2O-t töltött.

```
1  Izolda 1 km<sup>3</sup>-es hordóba pontosan 1000
   liter H<sub>2</sub>O-t töltött. Majd ivott egy
   nagy pohár dihidrogén-oxidot.
```

5. Mozaikszavak

[29] http://www.w3schools.com/tags/tag_i.asp, 2014.

[30] http://www.w3schools.com/tags/tag_u.asp, 2014.

„A **mozaikszó** a szóalkotásmódok egyik formája, amely magában foglalja a **betűszókat** (amelyek az eredeti kifejezés egyes szavainak kezdőbetűiből állnak) a **szóösszevonásokat** (amelyek az eredeti kifejezés nagyobb egységeit tartalmazzák), valamint az egyéb mozaikszókat (amelyek teljes szavakat megőriznek).”[31] Tehát a betűszavak és szóösszevonások között van különbség, s korábban ezeket eltérő jelöléssel használták[32]. HTML5-ben mindegyik mozaikszóra ugyanazt a páros jelölőt használjuk, erre a célra van az **<abbr> ... </abbr>** pár. A címke neve az angol **abbreviation** szóból ered, amely magyarul **rövidítés**t jelent**.**

Az alábbiakban lássunk példát két rövidítés mögött rejlő szavak böngészőben való megjelenítésére, az egyik egy betűszó: OTKA (**O**rszágos **T**udományos **K**utatási **A**lapprogramok), a másik pedig egy szóösszetétel (amely a szavak egyes részeit tartalmazza): Ofotért (**O**ptikai Finommechanikai és **Fot**ócikkeket **Ért**ékesítő Vállalat).

```
1  <p>Az <abbr
   title=„Országos Kutatási Tudományos&nb
   sp;Alapprogramok”>OTKA</abbr> egy független nem-
   zeti intézmény, amely több mint negyed évszázada
   működik. Ez magyarországi munkahelyeken végzett,
   nemzetközileg is kiemelkedő alapkutatásokat tá-
   mogat pályázati szinten magyar és külföldi bírá-
   lók bevonásával.
2  </p>
```

```
1  <p>Az <abbr
   title=„Optikai Finommechanikai és 
   ;Fotócikkeket Értékesítő Vállalat”>Ofo
   tért</abbr> az a cég, amely Magyarországon elő-
   ször forgalmazott kontaktlencsét.
2  </p>
```

A title attribútumokban szereplő szavak közé „ ” karaktersor áll, ez a **nem törhető szóköz**nek a jele (angolul non-breaking space), azért kell ezt a

[31] A meghatározás a http://hu.wikipedia.org/wiki/Mozaiksz%C3%B3 weboldalon lévő Wikipédia szócikkből van idézve.

[32] A betűszavakra az ún. <acronym> páros jelölőt használhattuk, ami magyarul mozaikszót jelent, de ez már a HTML5-ben nem támogatott.

karaktersort elhelyezni, hogy ne csak az első szót jelenítsék meg a böngészők, így a jel közbeiktatásával az egész kifejezést egyetlen szóként értelmezik a böngészők. A mozaikszavak esetében a weboldalon történő első előfordulásakor szokás megjeleníteni a teljes elnevezést a fenti módon. Ez a kurzornak a böngészőben való mozaikszóra húzásával jeleníthető meg, Firefoxban aláhúzásra kerülnek ezek a mozaikszavak a böngészőben.

6. Meghatározás, definíció

Az előzőhöz hasonlóképpen működik a definiálás, a fogalmak jelentésének a meghatározása a **<dfn> … </dfn>** címkepár segítségével. A címke elnevezése az angol **definition**, magyarul **meghatározás** szóból ered. A meghatározás szövege maga nem íródik ki a képernyőre, csak meg tudjuk azt jeleníteni, amennyiben szeretnénk. Lássunk egy példát a megadására.[33]

```
<p>A <dfn
title=„A nyelv pszichológiája, egy tudományág.”>pszicholingvisztika</dfn>
biztosítja azt a kognitív folyamatot, amely lehetővé teszi, hogy nyelvtanilag és tartalmilag helyes mondatot alkossunk a felhasználható szókincs és nyelvtani struktúrák segítségével.
</p>
```

7. Címek

Az **<address> … </address>** jelölőpár határozza meg egy mű szerzőjének, alkotójának vagy tulajdonosának az elérhetőségét. Ezt a taget használjuk arra is, ha egy dokumentumnak vagy cikknek az elérését szeretnénk megadni.

Ha az <address> tag a <body> tageken belül helyezkedik el, akkor az a dokumentumok elérhetőségét jelzi, ha viszont az <address> egy <article> elemen belül foglal helyet, akkor a cikk eléréséről nyújt információt. A böngészők az <address> tagbe írt szövegeket általában dőlten jelenítik meg, s előtte/utána egyaránt sortörést tesznek.

Fontos megjegyezni, hogy ezt a taget alapvetően nem a postai címek leírásának közrefogására találták ki. Ne használjuk erre a célra, hacsak nem a cím maga a dokumentum illetve cikk eléréséhez szükséges.

```
<address>
  Készítette XY<br>
```

[33] A példa szövege a Wikipédia pszicholingvisztika szócikkéből vett.

```
5    A mű megtalálható a <a href=„
6    http://www.edx.org”>
7    http://www.edx.org webcímen.</a><br>
8  </address>
```

8. Előformázott szöveg

Szó esett már arról, hogy az editorban szóközökkel, tabulátorokkal illetve sorközökkel (gyűjtőnéven: elválasztó karakterek) formázott szövegek nem úgy jelennek meg a böngészőben, mint ahogyan az az editorunkban látható, hiszen ezen karakterekből minden esetben csak egy kerül megjelenítésre a böngészőkben. A **<pre> … </pre>** tagek által közrefogott karakterek hagyományos értelmezésétől a fordító eltekint, így ha ezen jelölők közé tesszük a karaktereinket megmarad a formázottság, természetesen ez csak speciális esetekben használható, nem szabad teljes szövegrészeket ilyen tagek közé helyezni. Alább látható egy példa az ASCII karakterekkel történő rajzolásra, amely a címkéknek köszönhetően a böngészőben is az itt látott képet fogja nyújtani. A címke elnevezés a **preformatted** angol kifejezésből ered.

```
1  <pre>
2  -      -
3  \ \ / /
4   \ ˇ /
5   /. .\
6  =\ I /=
7    ---
8  </pre>
```

9. Hosszú szavakban sortörés

A **<wbr>** páratlan, azaz üres címkével oldható meg a hosszú szavak törése, a címke neve **a word break** angol kifejezésből ered. Amennyiben úgy gondoljuk, hogy a szövegünkben lévő egy-egy hosszabb szó lehetséges, hogy nem fér el majd a böngészőablak egy sorában, úgy a <wbr> tag szövegbeiktatásával adhatunk meg olyan helyeket, ahol a szó törhető lesz. Amennyiben a szövegben szótörésre volna szükség a böngészőben való megjelenítéskor az említett a taggel annak helye megadható. A böngészőben ezeken a helyeken, szükség esetén megtörik a szó, de a sorok végére elválasztójel nem kerül. Ez a tag egy új HTML5-ös elem.

Léteznek még más szövegjelölő elemek is azonban ezek használata túl speciális ahhoz, hogy bekerüljenek ebbe a leírásba. Ezek a következőek: <bdo>; <ins>, <del>, <s>; <big>; <min>, <max>; <meter>, <kdb>, <ruby>, <var>. Sok pedig bekerült a leírásba, mint például a b, i, u, big, sub, sup, pedig ezek nem jelölnek tartalmi jelentéseket, inkább csak formázásokat érhetünk el velük, ez esetben CSS-ben is megadhatóak a megjelenések.

Amennyiben valamely címke használatával kapcsolatban kérdéseink merülnek fel, látogassuk meg a W3C School weboldalát, ahol mindig friss, pontos és részletes információkat kapunk az adott elemekről s azok attribútumairól.

A hasznos linkek:

- http://www.tutorial.hu/html5-es-css3/;
- http://www.w3schools.com/;
- http://www.w3schools.com/html/html5_intro.asp;
- http://www.w3schools.com/css3/default.asp.

5.2.2 Fontosabb HTML jelölők

1. Képek

A weboldalakon számos állókép található. Az **<img>** páratlan, üres jelölővel szúrhatunk be képeket weboldalunkra. A tag elnevezése az angol **image**, magyarul kép szóból ered. Kötelezően megadandó attribútuma az **src** (source, magyarul forrás), amelynek értéke a beszúrandó kép neve és annak relatív elérése az adott .html-hez képest.

Van még egy fontos attribútum: az **alt** attribútum, amelynek megadása nagyon erősen ajánlott, ennek akadálymentesség szempontjából van jelentősége. Az attribútum elnevezése az **alternative text** kifejezésből ered, amelyben a képnek alternatívaként egy pár szavas leírását adhatunk meg. Ez azok száméra fontos, akik valamilyen fogyatékosság okán nem látják a képeket az oldalon. A vakok és gyengénlátók a képernyő-felolvasó programok segítségével tájékozódnak a weboldalak tartalmáról, számukra amiatt nagy segítség a képekhez rendelt alternatív szöveg, mert ezeket fel tudják olvasni számukra a képernyő-olvasók. Az alternatív szövegek megadását mindig úgy kell megfogalmazni, hogy a lehető legrövidebben írja le a kép tartalmát, fontos, hogy a leírás vegye figyelembe a szöveg-kontextust. Ide kerülnek a képen található szövegek leírása is.

Például vegyük azt, hogy megjelenik egy fotó egy weboldalon, amely egy kézfogást ábrázol. Amennyiben például ez egy lakásbiztosítási

oldalon egy megbizhatóságot sugárzó fotó, úgy írjuk azt az alternatív szövegbe, hogy: „kézfogás, /megbízhatóságot tükröző fotó”. Ha pedig egy volt és a jelenlegi miniszterelnökök fognakkezet egymással, akkor fontos, hogy megnevezzük a képen szereplő kézfogó személyeket.

A kép szélességének és magasságának megadására a **width** illetve a **height** attribútumok szolgálnak, ezen értékeket pixelben kell érteni, a böngésző számokat vár. Ezekkel tudjuk megadni az adott kép böngészőben megjelenítendő méreteit, amelyek lehetnek nagyobbak vagy kisebbek is a kép eredeti méreteinél. Ha nem adunk meg ezeknek az attribútumoknak értéket, akkor a kép eredeti méretben jelenik meg a böngészőben. Ezzel szokás a fotógaléria bélyegképeit is létrehozni: nem tároljuk a kisméretű képeket, csak a nagyokat, melyek megjelenítési méretét a bélyegképeket tartalmazó oldalon width és height attribútumokkal lekorlátozzunk. Ezen attribútumok használatakor ügyeljünk a méretarányos értékek megadására annak érdekében, hogy a képek ne torzuljanak.

A **title** attribútumot akkor használjuk, ha azt szeretnénk elérni, hogy a képre húzva az egeret megjelenjen egy rövidke szöveg. Például nagyon hasznos, ha a weboldalon lévő logók <img> tagjében adunk meg title attribútumként egy „Vissza a főoldalra” szöveget.

Ez az elem is üres, nincs lezáró párja, tehát XHTML-ben a helyes megadás az, hogy még a hegyes zárójel bezárása előtt beírunk egy szóközt és egy perjelet.

```
1  <img
   src=„http://www.w3schools.com/images/w3html.png”
   alt=„w3c oktatóoldal logo” width=„100”
   height=„140” />
```

Korábban létezett egy igen népszerű attribútuma az <img> tagnek, nevezetesen a longdesc, amely a long description angol kifejezésből eredt, ebben megadható volt a képekhez egy hosszú leírás, amely nagyon hasznos volt abban az esetben, amikor nem volt elegendő az alt attribútum a szövegek leírásához. HTML5-től a longdesc attribútum nem támogatott.

Továbbá nem támogatott a korábban népszerű border tag sem, ezt a képformázási lehetőséget a HTML5 óta .css állományokban lehet megadni. Akkor

van haszna, amikor a képekre linket helyezünk, s meg szeretnénk előzni, hogy azt a böngésző automatikusan kékkel körbekeretezze jelezvén a kép linkelését. Ezt nem szeretjük, mert kicsit ízléstelen, a linkelhető képeket könnyen felfedezik az említett jelzés nélkül is a böngészők, hiszen ahogyan pásztázzák a tartalmakat, úgy haladnak az egérkurzorral is a monitoron, s amint egy hivatkozást is tartalmazó kép felé érnek az egérkurzor formája automatikusan megváltozik.

Képek szerepelhetnek listaelembeken (<li>), táblázatcellákban (<td>) és a hivatkozásokban (<a>) is.

2. Az ábra gyűjtőelem

HTML5-ben új elemként jött létre az **ábrák** gyűjtőeleme. A **<figure> … </figure>** címkepárok között képek, diagramok, ábrák, listák és a hozzájuk tartozó cím, felirat vagy magyarázat adható meg. A beágyazott elemeket és azok címét fogja közre a <figure> kulcsszó páros. Az elemek címe a **<figurecaption> … </figurecaption>** páros között adható meg. A beillesztett objektum és a címet a <figure> párok fogják össze. A figure angol kifejezés magyarul ábrát, a figurecaption **ábracímet** jelent. A weblapfejlesztő dönti el, hogy a cím a beágyazott elem felett vagy alatt helyezkedjen-e el, ennek megfelelően változik a beágyazott objektum s annak címének a sorrendje. Az alábbiakban két példa kódrészlet látható. Az elsőben egy képet helyezünk el, mely alatt található a címe, a másikban pedig egy listát, amely felett található a cím. Az objektumok igazítása CSS-sel történik.

```
1  <figure>
     <img src=„eyetracker.png”
     alt=„Szemmozgáskövető eszköz”>
     <figurecaption>SMI Szemmozgáskövető
     eszköz</figurecaption>
2  </figure>
```

```
1  <figure>
     <figurecaption>Felhasználói viselkedést tesz-
     telő eszközök
     </figurecaption>
     <ul>
       <li>szemmozgáskövető eszköz</li>
       <li>érzelemfelismerő szoftver</li>
       <li>EEG készülék</li>
     </ul>img src=„eyetracker.png”
2  </figure>
```

3. Hivatkozások

Az webes felületeket el sem lehetne képzelni hivatkozások (linkek) nélkül, hiszen a hypertextes felületeknek éppen az a lényegük, hogy az egyes különálló, önmagukban is értelmezhető szövegrészeket linkekkel kapcsoljuk össze. Természetesen nem csak szövegeket linkelhetünk egymásra, hanem a hiyperhálós szerkezetben részt vesznek még képek, videók, s hivatkozhatunk akár külső weboldalakra is.

A hivatkozásokat az **<a> … ** jelölőpárral lehet megadni, amelynek neve az **anchor** (magyarul horony) szóból ered. Kötelezően megadandó attribútum a **href**, amely értékeként azt adjuk meg, hogy **MI TÖLTŐDJÖN BE**, azaz mi töltődjön be a böngészőben megjelenő elemre (ami lehet szöveg vagy kép) klikkelve. A nyitó és záró <a> tagek fogják közre azt a szöveget vagy képet, **AMIRŐL LINKELÜNK**, ez a tartalom jelenik meg a böngészőben. Az érthetőség kedvéért nézzük a szerkezet általános alakját!

```
1  <a href=„MI TÖLTŐDJÖN BE?”>MIRŐL LINKELÜNK?</a>
```

Linkeléskor alapértelmezetten a weboldal helyére, azaz az aktuális ablakba töltődik be a tartalom. A böngészőben megjelenő hivatkozások szövege alapértelmezetten kék színű és aláhúzott. Amennyiben a link szövegére húzzuk a nyíl formájú egérkurzort, annak alakja egy mutató ujjra változik jelezvén, hogy tovább lehet arról a pontról ugrani. A szövegre klikkelve pedig betöltődik az az objektum, amire az <a> címke href attribútumában hivatkozunk.

3.1 Fájlokra hivatkozás

Ezzel a szerkezettel linkelhetünk fájlokra. A href attribútum értékeként bármilyen típusú állomány megadható, akár egy másik HTML dokumentum, akár egy kép-, hang-, vagy videoállomány. De linkelhetünk .pdf kiterjesztésű állományra is, vagy akár Power Point prezentációra. Ha olyan állományra hivatkozunk, amit az adott böngésző nem tud megjeleníteni, akkor az a hivatkozás segítségével winchesterre menthető. A következő példákban rendre egy másik .html dokumentumra való hivatkozás, egy .pdf állományra való hivatkozás és egy képre hivatkozás kódja látható.

```
1  <a href=„galeria.html”>Galéria</a>
```

```
1  <a href=„doksi.pdf”>Olvassa el a dokumentáci-
   ót!</a>
```

```
1  <a href=„foto.jpg”>
     <img src=„foto.jpg” width=„200” height=„150”
     alt=„Fotó egy virágról” />
2  </a>
```

Amennyiben az <a> címkék között egy képet adunk meg úgy az a kép megjelenik a böngészőben és annak teljes felülete érzékennyé válik. Ha a képre visszük az egérkurzort, akkor annak kézformára változó alakja jelzi, hogy a képen hivatkozás van. A kép bármely pontjára klikkelve betöltődik a hivatkozott objektum.

A példában ugyanaz a kép töltődik be eredeti méretben, mint amiről linkelünk (ez a fileok nevéből látható), csak a böngészőben megjelenő, hivatkozást hordozó kép mérete le van korlátozva. Így valósítható meg az, hogy egy bélyegképre klikkelve betöltődik annak normál méretű változata. Ekkor alapértelmezetten a kép körül egy kék színű keret jelenik meg, amelyet CSS-ben eltűntethetünk vagy színét módosíthatjuk.

3.2 Külső weboldalra (URL-re) való hivatkozás

Amennyiben egy külső weboldalra hivatkozunk a href attribútum értékeként egy weboldal URL címét kell, megadni, azaz annak kötelezően a „**http://**” vagy a „**https://**” kifejezéssel kell kezdődnie. Külső weboldalakat nem szokás a saját weboldalunk ablakában megnyitni (ez az alapértelmezés), hanem azok számára rendszerint új böngészőablak nyitandó. Ennek eléréséhez az ún. **target** attribútumban a „**_blank”** értéket kell megadjuk. Az alábbiakban egy példa látható.

```
1  <a href=„http://www.w3c.org” target=„_blank”>
      A W3C weboldala
   </a>
```

3.3 E-mail címre való hivatkozás

Hivatkozhatunk egy e-mailcímre is, ekkor a hivatkozott szövegre klikkelve levelet írhatunk a megadott címre, amennyiben a számítógépünkön ennek minden beállítási feltétele adott. Ez esetben a href attribútumban az e-mailcím megadása előtt közvetlenül a „**mailto:**” kifejezést kell beírni.

```
1  <a href=„mailto:valaki@ektf.hu”>Írjon e-
   mailt!</a>
```

4. Táblázatok

A táblázatok az összetartozó adatok egymáshoz való viszonyának az áttekinthető módon való megjelenítését biztosítják. A táblázatok sorokból és oszlopokból épülnek fel. A sorok és oszlopok találkozásaiban helyezkednek el az cellák, amelyek adatokat tartalmaznak. Tartalmazhatnak szövegeket, képeket, listákat, videókat, sőt újabb táblázatokat is.

A táblázat határoló-elemeiként a **<table> … </table>** címkéket használjuk. Ezeken belül az egyes sorokat a **<tr> … </tr>** párokkal adjuk meg, amelyeken belül helyezkednek el a cellák, a cellákat **<td> … </td>** párokkal jelöljük. Fontos, hogy a táblázat minden során belül ugyanannyi cellát adjunk meg, az üres cellák számára is kell <td> tagpárokat írni a kódba, ha nem így tennénk, akkor szétcsúszna a táblázatunk. A tag a nevét az angol **table** (magyarul táblázat) szóról kapta. A <tr> tag a **table row** kifejezésről, míg a <td> a **table data** kifejezésről kapta a nevét.

A <table> nyitótag után közvetlenül, még az első sor nyitó tagje előtt megadható az ún. táblázatcím a **<caption> … </caption>** jelölők között. Az ezek közé írt cím a táblázat tetején, annak egész szélességben, egyetlen egyesített cellában jelenik meg. A táblázat címének megadása opcionális. A <table> nyitótag záró, hegyes zárójele előtt, azaz még a címkén belül adhatjuk meg a **summary** attribútum értékét, mely egy általános leírást tartalmaz a táblázatról.

Az első sorok és az első oszlop is megjeleníthető **table header**, azaz táblázat fejlécként. Ezt úgy oldjuk meg, hogy a megfelelő <td> párokat <th> párokra cseréljük le, amelynek eredményeképpen azon cellák tartalma félkövéren lesz szedve és a cellában a szövegek vízszintesen középre igazítva jelennek meg. Alapértelmezetten a cellák tartalma (azaz a <td> tagpárban megjelenő szövegek) vízszintesen balra igazítottan, függőlegesen pedig középre igazítottan, normál betűstílusban jelennek meg.

A táblázat felépítéséhez fontos megérteni azt, hogy a HTML táblázatokban sorok vannak, azokon belül pedig a cellák sorfolytonosan elhelyezve egymás mellett, s azokban vannak a cellatartalmak. A tagek a táblázatban a hagyma szerkezetéhez hasonló módon vannak egymásba ágyazva. Belül a cellatartalom, aztán a cella címkéi, majd a soroké, majd végül legkívül a táblázat jelölője helyezkedik el.

Az alábbi képen látható, hogy a cellák egyes tartalmai miként jelennek meg a böngészőben.

```
<html>
<head>
  <title>teszt weboldal</title><!-- az oldal címe -->
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
  /><!-- ez a sor a magyar karakterkészletért felelős -->
</head>
<body><!-- háttérszín szövegszín -->
  <table>
    <tr>
        <td>A</td>
        <td>B</td>
        <td>C</td>
    </tr>
    <tr>
        <td>D</td>
        <td>E</td>
        <td>F</td>
    </tr>    <tr>
        <td>G</td>
        <td>H</td>
        <td>I</td>
    </tr>
</table>
</body>
</html>
```

A B C
D E F
G H I

21. ábra: A táblázat kódja és megjelenése a böngészőben

Az alábbi kódba már be van építve a táblázat címe, összefoglalója, s vannak fejlécek is.

```
<table summary=„Ide kerülhet egy rövid, szöveges
összefoglaló a táblázatról”>
  <caption>Ez a táblázat címe</caption>
    <tr>
        <th>A</th>
        <th>B</th>
        <th>C</th>
    </tr>
    <tr>
        <th>D</th>
        <td>E</td>
        <td>F</td>
    </tr>    <tr>
        <th>G</th>
        <td>H</td>
        <td>I</td>
    </tr>
</table>
```

Érdemes tudni, hogy a táblázat egymás alatti celláinak a szélessége és az egymás mellettiek magassága mindig megegyezik. Említettük, hogy az is fontos, hogy minden sorban ugyanannyi cella, azaz ugyanannyi <td> vagy <th> pár szerepeljen. Viszont néha szükségünk van arra, hogy egymás melletti vagy egymás alatti cellákat egyesítsünk, ezt a **colspan** (magyarul oszlopátfogás) illetve a **rowspan** (magyarul sorátfogás) attribútumok segítségével tehetjük meg. Mindkét esetben az attribútumok értékeként pozitív egész számot kell megadjunk. A sorok <tr> címkéibe írható be a rowspan kifejezés az oszlopok <td>-, illetve <th>-jaiba pedig a colspan kifejezés. Ha egymás melletti két cellát egyesíteni szeretnénk, akkor egy <td> párt kihagyunk, s az előző nyitó tagjében a colspan attribútum értékeként 2-t adunk meg.

Lássunk egy példát egy olyan táblázatra, melynek két sora van, s mindkét sorában egymás mellett két-két cella helyezkedik el. Az alsó sorban lévő kettő cela egyesítve van.

```
18 <table>
19     <tr>
20         <td>A</td>
21         <td>B</td>
22     </tr>
23     <tr>
24         <td colspan=„2”>Egyesített cella!</td>
25     </tr>
26 </table>
```

A táblázatok formája és megjelenése CSS állományban adhatóak meg.

5.2.3 A weboldal szakaszainak formázását elősegítő jelölők

A ** … ** páros jelölők egy logikailag összefüggő **szakaszt** jelölnek, amelynek tetszőleges tartalma lehet. Lehetnek benne szövegek, képek, táblázatok, hivatkozások videók vagy akár más elemek is. Több különböző, -vel kijelölt szakasz is megadható egy HTML dokumentumban, s mindegyikhez formázásokat lehet rendelni CSS-ben. A weboldalak vizuálisan megjelenő szerkezetét is ezekkel az elemekkel lehet meghatározni, pontosabban a -ek használatával szokás kialakítani a weboldalak megjelenését. A tag elnevezése a **division** (magyarul szakasz) angol szóból ered.

1. A tagek szerepe

Egy weboldalon rendszerint több -vel meghatározott szakasz is létezik, ezért ezek egyedi azonosítására van szükség, ezt tehetjük meg a tag **id** attribútumával, eredete **identifier**, magyarul azonosító. Az id attribútumok értékeinek egyedieknek kell lennie, azaz egy .html-ben egy id elnevezés csak egyszer szerepelhet. Az id-k értékeinek mindenképpen karakterekkel kell kezdődniük, ügyeljünk arra, hogy az elnevezésekben ne szerepeljenek speciális karakterek, s magyar ékezetes betűk sem, továbbá úgy válasszuk meg a rövid elnevezéseket, hogy azok utaljanak a tartalomra.

A HTML dokumentumokban a párokkal tehát csak szakaszokat definiálunk, s azokhoz az id attribútumokkal elnevezéseket rendelünk, ám míg azokra az elnevezésekre a CSS-ben nem hivatkozunk, addig az oldalon nem látható semmilyen változás, hiszen formát csakis a CSS-sel tudjuk adni oldalunknak. Az egyes szakaszokhoz tartozó tartalmaknak más-más .css állomány belinkelésével teljesen más-más megjelenítés adható. Változtatható a szöveges tartalmak karaktereinek betűtípusa, mérete, színe, a szakasz háttere, kerete, igazítása stb.

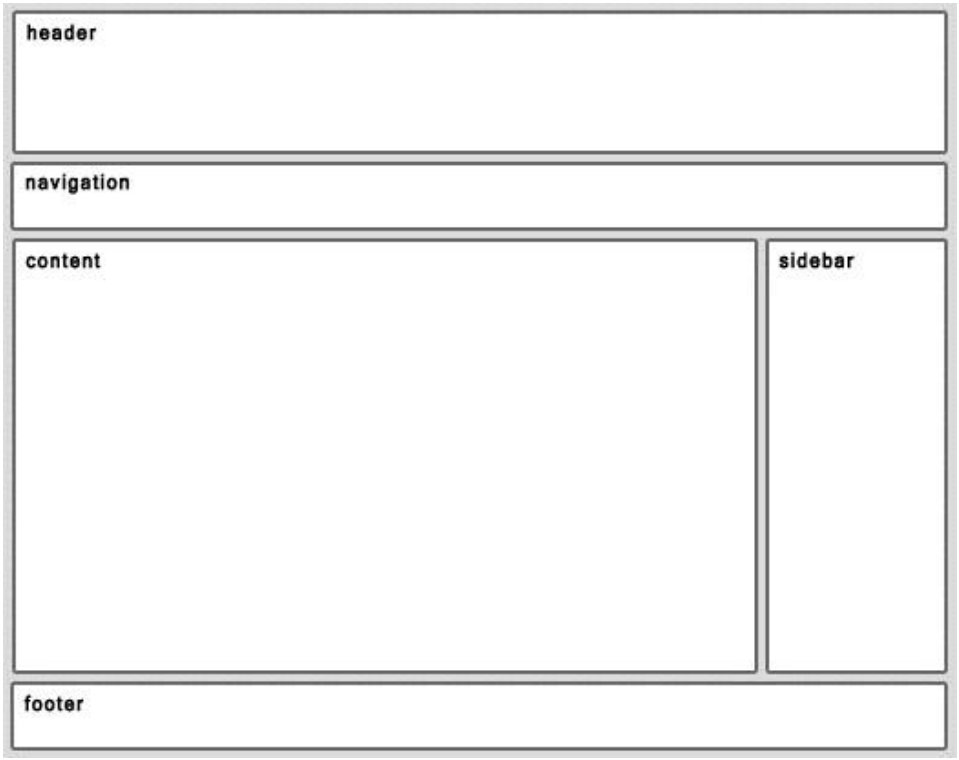
2. A weboldalak layoutjanak felépítése -ekkel

Lássunk példát arra, hogy miként szokás a HTML dokumentumokban jelölni a szakaszokat. Azaz mit kell tegyünk ahhoz, hogy -ekkel építsük fel a layoutunkat?

Mielőtt belefognánk a layout-készítés tárgyalásába vegyük sorra, hogy milyen egységekből (szakaszokból) épülnek fel nagy általánosságban a weblapok. A következő szakaszokra biztosan szükség lesz, ezeket jó ha angolul nevezzük el, mert ebben az esetben könnyen elkerülhető az a hibalehetőség, hogy magyar ékezetes betűket használunk.

- fejléc (=header)
- navigáció (=navigation/menu)
- tartalmi rész (=content/wrapper)
- jobboldali sáv (=sidebar)
- lábléc (=footer)

Nézzünk erre a felépítésre egy lehetséges HTML kódot, a kódban már csak a <body> ... </body> tagek közötti rész van leírva. A 22-es ábra szemlélteti a kódnak megfelelő struktúrát.



22. ábra: Egy általános weboldalnak a layout képe az egyes szakaszok neveinek feltűntetésével

```
1  <div id=„header”>
     <!-- Ide kerül a fejlécben szereplő elemek:
     logó, weblapcím, szlogen, egy szép képmontázs
     stb. -->
2  </div>
```

```
3  <div id=„navigation”>
     <!-- Ide kerülnek a menüpontok, s rájuk lin-
     kelve az egyes oldalak. -->
4  </div>
```

```
5  <div id=„content”>
     <!-- Ide kerülnek a weboldal fő tartalmi ré-
     szei, szövegek, képek, táblázatok, linkek
     stb. -->
6  </div>
```

```
7  <div id=„navigation”>
     <!-- Ennek a szakasznak a tartalma kerül az
     oldalsávba, például kisalkalmazások, naptár,
     címkefelhő stb. -->
8  </div>
```

```
9  <div id=„footer”>
     <!-- Ide kerülnek a láblécadatok, például az
     oldal készítőjnek a neve, elérhetőség, elérhe-
     tőségek, támogatók stb. -->
10 </div>
```

Az alábbiakban lássunk egy nagyon rövid példát arra, hogy miként hivatkozhatunk ezekre a HTML jelölőkre a .css állományban, lássuk a header (a fejléc) formázásának megadását kiragadva egy pici részt a .css egészéből!

```
#header {
   margin-top: 0px;
   margin-right: auto;
   margin-bottom: 0px;
   margin-left: auto;
   height: 80px;
   width: 713px;
}
```

3. Layout felépítés az új HTML5-ös tagekkel

HTML5-ben nagyon sok új elemet hoztak létre, nagy újítás az, hogy számos új elemet a weboldal felépítésének céljára hoztak létre, ezeket oldalszerkezeti elemeknek is nevezzük. Az új elemek mindegyike páros jelölő, lássuk ezeket: <header>, <hgroup>, <nav>, <section>, <aside> és <footer>. Egyértelmű az, hogy melyik jelölő mire utal, hiszen éppen olyan elnevezésű tageket hoztak létre újdonságként, amely elterjedtek a weboldalak -ekkel történő felépítésének id attribútum-értékeiként.

Az alábbiakban egy HTML5-ös kódot látunk, amelyben már az új oldalszerkezet jelölők vannak használva. Az előző layout képhez képest annyi változást kell csak elképzelni, hogy a „content” rész helyett van a „section”, s azon belül újabb szakasz található, melynek neve „article”, továbbá a jobboldali sávot ez esetben a „sidebar” helyett az ún. „aside” elnevezés jelöli, s a „navigation” helyett a helyes név ezúttal a „nav”.

```
1  <header>
2    <p>Ez a szöveg a fejlécben lesz látható.</p>
3  </header>
```

```
4  <nav>
5    <p>A menüpontok helye.</p>
6  </nav>
```

```
7  <section>
8    <p>Egy összefüggő szekció.</p>
9    <article>
10     <p>A szekción belüli cikk.</p>
11   </article>
12 </section>
```

```
13 <aside>
14   <p>A jobboldali sáv, benne általában funkciók.</p>
15 </aside>
```

1.

```
16 <footer>
17   <p>Ez a szöveg a láblécben lesz látható.</p>
18 <footer>
```

Lássuk egy rövid példán keresztül azt, hogy miként hivatkozhatunk a .css állományban az új oldalszerkezeti jelölőkre, a példábana .css állomány csak egy része, a fejléc formázásával kapcsolatos rész van kiragadva és a <body> tagre vonatkozóak.

```
body {
  background-color : #35643C;
  background-repeat : no-repeat ;
```

```
 margin-left: auto;
 margin-right: auto;
 margin-top: 0px;
 padding: 0px;
 width: 90%;
 max-width: 780px;
}

header {
 background: #133217;
 background-image:url('header.jpg');
 border: 0;
 text-align: right;
 font-weight: bold;
}

header > p{
 font-size:100%;
 text-align:right;
   color:#FFFFFF;
}
```

5.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

5.3.1 Összefoglalás

A fejezetben először áttekintésre kerültek a HTML5-ös szövegjelölő elemek, majd példákon keresztül megtekinthettük a leghasználatosabb szerkezeteket (képek, linkek, táblázatok), végül két speciálisabb jellegű ám elengedhetetlen fontosságú jelölőről esett szó.

5.3.2 Önellenőrző kérdések

1. Mondja el mire valók a szövegjelölő elemek?
2. Sorolja fel a szövegjelölő címkéket és röviden fejtse ki azt, hogy ezeket mire használjuk!
3. Sorolja fel az <img> HTML jelölő lehetséges attribútumait!
4. Az <a> jelölővel milyen jellegű linkek hozhatóak létre?
5. Mondja el, hogy miként lehet <table> és miként tagekkel felépíteni egy weboldal szerkezetét! Melyik megoldás elavult, s melyik ma használatos?

6. MÉDIAELEMEK KEZELÉSE A HTML-BEN

6.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Ebben a leckében az XHTML, illetve a HTML5 azon szolgáltatásait tekintjük át, amelyek segítségével a weboldalakon multimédiás tartalmakat jeleníthetünk meg. Az új technológiák lehetővé teszik, hogy a weboldalakon úgy tudjunk pl. zenét vagy mozgóképet lejátszani, hogy ehhez ne kelljen előzetesen külső sedégprogramokat, plug-ineket telepítenünk.

A leckében áttekintjük a multimédiás tartalmak megjelenítésének hagyományos módszerét, majd megvizsgáljuk az új technológia lehetőségeit a hagyományos módszerek kiváltására.

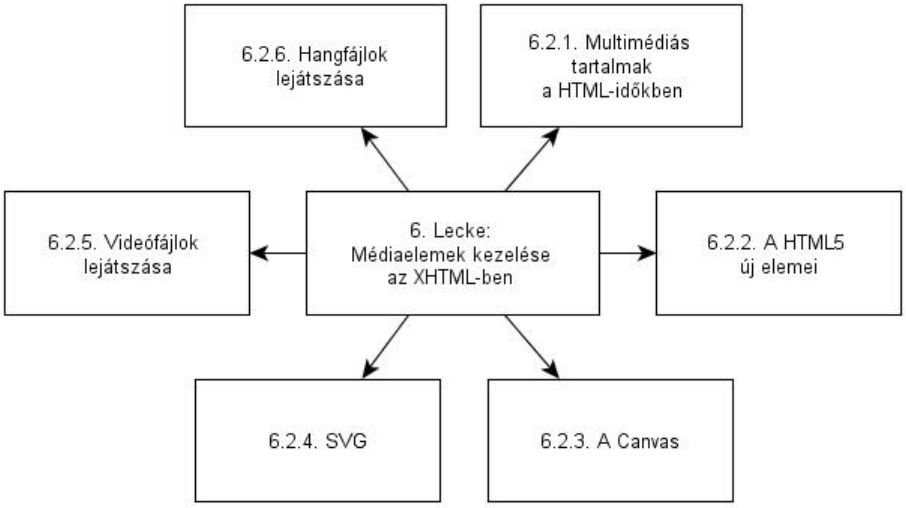
A lecke feldolgozásához szükséges kompetenciák az alábbiak:

Tudás: a lecke végén a hallgatók ismerni fogják a weboldalakon megjelenő multimédiás tartalmak használatának hagyományos, illetve az új technológia (HTML5) adta lehetőségeinek alapjait.

Attitűdök/nézetek: a lecke feldolgozása során belátjuk, hogy tökéletes megoldás egyelőre nincs az ilyen feladatok megoldására, de az új technológia mindenképpen reményt keltő a multimédiás tartalmak egyszerű és szabványos megjelenítését illetően.

Képességek: a lecke a következő képességeket fejleszti közvetlenül: áttekintő képesség, következtetési képesség, tervezési képesség, lényegfelismerés, rendszerben való gondolkodás, absztrakt gondolkodás.

6.2 TANANYAG



23. ábra: Fogalomtérkép

6.2.1 Multimédiás tartalmak a HTML-időkben

Amikor az 1990-es évek elején megjelent a world wide web, az internet (akkor) legújabb szolgáltatása, a fejlesztők elsősorban a tudományos élet szereplői számára kínálták a hozzá tartozó új dokumentumformátumot, a HTML-t. A cél egy olyan formátum létrehozása volt, amely hipertext alapú formázott szövegek leírását tette lehetővé. A formázott szövegeket állóképek is illusztrálhatták, az oldalak összekapcsolását pedig hiperlinkek tették lehetővé. A tudományos élet, a kutatók számára ez is hatalmas előrelépés volt az addigi szövegformátumokhoz és protokollokhoz (pl. Gopher) képest. Számukra talán még ma is elegendőek lennének a HTML első változatának szolgáltatásai, azonban az 1990 óta eltelt idő alatt kiderült, hogy ezek a szolgáltatások nem elegendőek a webes felhasználók igényeinek kielégítésére. Ennek elsősorban az az oka, hogy a www-t ma egészen mások és egészen másra használják, mint amire a ’90-es években kifejlesztették.

A megváltozott igények azt követelték a fejlesztőktől, hogy megoldást találjanak az olyan problémákra, amelyek megoldását a HTML közvetlenül nem tudja támogatni. A multimédiás weblapok esetében tipikusan ilyen feladat a hang- és zeneállományok, illetve a mozgóképek lejátszása. (Itt a mozgókép kifejezés alatt a hagyományos videókat, illetve az animációkat is értjük.) De ha az állóképekre gondolunk, a közvetlenül a weboldalra rajzolás lehetősége eddig

szintén nem állt rendelkezésünkre. Ha egy ábrát (pl. vonalas rajzot) szeretnénk illusztrációként elhelyezni a weboldalunkon, akkor azt a hagyományos technológia mellett egy rajzolóprogrammal kell elkészítenünk, majd fájlként elmentve ágyazhatjuk be a weboldalba.

A HTML és a www fejlődése során számos egyéb fejlesztés is született, amelyek idővel vagy megerődösdtek, vagy eltűntek. Ez utóbbi kategóriába sorolhatjuk a VRML-t (Virtual Reality Markup Language), amely segítségével virtuális világokat lehetett definiálni, pl. 3d-s helyszíneket, amelyeket a felhasználó ezután virtuálisan bejárhatott.

A HTML-ben közvetlenül nem támogatott szolgáltatások, pl. különféle médiaelemek lejátszását lehetővé tévő eszközök alkalmazására az utóbbi időkig a plug-inek és egyéb beépülő modulok, programok telepítse volt az egyetlen járható megoldás. Ezek között a legismertebb a Flash-állományok lejátszását lehetővé tévő FlashPlayer. A Flash jelenleg az Adobe cég tulajdonában van, korábban a Macromedia cég fejlesztette, amely pedig eredetileg a FutureWave cégtől vásárolta meg. A Flash kezdetben egy vektorgrafikus rajzolóprogram volt, amelyet később kiegészítettek animációkészítő lehetőségekkel is, mára pedig egy teljes platformmá nőtte ki magát. A Flash része egy szkriptnyelv is (ActionScript), amelynek verziószáma a jegyzet készítése idején a 3.0 volt.

A Flash-technológia rendkívül népszerűvé vált a www-használók körében. Flash segítségével olyan látványos vizualitást kaphattak a weboldalak, amelyek azelőtt elképzelhetetlenek lettek volna.

A Flash-állományokat a böngészők nem képesek lejátszani. Ahhoz, hogy egy weboldalba ágyazott Flash-animációt le lehessen futtatni, a böngészőprogramban egy plug-int kell telepíteni. (Hasonló az elv más multimédiás tartalmak megejelenítői, például a Shockwave esetében is.)

A Flash a hihetetlen népszerűség és látványos megjelenés mellett azonban rendelkezik néhány negatív tulajdonsággal is. Általában elmondható róla, hogy nagyon nagy az erőforrásigénye, ami régebben elsősorban annak volt az eredménye, hogy a Flashben a videokodekek optimalizációja nem volt tökéletes. Később ezen a problémán részben javítottak azzal, hogy a videók lejátszását a GPU segítségével hardveresen gyorsították. Ennél nagyobb gond az, hogy a teljesen Flashben készült weboldalak tartalmát a webes keresők nem tudják indexelni (pontosabban nem mindegyik tudja indexelni), így azok találati listájában sem (vagy csak kevésbé) jelennek meg az ilyen oldalak. A legfőbb probléma azonban az, hogy az egyre inkább terjedőben lévő okostelefonok és más mobil eszközök (pl. táblagépek) jelentős része egyáltalán nem támogatja a Flash-tartalmak lejátszását, így azok a fejlesztők, akik számítanak az ilyen eszközzel rendelkező felhasználók látogatására is a weboldalakon, nem használhatják a Flash-technológiát.

A plug-inek használata biztonsági kérdéseket is felvet. Ha egy plug-inben van biztonsági rés, az hibákat okozhat a böngésző egyébként biztonságos működésében.

A Microsoft saját eszközt készített Silverlight néven, amely a Flash lehetőségeihez hasonló szolgáltatásokat nyújtott, és többféle böngészőn (és platformon) is elérhető volt. De részben épp a HTML5 megjelenése miatt a Microsoft 2011 novemberében bejelentette, hogy befejezi a szoftver fejlesztését.

Amint azt látni fogjuk, a HTML5 számos, korábban nehézkesen vagy nem megbízható módszerekkel megoldható feladatra ad rendkívül egyszerű megoldást. Azonban azt is látnunk kell, hogy például a Flash „leváltása" HTML5-eszközökkel nem fog egyik napról a másikra megtörténni, ha egyáltalán be fog ez valaha is következni teljes mértékben. A Flash és a HTML5 egymással való helyettesíthetőségét illetően meglehetősen heterogén vélemények hallhatók szakmai berkekben. Az általános vélekedés szerint egyik technológia sem tudja teljesen kiváltani vagy helyettesíteni a másikat. Az mindenesetre biztosnak látszik, hogy a mobileszközök jelentős részén a Flash-technológia nem lesz elérhető a jövőben sem, ugyanakkor a HTML5 támogatottsága a különböző platformokon és böngészőkben jelenleg még nagyon eltérő szintet mutat.

6.2.2 A HTML5 új elemei

A HTML5 számos új elemet vezet be. Ezek részben az eddig közvetlenül nem támogatott feladatok megoldását segítik, részben pedig a már eddig is megoldható problémák megoldását egyszerűsítik. Az új elemek a következők:

1. *A HTML5 új elemei*

Jelölő	Leírás
<article>	egy cikket határoz meg
<aside>	az oldal tartalmához kapcsolódó tartalmat határoz meg
<bdi>	elkülöníti a szöveg azon részét, amely más irányban (jobbról balra író írások) van formázva, rendezve
<command>	egy parancsgombot definiál, amelyet a felhasználó meghívhat
<details>	további információkat definiálhat, amit a felhasználó megjelenít-het, elrejthet
<summary>	fejlécet definiál a <details> tag-hez
<figure>	elkülönített tartalmat határoz meg, például: illusztráció, diagram, fotó
<figcaption>	képaláírást definiál a <figure> elemhez
<footer>	láblécet határoz meg a dokumentum számára
<header>	fejlécet definiál a teljes dokumentumhoz vagy egy részéhez

Jelölő	Leírás
<hgroup>	<h1>-től <h6>-ig egy csoportot alkot a különböző címsorok definiálására
<mark>	megjelölt/kiemelt szövegrészt definiál
<meter>	ismert tartományon belül definiál egy skaláris mértéket
<nav>	navigációs linkeket tartalmazó blokkot definiál
<progress>	folyamat állapotát mutatja
<ruby>	kelet-ázsiai írásrendszerhez, megmutatja a karakterek kiejtését (Ruby annotation)
<rt>	a kelet-ázsiai karakterek kiejtését/jelentését definiálja
<rp>	definiálja, hogy mit jelenítsen meg a böngésző, ha nem támogatja a „Ruby annotációt”
<section>	a dokumentum egy bekezdését definiálja
<time>	dátum/idő definiálása
<wbr>	lehetséges sortörési pontokat definiál (hosszú sorok tördelésére)

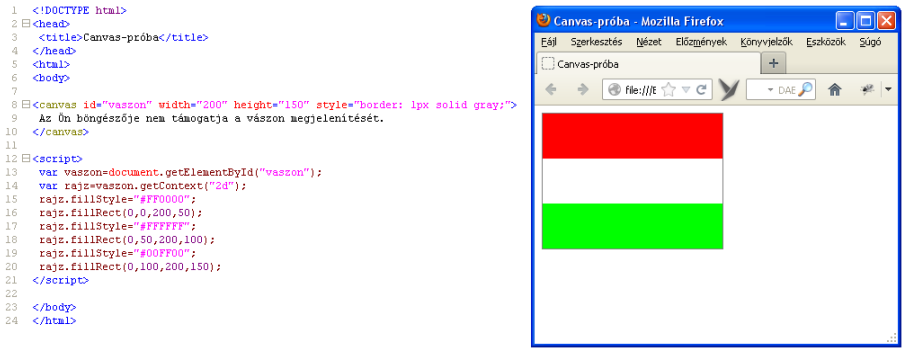
Új médiaelemek, vászonelem és új űrlapelemek:

2. *A HTML5 új médiaelemei*

Jelölő	Leírás
<audio>	hanganyagot definiál
<video>	mozgóképanyagot definiál
<source>	a hang, videó forrását definiálja
<embed>	egy tárolót definiál külső alkalmazás számára (pl. plug-in)
<track>	felirat vagy dalszöveg forrását definiálja
<canvas>	menet közbeni rajzolást tesz lehetővé (JavaScript segítségével)
<datalist>	beviteli mező, előre meghatározott lehetőségek listájával
<keygen>	titkosított kulcsszógenerátort definiál
<output>	kimenetet definiál számításokhoz

6.2.3 A Canvas

A Canvas (vászon) egy két dimenziós (síkbeli) bitképes rajzterület. A weboldalon grafikonok, webgrafikák, játékok képi elemei és egyéb képek rajzolására szolgál. A Canvasra való rajzoláshoz a HTML5-nek nincsenek eszközei, a rajzolás JavaScripttel valósítható meg.



24. ábra: Rajzolás JavaScripttel a Canvas elemre

A fenti példában látható zászlót az alábbi kód generálja:

```
<!DOCTYPE html>
<head>
 <title>Canvas-próba</title>
</head>
<html>
<body>

<canvas id=„vaszon” width=„200” height=„150”
   style=„border: 1px solid gray;”>
 Az Ön böngészője nem támogatja a vászon megje-
   lenítését.
</canvas>

<script>
 var vaszon=document.getElementById(„vaszon”);
 var rajz=vaszon.getContext(„2d”);
 rajz.fillStyle=„#FF0000”;
 rajz.fillRect(0,0,200,50);
 rajz.fillStyle=„#FFFFFF”;
 rajz.fillRect(0,50,200,100);
 rajz.fillStyle=„#00FF00”;
 rajz.fillRect(0,100,200,150);
</script>

</body>
</html>
```

A <canvas> (nyitó)elem attribútumai között meg kell adnunk a vászon szélességét és magasságát. Mivel a vászon kezdetben teljesen üres, még szegélye sincs, a fenti példában kapott egy 1 pixel vastagságú szürke keretet.

Ha valaki olyan böngészőt használ, amely nem támogatja a HTML5 elemeinek megjelenítését, akkor az ő képernyőjén a <canvas> és a </canvas> elemek közötti szöveg jelenik meg.

Magát a rajzot (a zászlót) JavaScript készíti. A zászló három darab 200×50 pixel méretű téglalapból áll. A szkript első sorában objektumot rendelünk a <canvas> elemhez (getElementById). Ebből az objektumból létrehozzuk azt az objektumot, amelyen a tényleges rajzműveleteket hajtjuk majd végre:

```
43 var rajz=vaszon.getContext(„2d”);
```

A getContent metódus kötelező paramétere a „2d” érték.

Ezután beállítjuk a zászlót alkotó három téglalap festőszínét (fillStyle metódus), majd létre is hozzuk őket (fillRect metódus).

Szakaszok rajzolásához (többek között) az alábbi metódusokat használhatjuk:

moveTo(x,y) – a képzeletbeli ceruza hegyét az (x;y) pontra mozgatja, a vászon bal felső sarka a (0;0) pont

lineTo(x,y) – az aktuális pontból az (x;y) koordinátájú pontba húz egy egyenes szakaszt, amely egyelőre láthatatlan

stroke() – a még láthatlan szakaszokat teszi láthatóvá, lényegében ténylegesen megrajzolja a vonalakat a beállított színű és vastagságú vonallal, amelyet a **strokeStyle()** metódussal tudunk meghatározni.

A használható metódusok teljes listáját és részletes leírását megtaláljuk például az alábbi oldalon: http://www.w3schools.com/tags/ref_canvas.asp.

6.2.4 SVG

Az SVG az angol Scalable Vector Graphics (vektorgrafikus ábrázolás) rövidítése. Vektoralapú grafikák XML formátumban való meghatározására használhatjuk. Nagy előnye, hogy a képek nem veszítik el képminőségüket átméretezéskor vagy ráközelítéskor. Egy SVG-fájl minden egyes eleme és tulajdonsága animálható.

SVG-t (mint bármilyen XML-t) tetszőleges szerkesztővel létrehozhatunk. A HTML5-ben az SVG-k beágyazása egyszerűen történik.

A canvas (vászon) és az SVG is képes grafikák létrehozására a webböngészőben, mégis alapvetően különböznek. Az SVG síkbeli képek leírására szolgál. XML-alapú, azaz minden az XML-ben leírt elem elérhető az SVG DOM-ban (Document Object Model). JavaScriptes eseménykezelőket is hozzá-

csatolhatunk az elemekhez (kattintás, rámutatás stb.). Az SVG-ben minden megrajzolt elem egy objektumként van eltárolva. Ha az SVG fájl tulajdonságai megváltoznak, a webböngésző automatikusan újrarendereli a formát.

A canvasra a (rendszerint) JavaScript segítségével szintén közvetlenül rajzolhatunk 2D képeket. A vászon azonban pixelről pixelre renderelt. Amint egy képet megrajzoltunk a vásznon, azt „elfelejti” a böngésző. Ha a helyzetét kell megváltoztatni, akkor az elemet újra ki kell rajzoltatni, beleértve azokat az objektumokat is, amelyeket esetleg eltakar a kép.

Lássunk egy egyszerű példát az SVG-vel való rajzolásra:

25. ábra: SVG-példa

A fent látható zászlót az alábbi kód generálta:

```
<!DOCTYPE html>
<head>
 <title>SVG-próba</title>
</head>
<html>
<body>

<svg xmlns=„http://www.w3.org/2000/svg” version=„1.1”>
  <rect width=„200” height=„50” x=„0” y=„0” style=„fill:#FF0000;stroke-width:1;stroke:#000000” />
  <rect width=„200” height=„50” x=„0” y=„50” style=„fill:#FFFFFF;stroke-width:1;stroke:#000000” />
  <rect width=„200” height=„50” x=„0” y=„100” style=„fill:#00FF00;stroke-width:1;stroke:#000000” />
</svg>

</body>
</html>
```

```
57 </body>
58 </html>
```

3. *Az SVG és a Canvas főbb tulajdonságainak összehasonlítása*

SVG	Canvas
felbontásfüggetlen	felbontásfüggő
eseménykezelők támogatása	eseménykezelők támogatásának hiánya
kiválóan alkalmas nagy renderelést igénylő alkalmazásokhoz (Google Maps)	gyenge szövegrenderelő tulajdonság
összetett dolgok lassú renderelése	az elkészült kép elmenthető (.jpg vagy .png)
nem alkalmas játékprogramokhoz	kiválóan alkalmas nagy grafikus teljesítményt igénylő játékokhoz, ahol sok tárgyat kell gyakran kirajzolni

6.2.5 Videófájlok lejátszása

A HTML5-ben videofájlokat a <video> elem segítségével ágyazhatunk be a weboldalba. Egyszerűen az src attribútummal hivatkozhatunk a fájlra:

```
59 <video src=„videofile.mp4” width=„320”
   height=„240”></video>
```

A böngésző a <video> elem által definiált doboz közepére fogja igazítani a videót, ha annak kiterjedése netán kisebb lenne. Alapértelmezettként a <video> elem nem jelenít meg semmilyen vezérlőelemet. Ezeket létrehozhatjuk egyszerűen JavaScript kód használatával. A <video> elem rendelkezik az alábbi metódusokkal: play(), pause(), egy írható/olvasható tulajdonsággal (currentTime) és némítással (muted). Ha azonban nem szeretnénk saját felületet építeni, a böngésző beépített vezérlőit is megjeleníttethetjük. Ez a controls attribútummal lehetséges.

```
60 <video src=„videofile.mp4” width=„320”
   height=„240” controls></video>
```

Lehetőségünk van a böngészőt arra utasítani, hogy az oldal betöltődése után kezdje el rögtön letölteni az adott videót. Ehhez a preload jellemzőt kell használnunk. Ha ezt a lehetőséget nem kérjük, akkor a none értéket kell megadni.

```
61 <video src=„videofile.mp4” width=„320”
   height=„240” preload></video>
```

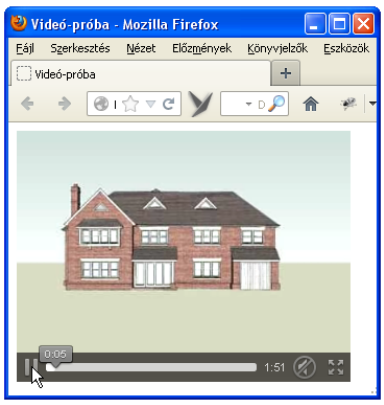
A másik lehetőség, ha a weboldal betöltődését követően nem csak a videó letöltését szeretnénk elérni, de azonnali lejátszását is kérünk. Ezt az autoplay attribútum segítségével tehetjük meg.

```
62 <video src=„videofile.mp4” width=„320”
   height=„240” autoplay></video>
```

Többféle formátumot is támogat a HTML5, a böngészők azonban nem mindegyiket. A HTML5 rendelkezik egy olyan <source> elemmel, amellyel a <video> elem tetszőleges számú videofájlra mutathat. A böngésző végigmegy a listán, és kiválasztja azt a formátumot, amelyet képes lejátszani. Érdemes megadni egy type jellemzővel, hogy milyen videofájlokra hivatkozunk, elkerülve ezzel azt, hogy a böngésző az összes videót elkezdje letölteni, és ezzel hatalmas erőforrásokat és sávszélességet emésszünk fel. A type jellemző három információt tartalmaz: a tárolóformátumot, a video- és az audiokodeket. (A kodekek olyan algoritmusok, amelyek a video- és hangfolyam kódolását határozzák meg.)

```
63  <video width =„320” height=„240” controls>
64  <source src=„videofile.mp4” type='video/mp4;
   codecs=„avcl.42E01E, mp4a.40.2”'>
65  <source src=„videofile.webm” type='video/webm;
   codecs=„vp8, vorbis”'>
66  <source src=„videofile.ogv” type='video/ogg;
   codecs=„theora, vorbis”'>
67  </video>
```

```
1  <!DOCTYPE html>
2  <head>
3   <title>Videó-próba</title>
4  </head>
5  <html>
6  <body>
7
8  <video src="video.ogv" width="320" height="240" controls></video>
9
10 </body>
11 </html>
```



26. ábra: Videó lejátszása a böngészőben

6.2.6 Hangfájlok lejátszása

A videók lejátszásához nagyon hasonlóan van módunk hangfájlok lejátszására is a HTML5 segítségével. Ehhez az <audio> jelölőt kell használnunk.

```
68 <audio controls=„controls”>
69   <source src=„zene.mp3” type=„audio/mpeg”>
70 </audio>
```

A hangfájlok formátuma mp3, ogg vagy wav lehet, de nem minden formátumot támogat minden böngésző. Érdemes ezért a videó lejátszásánál leírt módon többféle formátumban is feltölteni a webszerverre a hangfájlt, és az <audio> jelölőben minden formátumnak megadni az elérési útját. Ilyenkor az aktuális böngésző megvizsgálja, hogy milyen formátumot tud lejtászani a felsoroltak közül és azt a formátumot tölti le, majd játssza le.

Ahogyan a videók esetében, itt is van módunk arra, hogy vezérlőelemeket jelenítsünk meg a hangfájl lejátszásához. (Ezt a fenti példában is megtettük.)

Az autoplay és a preload attribútumok az <audio> elemnél is elérhetők és használhatók.

6.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

6.3.1 Összefoglalás

Ebben a leckében áttekintettük a multimédiás tartalmak (kép, hang, videó) weboldalkon való megjelenítésének hagyományos és a HTML5 új lehetőségeit kihasználó módszereit. Összehasonlítva a Flash-re épülő technológiát a HTML5 lehetőségeivel, azt állapíthatjuk meg, hogy a két módszer jól kiegészíti egymást, de jelenleg semmiképpen sem állíthatjuk, hogy az egyik száz százalékig ki tudná váltani a másikat.

A HTML5 megalkotásának egyik fő célja az volt, hogy a böngészőbe kényszerből telepítendő plug-inek számát csökkenteni lehessen. Ebből következően a HTML5 képes hang- és mozgóképtartalmak lejátszására, valamint vektorgrafika megjelenítésére. A HTML5 önmagában azonban nem képes rajzolásra is, ehhez mindenképpen szkriptnyelven (első sorban JavaScriptban) való programozásra is szükség van.

Mindenképpen a HTML5 elterjedését fogja segíteni, hogy sok mobil eszköz egyáltalán nem támogatja (és nem is fogja támogatni a jövőben sem) a Flash-technológiát. Ezeken bizonyos tartalmak kizárólag a HTML5 révén lesznek lejátszhatók. A HTML5 szélesebb körű elterjedését ma azonban még korlátozza az, hogy a különböző böngészők meglehetősen különböző szinteken kezelik a HTML5 elemeit.

Meg kell tehát állapítanunk, hogy tökéletes megoldás jelen pillanatban nincs. A HTML5 új szolgáltatásai azonban mindenképpen reménykeltőek a web használói számára.

6.3.2 Önellenőrző kérdések

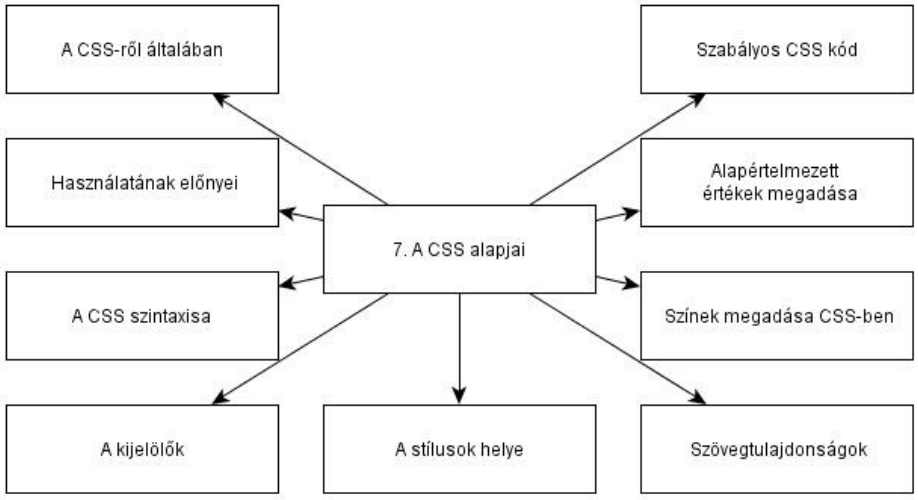
1. Mi az a plug-in? Miért van szükség Flash lejátszóra számos weboldal megtekintéséhez?
2. Mi lehet az oka a Flash sikerének? Milyen negatívumai vannak a Flash-technológiának?
3. Milyen eszközei vannak a HTML5-nek a weboldalra való közvetlen rajzolás terén? Mi a különbség ezek között az eszközök között?
4. Hogyan támogatja a HTML5 a videók lejátszását a weboldalainkon?
5. Hogyan támogatja a HTML5 a hang- és zenefájlok lejátszását?

7. A CSS ALAPJAI

7.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Cél megértetni a stílusok használatának a fontosságát az olvasóval, láttatni azt, hogy a stílusok alkalmazásával mennyire rugalmasan kezelhetővé válik egy weboldal megjelenése, mennyivel könnyebb az oldalak megjelenésének a kialakítása és módosíthatósága. A külső CSS állományokkal lerövidíthető a letöltések ideje és átláthatóbbá válik a kód. Cél a kijelölők és a színek CSS-ben való használatának a megértetése.

7.2 TANANYAG



27. ábra: A leckéhez tartozó fogalomtérkép

7.2.1 A CSS-ről általában

A **CSS** az angol **Cascading Style Sheets** kifejezés rövidítése, ami rangsorolt stíluslapokat jelent. Ez egy külön szabályrendszer, ami a weblapok megjelenésének kialakítására használatos. Maga a nyelvezet független a HTML-től de mégsem teljesen független tőle, mert azzal együttműködve használható. A HTML dokumentummal valamilyen módon össze kell kötni beágyazás útján, vagy csatolva. Az újabb böngészőkben lehetőség van rá, hogy a felhasználó választhasson egy stílust a felkínáltak közül, vagy akár a saját stílusát alkalmazza

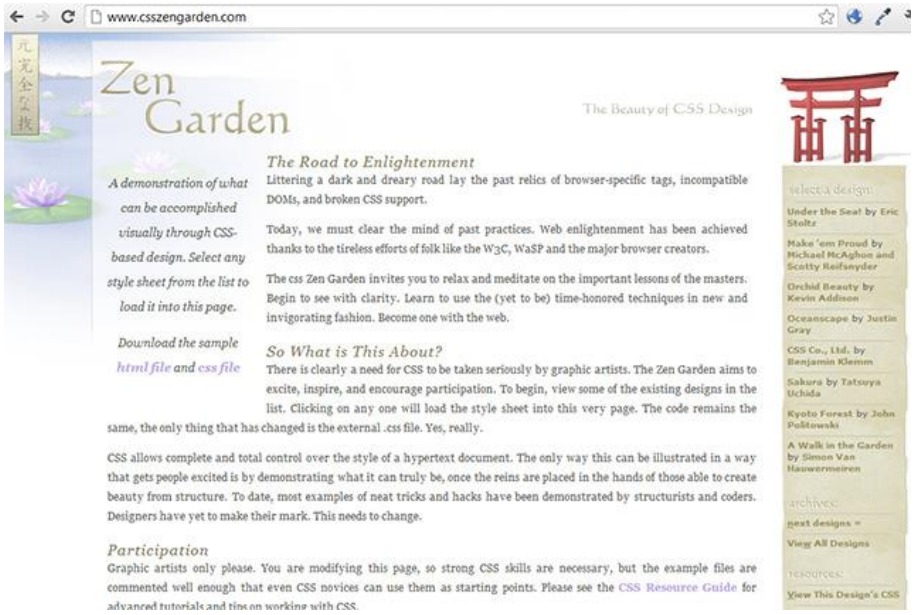
az adott oldalra. A felhasználó tehát felülbírálhatja az oldalon lévő stílusokat, valamint a böngésző is rendelkezik egy alapértelmezett stílusrendszerrel. Az alapértelmezett stílusok böngészőnként eltérhetnek, így gyakran azokat is be kell állítanunk, hogy minden felületen ugyanazt az eredményt kapjuk.

7.2.2 Használatának előnyei

A stíluslapok segítségével megváltoztathatók a leírókodon belüli elemek tulajdonságai. Segítségével beállíthatjuk a színeket, háttereket, szegélyeket, térközöket, valamint az elemek helyét az oldalon. Elhelyezhetjük a HTML dokumentumban, vagy akár önálló fájlban is, ilyenkor viszont kapcsolnunk kell az adott oldal kódjához. Ha külső stíluslapot használunk, akkor megvalósíthatjuk azt az elképzelést miszerint a tartalom elválik a megjelenéstől, így az innentől függetlenül kezelhető attól. Másik fontos tulajdonsága az önálló CSS dokumentumnak, hogy hozzárendelhetjük több oldalhoz is, így ha változtatunk benne, akkor az összes többi oldalra hatással lesz, és nem kell mindegyikben megváltoztatnunk az adott beállítást. Mindezek mellett a tartalomhoz sem kell hozzányúlnunk, így az változatlan marad.

Karbantartások és frissítések során nagyon sok időt lehet vele megtakarítani, így **nagy rugalmasságot biztosít az elkülönített stíluslap**. A stílusok használatával az oldal betöltésének ideje is lecsökken. A böngészők a gyorsítótárba mentik ezeket, így ha legközelebb hivatkozást talál erre a fájlra, nem tölti le még egyszer, hanem a gyorsítótárból veszi ki azt, így időt és sávszélességet takarít meg.

Nagyszerű kezdeményezés a **Csszengarden** oldal, ami arra az ötletre épül, hogy egy adott tartalom mellé bármennyi megjelenést párosíthatunk a tartalom megváltoztatása nélkül. Itt egy XHTML kód van közreadva és annak módosítása nélkül kell megoldani a weboldal megjelenésének beállítását tisztán css-sel és persze a vonatkozó grafikai elemekkel.



28. ábra: csszengarden.com

7.2.3 A CSS szintaxisa

Ahhoz, hogy egy stílust érvényesítsünk egy elemre szükségünk lesz hozzá kijelölőkre és meghatározásokra. A **kijelölők** segítségével megjelölhetjük az adott elemet, amelynek tulajdonságait szeretnénk megváltoztatni, azaz megmondják, hogy a HTML oldal melyik elemére vonatkozik. A legegyszerűbb fajtája az **elemkijelölő**, amely egy XHTML elemet jelez, mint például a **p**.

A kijelölőkhöz meg kell írnunk a **meghatározást**, ami megadja a kijelölt elem megjelenését a lapon. Ez a két komponens adja meg a stílust.

```
1  p {
2    color: green;
3    font-size: 14px;
4  }
```

A fenti kódrészleten láthatjuk, hogy a kijelölő a **p**, amely az összes bekezdésre hivatkozik. A kapcsos zárójelek között ({}) a stílus meghatározó része van, amely ebben az esetben két meghatározást ad meg.

A meghatározások **egy tulajdonság és egy érték páros**. A tulajdonságok után kettőspontot és egy szóközt írunk, az érték után pedig egy pontosvessző kerül, amelynek szerepe a meghatározások különválasztása. A példában az első tulajdonság a bekezdések betűszínére vonatkozik, a második pedig a betűméretet állítja be.

7.2.4 A kijelölők

Ha az elemekre vonatkozó kijelölőket használjuk, akkor az minden olyan elemet azonosít az oldalon. Gyakran ennél egyértelműbben kell meghatározni, hogy melyik részt szeretnénk beállítani. Azoknak a kijelölőknek, amelyeknek pontosabban kell megadnia a formázandó elemet két csoportja van, az **azonosítókijelölő és az osztálykijelölő**. Ezekkel megoldhatjuk, hogy az összes bekezdés formázása helyett csak egyetlen vagy egy csoportba tartozó bekezdéseket formázzunk.

Az **azonosítókijelölő** elé egy kettőskereszt (**#**) jelet írunk. Ilyenkor egy konkrét elemnek tudjuk módosítani a tulajdonságait, mégpedig annak, amelyet az XHTML kódban az azonos **id** attribútum jelöli.

```
5  #tartalom {
6    color: red;
7  }
```

A példában megfigyelhetjük, hogy a kijelölő neve elé kettőskeresztet írtunk, amit nem választunk el szóközzel a kijelölő nevétől. Az azonosítókijelölőket mi nevezzük el, célszerű olyan nevet adni neki, amely legjobban leírja, hogy mire szeretnénk használni. A példában a „tartalom” névvel azonosított részére hivatkozunk a weboldalnak és az összes benne előforduló szöveg színét vörösre állítjuk.

Abban az esetben, ha nem akarunk minden szöveget ilyen színűre állítani, akkor **környezetfüggő kijelölőt** használunk, ami az azonosított rész csak adott szakaszaira vonatkozik.

```
8  #tartalom p {
9    color: red;
10 }
```

A kijelölőt kibővítettük, mégpedig az azonosítókijelölő után szóközzel elválasztva egy elemkijelölőt írtunk (p). Ez a „tartalom” kijelölővel azonosított részben előforduló összes bekezdésre hivatkozik.

Az **osztálykijelölő** annyiban különbözik az azonosítókijelölőtől, hogy az több elemre is hivatkozhat az oldalon. Az osztálykijelölő nevét is nekünk kell

kitalálni. Akkor érdemes ezt választanunk, ha tudjuk, hogy az oldalon több hasonló elem is elő fog fordulni.

```
11 .kiemeles {
12   font-weight: bold;
13   color: green;
14 }
```

A kijelölő neve elé pontot írunk szóköz nélkül. Ez a stílus félkövérre és zöldre állítja az összes „kiemeles” osztállyal ellátott elemet. Ezt is tovább kombinálhatjuk elem és azonosítókijelölővel.

```
15 p.kiemeles {
16   color: darkgreen;
17 }
```

A fenti példában egy „p” jelölő jelzi azt, hogy csak azok a bekezdések színe legyen sötétzöld, melyeket elláttunk a „kiemeles” osztállyal;

Ha egy adott stílust az oldal több különböző elemére is szeretnénk érvényesíteni, megtehetjük úgy, hogy minden elemhez létrehozzuk ugyanazt a stílust, de van egy egyszerűbb és gyorsabb módja is. Ebben az esetben a kijelölőket csak fel kell sorolnunk vesszővel elválasztva, majd leírjuk a szükséges tulajdonságokat.

```
18 p, blockquote {
19   font-size: small;
20 }
```

Ez a stílusleírás a bekezdések és idézetblokkok szövegének betűméretét is kicsire állítja. A vesszőnek fontos szerepe van, hiszen ha elhagyjuk, akkor csak a bekezdésekben lévő idézetblokkokat azonosítanánk. Ez is mutatja, hogy célszerű nagyobb figyelmet fordítani a helyes írásmódnak, így később sok kellemetlenségtől kímélhetjük meg magunkat.

Elnevezés

Az azonosítók és osztályok neveinek megválasztásánál ügyeljünk azok megválasztására. Ezek az elnevezések többre szolgálnak, mint a CSS-ben való felhasználhatóság. Az adott elemeket eszerint megjelölhetjük a különböző funkcióik vagy az oldalon betöltött szerepük alapján. Tehát ha egy szöveget figyelmeztető jelleggel vörös színűre szeretnénk beállítani, használjuk a „figyelmeztetes” osztályt ahelyett, hogy „piros”-at adnánk meg neki. A szerepének jelölésén felül, ha például a későbbiekben megváltoztatjuk a figyelmeztetés szöveg színét, akkor a megadott „piros” név már nem is lesz releváns.

Továbbá az oldal szerkezetének kialakításakor sokkal egyszerűbb „#fejlec”-re és „#lablec”-re hivatkozni, mint „#elem1”-re és „#elem2”-re.

7.2.5 A stílusok helye

A CSS elnevezésénél a **Cascade** szó rangsorolásra utal, tehát nem mindegy hogy az adott stílus hol helyezkedik el a rangsorban. Az ütközések elkerülése végett még pár egyszerű szabály létezik. Az öröklés az XHTML elemek egymásba ágyazottságán alapul, így az egymásba ágyazott elemek öröklik az őket tartalmazó elem beállításait. A szűkítésnek a stílusok súlyozásánál van szerepe. Azaz ha meg akarjuk különböztetni egy stílus fontosságát, akkor szűkebben, vagyis pontosabban kell megadnunk azt, így vehetjük rá a böngészőt, hogy azt használja.

A stílusok elhelyezkedésüktől függően fontossági sorrendben állnak.

1. Böngészőstílus
2. Felhasználói stílus
3. Külső stílus
4. Beágyazott stílus
5. Szövegközi stílus

Fentről haladva egyre nagyobb jelentőséggel bírnak a stílusok. A **böngészőstílus** azt határozza meg, ha nem adtunk meg külön stíluslapot az oldalhoz, akkor hogyan jelenjen meg az elem. A **felhasználói stílusok** a böngészőben állíthatóak be. Általában senki nem szokott ilyen beállításokat alkalmazni, és nem is nagyon ismerik ezeket.

Külső stílusokról akkor beszélünk, ha az külön fájlban található meg. Ez az első szint ahol fejlesztőként beavatkozhatunk az oldal megjelenésébe. A hivatkozást ebben az esetben a HTML kód fejlécében kell elhelyeznünk a title elem után.

```
21 <link href=„stilus.css” rel=„stylesheet”
   type=„text/css” />
```

Amint ezt beírtuk a html kódba a böngésző innentől fogva keresni fogja a stilus.css állományt, amelybe mi a stílusokat helyeztük el. A külső stílus felülbírálja a böngésző által megadott stílusokat. Ha bármikor változtatunk a stíluson, akkor frissíteni kell a böngészőben az oldalt.

Másik megoldás, ha az @import utasítást használjuk külső css beáágyazására.

```
22 <style type=„text/css”>
```

```
23   @import url(stilus.css)
24 </style>
```

Többszörösen is használható, azaz egyidejűleg több stílust is adhatunk az oldalnak. Nem tér el nagyban az előző módszertől, viszont régebbi böngészők ezt nem ismerik, így nem is fognak foglalkozni a stílusokkal, azaz egyszerűen átugorják ezt a részt. Az itt megadott stílus származhat más webhelyről is nem kell a HTML dokumentum mellett lennie. Az @import utasítás a style elemben helyezkedik el, valamint tartalmazza a beolvasandó stílus fájl címét. Ha a style elembe más belső stílust is használni akarunk, akkor a beágyazásnak kell elől lennie.

A beágyazásnál lehetőségünk van megadni azt is, hogy a különböző megjelenítési felületeken a böngésző melyik stíluslapot használja. Ilyenek az all (minden), screen (képernyő), print (nyomtatás), projection (kivetítő).

Beágyazott stílusok esetén, azok csak arra érvényesek, amelyik dokumentumba beágyaztuk azokat. Akkor célszerű ilyet használni, ha az oldalunk csak egy oldalból áll, vagy ha még csak fejlesztjük az oldalt és könnyebb elérést akarunk biztosítani eszközeinknek. A beágyazott stílusok is a head részbe kerülnek a HTML kódban a title elem után.

```
25 <style type=„text/css”>
26   p {
27  font-weight: bold;
28  color: green;
29   }
30 </style>
```

A stílusokat a style elembe írjuk. Ez súlyozottabb a külső stílusnál, így a head részben ezt az esetleges külső hivatkozás alá írjuk. Régebben a beágyazott stílusoknál megjegyzéseket használtak, mert nem minden böngésző ismerte azokat, manapság már az összes modern böngésző ismeri és megjeleníti ezeket.

A **szövegközi stílust** a HTML kódban helyezzük el abban az elemben, amelyet formázni szeretnénk. A rangsorban ez van a legközelebb az elemhez így ez áll a legmagasabb szinten. Mivel ez a technika a HTML kódot gyarapítja, nagyobb mennyiségben kihathat a sávszélesség kihasználtságára és a letöltési időre is, így csak speciális esetekben használjuk. Ezeket mindig az adott elem style attribútumába írjuk be ahol az érték maga a meghatározás lesz.

```
31 <p style=„color: #0F0”>Ez a bekezdés zöld szö-
   veggel jelenik meg</p>
```

A stílus csak az adott elemre hat ki, esetünkben a többi bekezdést változatlanul hagyja. Az esetlegesen máshol elhelyezkedő más beállításokat változatlanul hagyja.

7.2.6 Szövegtulajdonságok

Tulajdonság	Learás	Megadható értékek
background-color	Az adott elem háttérszíne	A szín megnevezése vagy színkód
color	Szöveg színe	A szín megnevezése vagy színkód
font-family	Betűtípus vagy betűcsalád név	családnevek: Times, Helvetica, Zapf-Chancery, Western, vagy Courier. általános nevek: serif, sans-serif, cursive, fantasy, vagy monospace
font-size	A betűtípus nagysága	abszolút-méret | relatív-méret | hossz | százalék. Alapérték: medium (közepes). Az abszolút méret számokkal értelmezhető vagy kulcsszavak: xx-small | x-small | small | medium | large | x-large | xx-large. Relatív méret lehetséges értékei: larger | smaller. Hosszúság: Egy abszolút (szám)érték a méret részére. A százalékérték a szülőelem %-ában értendő. Csak egész szám lehet, melyet közvetlenül követ a %-jel
font-style	A betű stílusa	normal (= normál; alapértelmezett), italic (= döntött), oblique (= ez is döntött)
font-weight	A szöveg vastagsága	normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900. (Érthetőbben: 400=normal / normális; 700=bold / kövér)
line-height	A szöveg sorának magassága	normal (alapértékek), szám, hossz, százalék

Tulajdonság	Learás	Megadható értékek
text-align	A szöveg igazítása	left (balra; alapértelmezés), right (jobbra), center (középre), justify (hasábszerűen, de ez nem mindenhol működik megfelelően - ráadásul lassú!)
text-decoration	Megadható, hoyg a szöveg aláhúzott vagy áthúzott	none (semmi), underline (aláhúzott), overline (föléhúzott), line-through (keresztülhúzott), blink (villogó, pár böngésző nem támogatja)

7.2.7 Színek megadása CSS-ben

Több tulajdonság értékeként megadható szín vagy színkód. Ezt többféleképpen megtehetjük. A leg kézenfekvőbb megoldás angol nyelvterületeken a színek megadása azok neve alapján. A színek ezen módon történő definiálása azonban limitált, ugyanis nincs minden árnyalatnak név megfeleltetve.

A következő mód a közvetlen RGB szín meghatározása. Tekintve, hogy a weboldalak elsődleges megjelenítési formája a számítógép vagy mobil eszköz kijelzője, így az RGB színmodell használata a kézenfekvő. Ebben a megközelítésben.

- R (angol red rövidítése) - vörös
- G (angol green rövidítése) - zöld
- B (angol blue rövidítése) - kék

A három szín érték egy 0-tól 255-ig tartó skálán adható meg egymás után. Minél nagyobb az érték annál dominánsabb az adott szín. Tehát a (255,0,0) a tiszta vörös színkombinációja.

A harmadik megoldás szintén a szín RGB színkombinációjának használatával működik. Ebben az esetben azonban hexadecimális értékekkel határozható az meg. Alapértelmezésben a szín hat karakterből áll előtte egy kettőskereszt szimbólummal ami jelzi a böngészőnek a hexadecimális érték használatát. A hexadecimális azaz 16-os számrendszer használata feltételezi, hogy egy helyi értéken 16 különböző számot tudunk megjeleníteni. Ebből az okból kifolyólag a 9 fölötti értékeknél az ABC betűi vannak használva, tehát 10-16 az A-F értékek. Ha hat számjegyet használunk például: #0000ff, két helyi érték határoz meg egy színt. Az első kettő a vörös, a második kettő a zöld, a harmadik kettő pedig a kék. A példában használt szín tehát a kék lesz mivel annak az értéke a maximális míg a másik kettőé a minimum érték. Abban az esetben, ha a különböző színek

értéke megegyezik egymással, akkor rövidíthető a színkód. Így a #55bb00 megadható #5b0 formában is. Nem kell azonban megjegyezni a színek kombinációit, a legtöbb esetben azt megtudhatjuk táblázatokból, de leggyakrabban a grafikai szoftverek árulják el a megfelelő értékeket.

A tiszta kék szín például megadható a következő módokon:

```
32 a { color: blue }
33 a { color: #0000ff }
34 a { color: #00f }
35 a { color: rgb(0,0,255) }
```

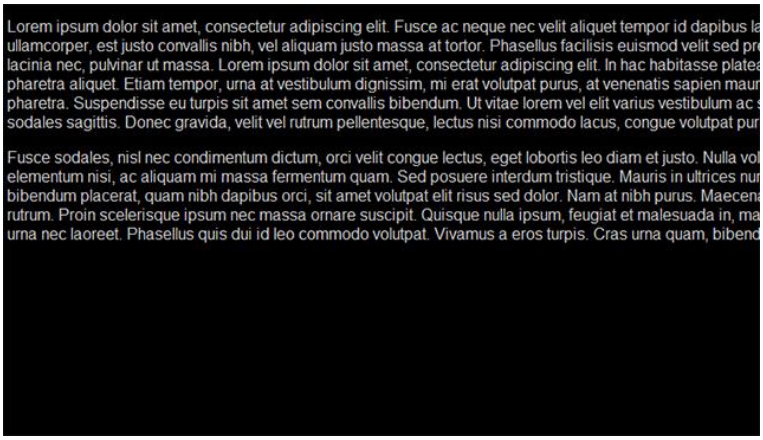
7.2.8 Alapértelmezett értékek megadása

Bármely böngészőben ha megnézünk egy tisztán XHTML oldalt amely nem rendelkezik CSS beállításokkal azt fehér háttérrel és fekete betűkkel láthatjuk. Ebben az esetben a hivatkozások kék színűek. Ezt azonban csak általánosságban mondhatjuk el, ez korántsem garancia, ugyanis amit láthatunk az a böngésző alapértelmezett CSS beállításai. A betűméretek és köztes területek méretei böngészőnként eltérhetnek, így nem hagyatkozhatunk ezekre a beállításokra, tehát javasolt saját magunknak újradefiniálni ezeket a beállításokat.

Ha weboldalt készítünk, elsők között merül föl az igény az általános szöveg megjelenés beállításokra. Azaz megadhatjuk globálisan az oldalon használt szövegek színét, illetve az alkalmazott betűtípusokat anélkül, hogy utána minden egyes elemnek külön meg kellene azokat határozni. A CSS-ben létezik öröklődés, tehát bizonyos tulajdonságokat a szülő XHTML elemektől átvesznek az elemek. Ebben az esetben keresni kell egy olyan XHTML elemet, amely minden másnak a szülője és megjelenik a böngészőben, ez pedig a **body** lesz. Ha tehát ennek adunk meg CSS beállításokat, akkor az oldalon az összes többi (pár kivétellel) szöveg ezeket a beállításokat fogja használni.

```
36 body {
37   font-family: Arial, helvetica, sans-serif;
38   color: #cccccc;
39   background-color: #000000;
40 }
```

A példát használó oldalon, ha új bekezdéseket helyezünk el, azok betűtípusa alapértelmezésben világosszürke, Arial típusú lesz, míg az egész oldal uralkodó háttere fekete marad. Megfigyelhetjük, ha a bekezdésekbe hivatkozások kerülnek, azok színe nem változik meg, ez képezi például az egyik kivételt, ilyenkor konkrétan a hivatkozásnak kell beállítani a kívánt színt.



29. ábra: A „body” beállítás eredménye

7.2.9 Szabályos CSS kód

Ahogy arról már az előzőekben is szó esett, a jelenkori technológiai elvárásoknak megfelelően elengedhetetlen, hogy az ajánlásoknak megfelelő forráskódot helyezzünk el a weboldalakon. Azon felül, hogy kiküszöböljük vele a különböző böngészők eltérő sajátosságait, nagyban megkönnyíthetjük saját és mások munkáját. A böngészők sajnos még a mai nap sem ugyanazt a technikai repertoárt tartalmazzák, ami persze innovációs szempontból pozitívum lehet, de a fejlesztőktől fegyelmezett hozzáállást vár el.

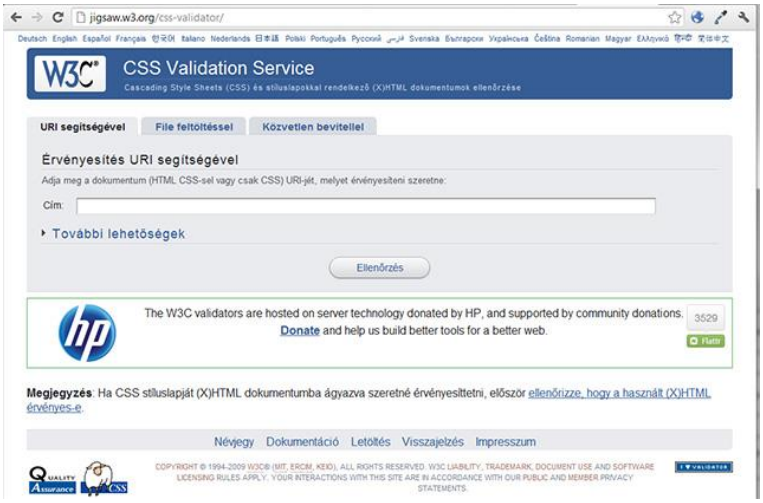
A legfigyelmesebb fejlesztő figyelmét is elkerülhetik apróbb hibák a hatalmas kódhalmazban. A W3C erre ad egy online hozzáférhető szolgáltatást. Hasonlóképpen a HTML-hez, itt is rendelkezésre áll egy validációs oldal, ami a CSS Validaton Service.

Ez a http://jigsaw.w3.org/css-validator/ url-en érhető el. Itt három opció közül választhatunk:

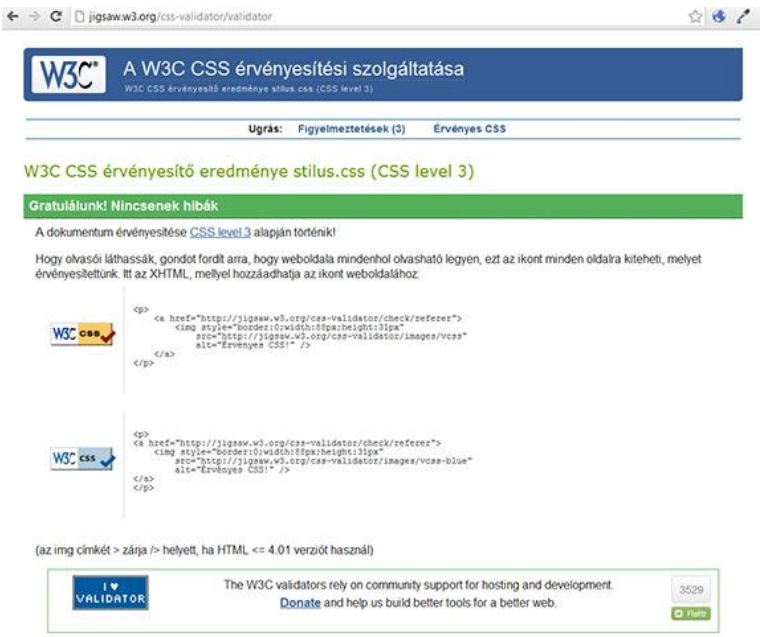
1. Lehetőség van a css dokumentum internetes címét megadni, a validátor ez alapján megvizsgálja a hivatkozott dokumentum kódját.
2. Arra az esetre, ha az oldal nem elérhető az interneten keresztül, lehetőség van a css dokumentum feltöltésére is.
3. Ha csak egy adott kódrészletet akarunk vizsgálni, akkor érdemes a harmadik opciót választani, ilyenkor a bemásolt kódrészletet analizálja a validátor.

Ha a dokumentum szintaxisa és formátuma megfelelő, akkor visszajelzést kapunk annak érvényességéről. Más esetben, ha a dokumentum hibát tartal-

maz, a validátor részletes hibajelentést ad tételesen felsorolva azokat, valamint magyarázatot és megoldási javaslatot.



30. ábra: Online CSS validátor



31. ábra: Szabályos CSS kód a validátorban

7.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

7.3.1 Összefoglalás

A fejezetben bevezetésre kerültek a CSS használatával kapcsolatos legalapvetőbb tudnivalók. Szó esett a CSS szintaxisáról, a kijelölők szerepéről és fajtáiról, megismerkedhettünk a stílusok megadásának lehetőségeivel, a CSS-ben való színkezeléssel és a szabványos CSS kód ismérveivel, továbbá a W3C által fenntartott validátor használatával.

7.3.2 Önellenőrző kérdések

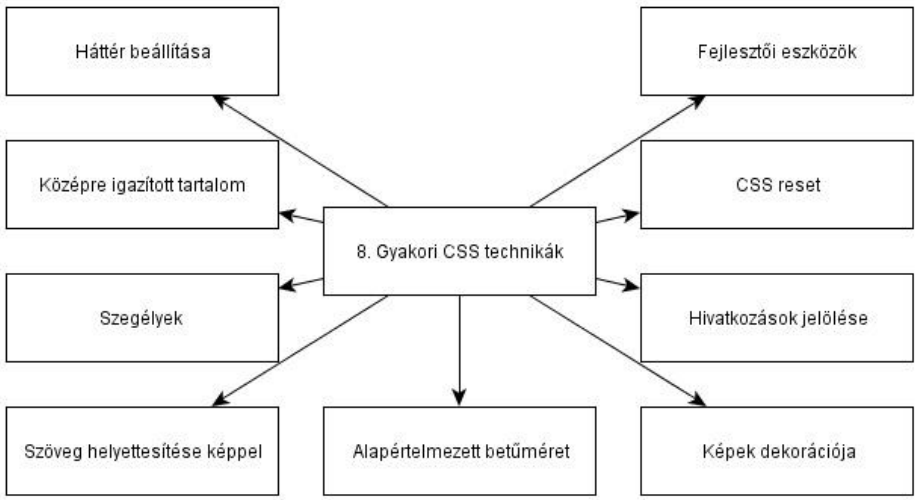
1. Mik az előnyei a CSS használatának?
2. Hogyan adhatunk meg színeket azon CSS tulajdonságoknak, melyek értékéül azok megadhatók?
3. Hogyan fűzhetünk CSS kódot az XHTML dokumentumunkhoz?
4. Milyen elveket kell figyelembe venni az azonosítók és osztályok neveinek megválasztásakor?
5. Hogyan ellenőrizhető legegyszerűbben a CSS kódunk helyessége?

8. GYAKORI CSS TECHNIKÁK

8.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Cél az olvasó megismertetése a leggyakrabban használt CSS technikákkal. A tanuló ismerje meg és tudja alkalmazni a fejezetben bemutatott módszereket. Értse meg az egyes módszerek lényegét és azt mindig az adott, aktuálisan készítendő weblapra specifikáltan legyen képes megvalósítani.

8.2 TANANYAG



32. ábra: Fogalomtérkép

8.2.1 Jól bevált megoldások

A weboldalak fejlesztésekor azok sajátosságaiból fakadóan, olyan problémákkal találkozhatunk, melyek megoldására jól bevált módszereket találhatunk. Ezek egy része olyan megoldások, melyek biztosítják a különböző böngészők közti átjárhatóságot. Gyakran sokkal egyszerűbben is megoldható egy adott feladat, viszont a megfelelő mértékű kompatibilitás hiánya miatt, egy összetettebb megoldást kell alkalmaznunk. Az itt leírtak a CSS2 ajánlás szerint vannak megvalósítva. Az újabb CSS3 már komplett megoldásokkal rendelkezik sok speciális esetre. Ide tartozik az elemek alakítása, forgatása, transzformáció-

ja és még az animálása is. A http://www.css3.info/ oldalon részletesebb információkat találunk angol nyelven.

Tulajdonság	Leírás	Megadható értékek
background-image	Háttérkép	pontos URL vagy none (semmi).
background-repeat	A háttérkép ismétlődésének szabályozása.	repeat (alapértelmezés; mindkét irányba ismétlődik) , repeat-x (csak vízszintesen), repeat-y (csak függőlegesen), no-repeat (nincs ismétlődés).
background	A háttér teljes meghatározása.	background-color || background-image || background-repeat || background-attachment || background-position
width	Az adott objektum szélességét szabályozza.	default (alapértelmezett; megegyezik a szülő-objektummal), width (hosszúság-egységek szerinti szélesség. Részletek: htmllang7.html), százalék_érték (egész szám, melyet egy %-jel követ)
height	Megadja az egyes elem magasságát.	auto (alapértelmezett), hosszúság-egység (Minták: http://tferi.hu/html-konyv/css-mertektablazat), százalék_érték (egész szám, melyet %-jel követ).
border	Az elem minden határának leírása	width (szélesség), style (stílus) és color (szín) sorrendben
border-width	Megadja az elem határainak szélességét - a 4 érték sorrendje: felső, jobb, alsó, bal.	thin, medium, thick valamint méret (számmal).
border-style	Megadja az elem határainak stílusát - mind a 4 értéket. Paraméterek: mint fent.	none - nincs vonal, dotted - pontozott vonal, dashed - szaggatott vonal, solid - folytonos vonal, double - kettős vonal

Tulajdonság	Leírás	Megadható értékek
border-color	Megadja az elem határainak színét a hagyományos paraméterekkel - mind a 4 értéket.	Bármely szín
overflow	Meghatározza, hogy mi történjen az elem tartalmával, ha eléri a szélességét, illetve magasságát.	visible (látható; alapértelmezett; a tartalom teljesen kiírásra kerül és az elemek mellé nem kerül oda a gördítőcsík; a méret átírható), scroll (gördíthető a tartalom teljesen kiírásra kerül és az elemek mellé odakerül a gördítőcsík is), hidden (rejtett; a tartóba be nem férő objektumok nem kerülnek kiírásra), auto (a gördítőcsík csak szükség esetén jelenik meg).
text-indent	Bekezdés első sorának behúzása hosszmértékkel vagy százalékkal.	Bármely méret

8.2.2 Háttér beállítása

Az elsők között merül fel az igény a weblapok hátterének megváltoztatására. Fontos tudni, hogy nem csak a weblap hátterét lehet módosítani, hanem bármely megjelenő elemét is. Az alapértelmezett háttér általában fehér, a többi oldalon megjelenő elemé pedig átlátszó. A háttérszín beállításához a *background-color* tulajdonságot használjuk, és ennek értékéül egy színt adunk meg.

```
41 body { background-color: #ff0055; }
```

Tartalom és megjelenés

A háttérképek beállítása a webdesign egyik sarkalatos pontja. A grafikai elemeket használó weboldalak jóformán csak ennek a használatával ruházzák fel az oldalt a megfelelő megjelenéssel. Ha visszatérünk ahhoz az elváráshoz, hogy a tartalomnak el kell válnia a megjelenéstől, nyilvánvalóvá válik, hogy dizájn célra nem használhatjuk az *<img />* elemet. Az XHTML a tartalomért felelős tehát egy cikkben szereplő híres személy fotója lehet a tartalom része, de az árnyékolt hatású fejléc semmiképpen nem ebbe a kategóriába tartozik. A meg-

oldás tehát az, hogy az összes dizájnért felelős grafikai állományt valamely HTML elem háttereként állítunk be.

Háttérképek

A háttérképek beállításához szükséges a grafikai állomány elérési útjának megadása. Magától értetődő, hogy a design hordozhatósága miatt ezt az elérési utat relatívan adjuk meg, és ezeket az állományokat a CSS állományokkal együtt mozgassuk. Felhasználhatunk erre a célra bármely böngészők által támogatott képformátumot, de a leggyakrabban a PNG és a JPG fordul elő. A beillesztéshez használhatjuk a *background-image* tulajdonságot. Értékül az url kulcsszó használatával megadjuk az elérési utat, persze ha a kép állomány a css-el egy könyvtárszinten van, elegendő csak a fájlnevet megadnunk.

```
42 #fejlec { background-image:
   url(fejlechatter.png) }
```

Ha háttérképet alkalmazunk általában az az alapértelmezett viselkedés, hogy a böngésző mozaikszerűen kitölti vele az elem hátterét. Ez persze megfelel, ha egy háttérmintát akarunk kialakítani, viszont, ha ettől eltérő viselkedést akarunk, be kell állítanunk a kép ismétlődésére vonatkozó szabályt a *background-repeat* tulajdonsággal. Eszerint a következő értékeket állíthatjuk be:

1. *repeat* - mozaikszerű ismétlés
2. *no-repeat* - nincs ismétlés
3. *repeat-x* - vízszintes ismétlés
4. *repeat-y* - függőleges ismétlés

Abban az esetben, ha a no-repeat értéket választjuk a kép csak egyszer kerül beillesztésre. Ha a HTML elem mérete ilyenkor megegyezik a háttérképével, nem kell további beállításokat eszközölni. Azonban ha a befoglaló elemnek egy bizonyos pontjára akarjuk pozícionálnia háttérképet, meg kell adnunk annak helyzetét a *background-position* tulajdonsággal, és a *top*, *right*, *bottom*, *left* értékekkel.

```
43 #fejlec {
44  background-color: #cccccc;
45  background-image: url(fejlechatter.png);
46  background-repeat: no-repeat;
47 }
```

A háttér beállítása viszonylag gyakori feladat ezért rendelkezésre áll egy rövidített forma a háttér tulajdonságainak megadására:

```
48 #fejlec { background: #ccc url(fejlechatter.png)
   no-repeat }
```

Képek optimalizálása

Általános elvárás, hogy a weboldalak bárhol bármilyen körülmények közt és bármely eszközön gyorsan és megbízhatóan jelenjenek meg. A nagyobb sávszélesség ellenére is fontos szempont a felhasznált állományok méretének minimalizálása, hogy ezzel biztosítsuk a gyors letöltést. Másrészt gondoljunk csak arra, hogy egy nagyobb látogatottságú oldalnál ki is kell tudni szolgálnia nagy mennyiségű adatot.

Weboldalunk grafikáinak tudatos tervezésével megoldhatjuk, hogy azok minél kisebb méretűek legyenek. Általános megoldás, hogy amit ismételni tudunk, azt ismételjük is. Ilyenkor mondjuk egy vonal adott szeletét kivágjuk és a background-repeat tulajdonság segítségével ismételjük azt az adott tengelyen. Ilyen módon készíthető árnyék vagy a weboldalunk keret is. A lábléc alsó árnyékos szegélyének kialakításához például használhatjuk ezt a megoldást:

```
49 #lablec { background: transparent
   url(alsoarnyek.png) bottom repeat-x; }
```

8.2.3 Középre igazított tartalom

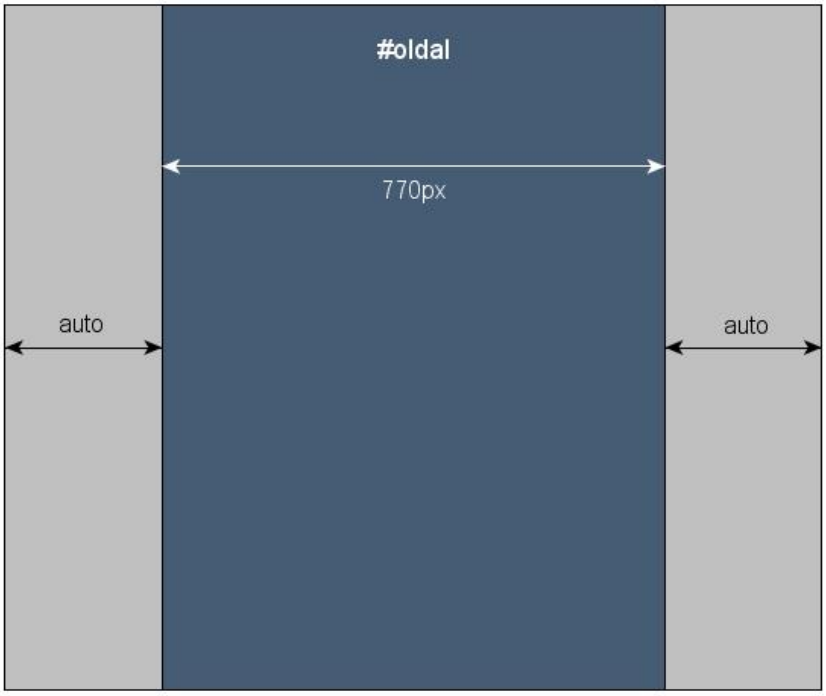
A legalapvetőbb feladatok egyike, amikor egy weboldalt vagy annak egy részét középre akarjuk igazítania befoglaló elemen belül. Erre az esetre az oldalt, vagy a választott tartalmat célszerű egy div elemen belülre elhelyezni. Mivel ezek mérete automatikusan igazodik azok tartalmához a pontos hatás érdekében meg kell adnunk annak a fix szélességét. A kívánt viselkedés az, ha az oldal átméretezésekor is folyamatosan a böngésző közepén marad az oldalunk. A megoldáshoz az elem margóit kell állítanunk a jobb és bal oldalán. Mivel nem ismerjük a rendelkezésre álló területet, nem adhatunk meg a margóknak fix értéket A megoldás, ha a két oldalsó margónak „auto” értéket adunk. Ebben az esetben a böngésző egyenlő mértékben osztja el a rendelkezésre álló szabad területet. Természetesen, hogy tudjunk megfelelően hivatkozni az egész oldalt befoglaló elemre, ellátjuk azt az „oldal” azonosítóval.

```
50 <body>
51   <div id=„oldal”>
52     Az oldal tartalma
53   </div>
54 </body>
```

A CSS kód a következőképpen alakul:

```
55 #oldal {
```

```
56   width: 770px;
57   margin-left: auto;
58   margin-right: auto;
59 }
```



33. ábra: Az oldalbeállítás eredménye

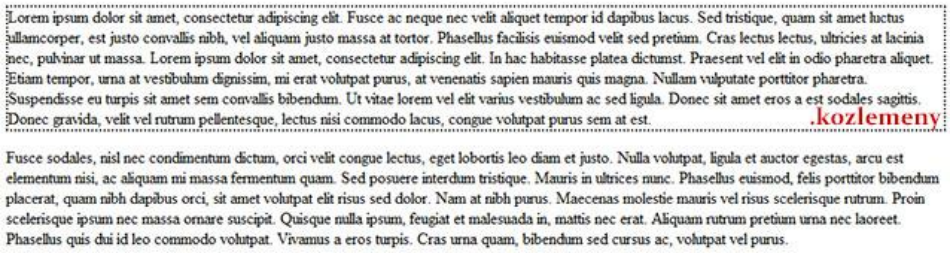
8.2.4 Szegélyek

A CSS lehetőséget biztosít az oldalon megjelenő elemek szegélyekkel történő ellátására. A következő fejezetben erről részletesebben is szó esik, de most induljunk ki abból, hogy egy elemnek négy oldala van. Minden oldalnak külön állítható a szegélye. Egy szegélynek ezen felül három beállítása van: vastagság, stílus és szín, amelyek sorban *border-width*, *border-style* és *border-color* tulajdonságokkal állíthatóak. A szegély vastagságát a legtöbb esetben pixelekben adjuk meg, a színét pedig az előző fejezetben ismertetett módszerek valamelyikével, leggyakrabban hexadecimális értékkel. A stílusokból több is a rendelkezésünkre áll, azonban sajnálatos módon kevés az, amely minden böngészőben egyaránt ugyanazt az eredményt adja. A leggyakoribbak ezek közül:

- *solid* - egybefüggő vonal
- *dotted* - pontozott vonal
- *dashed* - szaggatott vonal

Az alábbi példában a „kozlemeny” osztállyal ellátott elem sötétszürke, pontozott, 2 pixel vastagságú szegélyt kap:

```
60 .kozlemeny {
61  border-width: 2px;
62  border-style: dotted;
63  border-color: #333333;
64 }
```



34. ábra: A szegélybeállítás eredménye

Abban az esetben, ha csak az egyik oldalnak szeretnénk szegélyformázást megadni, ismernünk kell a paraméterek rövidített megadási módját. Ebben az esetben egy sorban írjuk a szegélyvastagságot, a szegélystílust és végül a szegély színét ebben a sorrendben. Az értékeket szóközökkel választjuk el egymástól. Ezzel a módszerrel megadhatjuk a négy oldal beállításait is egyszerre, ha azt a *border* tulajdonság után soroljuk fel. A különböző oldalakat az alábbi módokon tudjuk megcímezni: *border-top*, *border-right*, *border-bottom*, *border-left*. Hasonlóképpen az előző példához a „kozlemeny” osztállyal ellátott elemnek most csak az alsó szegéylét állítjuk:

```
65 .kozlemeny {
66  border-bottom: 2px dotted #333333;
67 }
```

8.2.5 Szöveg helyettesítése képpel

Szinte minden szervezet vagy vállalat oldalán megfigyelhető, hogy az oldal szöveges neve helyett annak logója látható. Általánosan a weboldalak többsége rendelkezik logóval, ami jellegéből fakadóan egy grafikus állomány. Azonban

elengedhetetlen, hogy ilyen esetekben a logó feltüntetése mellett az oldalon megmaradjon annak neve szöveggel kiírva. Abban az esetben, ha ezt egy az egyben képre cseréljük, az oldal tartalmát csonkítjuk vele. Másik fontos megközelítés, hogy a keresőrobotok, amikor feltérképezik az oldalunkat, megtalálják az oldal nevét, és később, ha az oldalunkat keresik az érdeklődők könnyebben megtalálják azt a neve alapján.

Ezen elvárásoknak megfelelően létezik olyan megoldás, amely megtartja a szöveget, de azt grafikus böngészőben nem mutatja meg a látogatónak, viszont ha az oldal szerkezetét vizsgáljuk, az még mindig jelen van. A megoldásnál a szöveget külön span elembe helyezzük el, amit eltüntetünk, de az azt befoglaló elemnek megadjuk a háttérképet, amely a szöveg helyett fog funkcionálni.

```
68 <h1 id=„logo”><span>Weboldal neve</span></h1>
```

Az ehhez tartozó CSS kódban meg kell adni a logónak megfelelő méretű magasságot és szélességet (*width*, *height*). Valamint azt, hogyan viselkedjenek az azon túlnyúló elemek, ugyanis így rejtjük el a benne foglalt szöveget. Ehhez az *overflow* tulajdonság értékéül *hidden*-t állítunk be, így a túlnyúló szöveg azon kívül nem fog látszani. A szöveg elrejtésére annak a behúzásán kell módosítanunk olyan mértékben, hogy az eltűnjön az oldalról. Ez persze csak a látszat, ugyanis a szöveg még mindig az oldalon marad.

```
69 #logo {
70   width: 100px;
71   height: 100px;
72   overflow: hidden;
73   background-image: url(logo.png);
74 }
75 #logo span {
76   text-indent: -9999px;
77 }
```

8.2.6 Alapértelmezett betűméret

A méreteknél használhatunk egy rendkívül hasznos mértékegységet, ami az *em*. Ez egy relatív mértékegység, azaz a szöveg méretezése esetén ez azt jelenti, hogy a normál módon örökölt szövegmérethez viszonyítva mekkorára akarjuk azt beállítani. Az 1em jelenti a 100%-ot, tehát egy 16 pixeles szövegmérethez beállított 2em 32 pixeles méretet fog eredményezni. Nagyon hasznos tud lenni, ha konzekvensen így adjuk mega méreteket, így azok későbbi változtatásával az oldalon használt elemek megfelelőképpen fognak idomulni.

Problémát okoz azonban, hogy kikalkuláljuk a megfelelő méretet, így a legegyszerűbb emberi gondolkodásnak megfelelő megoldás, ha kezdetben a body

elem alapértelmezett szövegméretét 62.5%-ra állítjuk be. Innentől az 1em egyet jelent a 10px-es beállítással, az 1.2em a 12px-el és így tovább. Az alapértelmezett betűméret ugyanis 16 pixeles, így ki lehet számolni a továbbiakat. Ez a megoldás sajnálatos módon nem működik régebbi böngészőkben.

A példában a bekezdések betűméretét 12 pixelesre állítjuk:

```
78 body { font-size: 62.5%; }
79 p { font-size: 1.2em; }
```

8.2.7 Képek dekorációja

Egyszerű keret

Ha koncepció nélkül helyezünk el képeket az oldalon az a leggyakrabban annak rovására megy. Adott méretben és adott szisztémában elhelyezve a legtöbb grafikai tervben jó benyomást kelt, viszont előfordulhat, hogy más megoldáshoz kell folyamodni.

Főleg ha a képek megegyező hátterűek a weboldal hátterével azok könnyen egybemosódhatnak, így felborítva a megcélzott elrendezést. Erre az esetre elegendő egy egyszerű keretet helyezni a kép köré, amelynek megvalósítására a *border* CSS tulajdonság használata bőven elegendő. A keret vastagsága és színe innentől csak egyszerű beállítás kérdése.

```
80 .keret { border: 1px solid #333333 }
```

35. ábra: Keret a kép körül

Árnyékolás

Az egyszerű keretnél valamivel kifinomultabb megoldás az árnyék hozzáadása. Ez tulajdonképpen egy trükk ami egy előre elkészített grafika képhez

adásával valósul meg. Ilyenkor készítenünk kell egy befoglaló elemet amelybe az árnyékolni kívánt kép kerül. Ennek hátteréül kell megadni az előre elkészített árnyékot.

```
81 <div class=„arnyekkeret”><img src=„kep.jpg”
   /></div>
```

Fontos, hogy ez a befoglaló elem annyival nagyobb legyen a belé helyezett képtől, hogy látszódjon pár pixelnyi árnyék. Alapértelmezésben a beillesztett képelem, ennek a bal felső sarkába kerül, így a jobb alsó sarokban kilátszik a háttérként beillesztett jobb alsó árnyék. A példában a beillesztett kép 100x100 pixel méretű, a háttérül szolgáló árnyék pedig 110x110 pixel.

```
82 div.arnyekkeret {
83   width: 110px;
84 height: 110px;
85 background: transparent url(arnyek.png);
86 }
```

Ez talán a legegyszerűbb árnyék megoldás, de léteznek ettől sokkal összetettebb és elegánsabb megoldások is. A CSS3 erre már beépített megoldással rendelkezik.

8.2.8 Hivatkozások jelölése

Az alapértelmezett hivatkozás kék színű és aláhúzott minden böngészőben. Ezt a viselkedést az általánosan beállított szövegszínnel nem tudjuk felülbírálni. Ebben az esetben külön a hivatkozásra vonatkozó beállítást kell megadnunk.

```
87 a { color: #555555; }
```

A hivatkozás alatt megjelenő aláhúzás nem szegély hanem a *text-decoration* beállítás eredménye ami alapértelmezésben *underline* (azaz aláhúzott). Az aláhúzás eltüntetéséhez ezt kell módosítanunk, így az oldalon lévő hivatkozások enélkül fognak megjelenni.

```
88 a { text-decoration: none; }
```

Élő hivatkozás

A hivatkozás több állapotban fordul elő az oldalon annak függvényében, hogy az már látogatott, vagy az egér milyen interakcióban van vele. Gyakran használt megoldás, hogy a hivatkozás az egérmutató fölé vitelekor valami megváltozik, ezzel jelezve, hogy nem egyszerű szövegről van szó. Az egérmutatóval való érintkezéshez a „:hover” kijelölőt kell a hivatkozáshoz adni, így módosítható annak viselkedése. Példánkban a hivatkozás világosszürke aláhúzott, de az egér fölé vitelekor aláhúzott sötétszürke lesz.

```
89 a {
90 color: #cccccc;
91 }
92 a:hover {
93   color: #333333;
94   text-decoration: none;
95 }
```

Külső hivatkozás

Abban az esetben, ha olyan hivatkozást helyezünk az oldalra, amely másik oldalra mutat ezt célszerű jelölni a felhasználók számára. Elképzelhető, hogy ezt elkülönülő színnel jelöljük, de sokkal elegánsabb, ha olyan háttérképet helyezünk el, amely jelzi, hogy nem oldalon belüli hivatkozásról van szó.

8.2.9 CSS reset

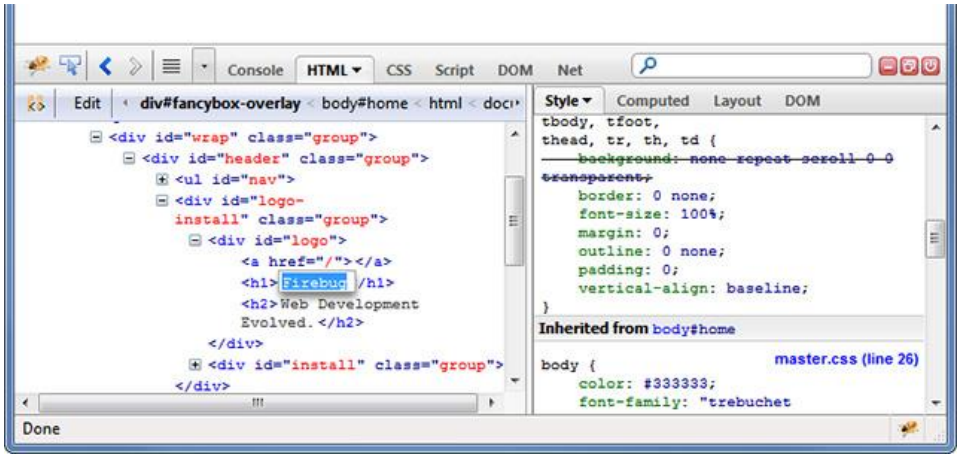
A böngészők eltérő előformázási beállításokkal rendelkeznek. Ha szeretnénk olyan webdesignt készíteni, mely minden böngészőn hasonlóan jelenik meg, egy úgynevezett „css reset”-et kell használnunk. Ezt természetesen a stíluslapunk elejére kell rakni. Segítségével minden böngésző alapbeállítást felülírjuk, így bármelyiket is használjuk, ugyanazt fogjuk látni, mint a többin. Természetesen ez egy nagyon letisztult vagy inkább puritán oldalt eredményez, így további munka szükséges a megfelelő eredményhez. Érdemes ezzel kezdeni a fejlesztést, mert később már nehezebb dolgunk lesz.
A http://meyerweb.com/eric/tools/css/reset/ oldalon egy minden igényt kimerítő megoldást találhatunk.

8.2.10 Fejlesztői eszközök

A fejlesztéshez használt eszközök megválogatása kulcsfontosságú momentum. Kifejezetten a CSS-el való munkánál egyszerre három helyen kellene dolgoznunk. Az egyik a HTML dokumentum a másik a CSS állomány a harmadik pedig a böngésző ahol megtekinthetjük a végeredményt. Itt nagyon fontos megjegyezni, hogy egyik fejlesztőeszköz előnézeti képében sem bízhatunk meg teljes mértékben, a végeredményt mindenképpen böngészőkben kell kipróbálni!

Annak elkerülése végett, hogy a fejlesztés során folyamatosan egyik programról a másikra kelljen váltanunk megfelelő eszközt kell alkalmaznunk. A CSS fejlesztés sok eleme próbálgatásos alapon működik, ritka az amikor a fejlesztő mindig pixelre pontosan adja meg elsőre az elemek beállításait. Kiváló eszközök ilyenkor a böngészőbe telepített vagy telepíthető elemzők.

Ilyen például a széles körben elterjedt **Firebug** (http://getfirebug.com) amely fejlesztési szolgáltatásainak kifejtése meghaladja ezen kiadvány tartalmi korlátait. Hasonló megoldás a Webkit böngészőmotorral rendelkező böngészőkben lévő **Web inspector**.



36. ábra: Firebug

Ezek segítségével részletekbe menően vizsgálhatjuk meg a weboldalunk bármely elemét. HTML elem vizsgálatakor részletes eredményeket kapunk arról, hogy milyen CSS beállítások az aktuálisan érvényesek rá vonatkozóan. Információt kapunk arról, hogy milyen CSS állományok befolyásolják az adott elem beállításait. Láthatjuk, hogy mely szabályok bírálják felül a másik beállításait. A legnagyobb előnyük viszont az, hogy lehetőségünk van ezen beállításokat közvetlenül a böngészőben állítgatni, így elkerülve a folytonos váltogatást az eszközök között. Ilyenkor a változás rögtön megmutatkozik a böngészőben, így elég a megfelelő beállítás megtalálása után felvinni azt a CSS állományba.

Egy másik nagyon egyszerű megoldás, amely még külön telepítést sem igényel az XRAY (http://westciv.com/xray/). Ezt könyvjelzőként kell elmentenünk. Valójában egy Javascript kód, amelyet a könyvjelzőre való kattintással futtatunk az oldalunkon. Hatására kiválaszthatunk egy tetszőleges elemet és az XRAY megmutatja annak aktuális CSS beállításait.

8.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

8.3.1 Összefoglalás

A fejezetben szó esett és példákon keresztül bemutatásra kerültek a leggyakrabban használt CSS technikák. Szó esett a hátterek, a szegélyek, a képek és hivatkozások CSS-sel történő megadásáról. Továbbá olvashattunk a középre igazított tartalmak és a szövegek képekkel való helyettesítéséről, továbbá a CSS reset jelentőségéről.

8.3.2 Önellenőrző kérdések

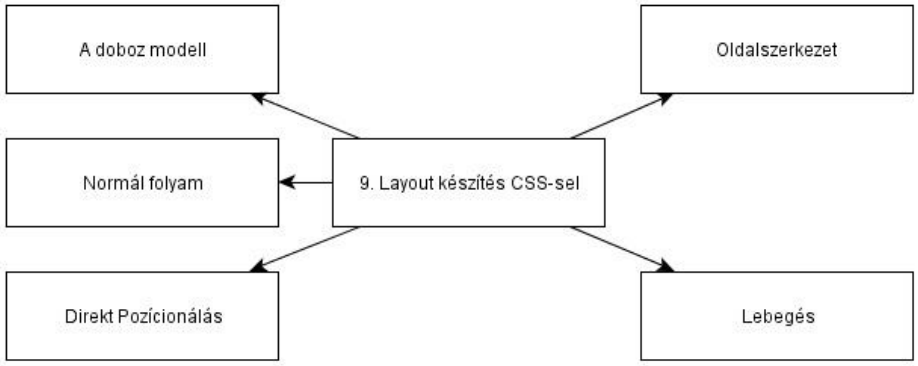
1. Milyen beállítások adhatók meg egy elem háttérképénél?
2. Miért érdemes az alapértelmezett betűméretet 62.5%-ra beállítani?
3. Milyen beállítások adhatók meg az elem szegélyeinél?
4. Hogyan lehet egyszerűen árnyékot helyezni egy HTML képhez?
5. Mire jó az úgynevezett CSS reset?

9. LAYOUT KÉSZÍTÉS CSS-SEL

9.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Cél, hogy az olvasó megértse a CSS-sel való layotkészítés lényegét és tanulja meg, hogy a képernyő tartalmát miként érdemes optimálisan, jól elkülöníthet részekre osztani. Értse meg a tanuló a doboz-modell elvét és módszerét, ismerje az egyes dobozok beállításához tartozó attribútumok lehetőségeit. Ismerje a különféle pozícionálási lehetőséget.

9.2 TANANYAG



37. ábra: Fogalomtérkép

9.2.1 A tulajdonságokhoz kapcsolódó értékek

A megfelelő oldalszerkezet kialakításához ismernünk kell az elemek működését és a CSS pozícionálási lehetőségeinek alapvető természetét.

Tulajdonság	Leírás	Megadható értékek
padding	Megadja a szükséges helyet az objektum oldalai és az egyéb tartalom között, a 4 érték sorrendje: felső, jobb,	hosszúság a mértékegységek szerint, percentage (százalék; egész érték, melyet egy %-jel követ)
margin	Megadja az elemek margóinak 4 értékét felső, jobb, alsó, bal	hosszúság a szokásos mértékegységek szerint: percentage (százalék; egész

Tulajdonság	Leírás	Megadható értékek
	sorrendben.	érték, melyet egy %-jel követ), auto (alapértelmezett; a felső és az alsó margók egyenlőek)
position	Mi legyen az elem pozíciója a szülőhöz képest?	static (alapértelmezés; az elemnek nincs speciális pozicionálása), absolute (abszolút pozicionálás egységek), relative (relatív pozicionálás egységek, fixed (Internet Explorer 7 és utána; az objektum a tartalmazóhoz képest hol helyezkedik el)
top	Megadja vagy beállítja az objektum felső szélét a következő pozicionálási művelethez.	auto (alapértelmezett; a HTML-összeállításától függő helyi érték), hosszúság-egység (Minták: http://tferi.hu/html-konyv/css-mertektablazat), százalék_érték (egész szám, melyet %-jel követ)
left	Megadja az egyes elem bal oldali elhelyezkedését a dokumentum-hierarchiában.	auto (alapértelmezett; a HTML-összeállításától függő helyi érték), hosszúság-egység (Minták: http://tferi.hu/html-konyv/css-mertektablazat), százalék_érték (egész szám, melyet %-jel követ)
right	Megadja az egyes elem jobb oldali elhelyezkedését a dokumentum-hierarchiában.	auto (alapértelmezett; a HTML-összeállításától függő helyi érték), hosszúság-egység (Minták: http://tferi.hu/html-konyv/css-mertektablazat), százalék_érték (egész szám, melyet %-jel követ)
bottom	Megadja az objektum alsó szélét a következő pozicionálási műveletnek.	auto (alapértelmezett; a HTML-összeállításától függő helyi érték), hosszúság-egység (Minták: http://tferi.hu/html-konyv/css-mertektablazat), százalék_érték (egész szám, melyet %-jel követ)
float	Megadja, hogy az adott doboz melyik oldalon lebegjen.	none (sehol; alapértelmezett), left (bal), right (jobb).
overflow	Meghatározza, hogy mi történjen az elem tartalmával, ha eléri a szélességét, illetve magasságát.	visible (látható; alapértelmezett; a tartalom teljesen kiírásra kerül és az elemek mellé nem kerül oda a gördítőcsík; a méret átírható), scroll (gördíthető a tartalom teljesen kiírásra kerül és az

Tulajdonság	Leírás	Megadható értékek
		elemek mellé odakerül a gördítőcsík is), hidden (rejtett; a tartóba be nem férő objektumok nem kerülnek kiírásra), auto (a gördítőcsík csak szükség esetén jelenik meg).
z-index	Megadja az egyes elemek sorrendjét.	auto (ahogy a kódban következik; alapértelmezett) ,vagy egész_szám. Azonos értékű elemeknél a leírási sorrend számít.
display	Meghatározza a megjelenített elem rendezési tulajdonságait.	none (nincs rendezve), block (blokk-elemként), inline (egysorban; alapértelmezett), inline-block (egysorban, de blokk-elemként van rendezve), list-item (lista-elem előtag megjelenítése és blokk-elemként;

9.2.2 A doboz modell

Ahhoz, hogy a CSS-t gyakorlottan tudjuk használni, tisztában kell lennünk a HTML elemek alapvető felépítésével. Ez a felépítés a megjelenésre vonatkozik a CSS tekintetében. Minden weboldalon elhelyezett objektumok megjelenését írja le a **doboz-modell**. Ebben a megközelítésben minden elem egy téglalapon belül helyezkedik el, amely különböző rétegekből áll. A doboz közepén maga a tartalom van, tehát az általunk beillesztett elem. A weboldalakon sok esetben ebből a dobozból semmit nem látunk legfeljebb a benne foglalt szöveget, de ettől függetlenül a többi része is jelen van láthatatlan formában.

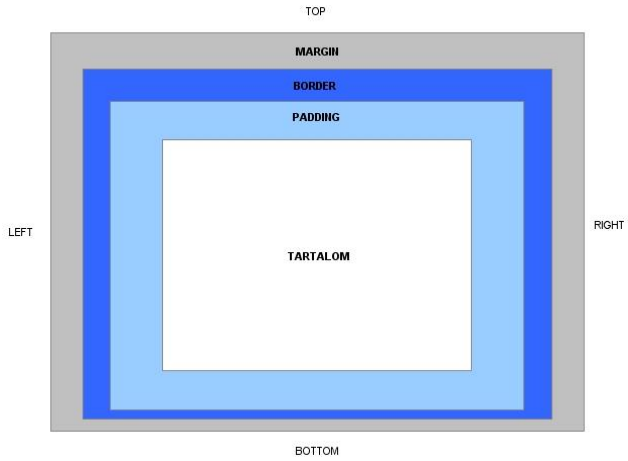
Ezek a dobozok egymásba ágyazhatóak, korlátlan mélységben és mennyiségben, de a befoglaló elem limitálja a benne foglaltak kiterjedését. Abban az esetben, ha egy elemben több másik helyezkedik el, azok hatással vannak a másik méreteire is. A modell vonatkozásában háromféle elemtípust különböztetünk meg egymástól.

1. Blokk-szintű (*block-level*) elemekhez önálló doboz tartozik, amely rendelkezik margóval, kerettel, és a kereten belüli belső margóval. Amennyiben az elem tartalma szöveg, az sordobozokba rendeződik.
2. A sordobozok nem rendelkeznek sem belső sem külső margóval. Ezek tulajdonképpen a bekezdés sorait jelentik: egymás alá kerülnek a dobozon belül, annak teljes szélességét kitöltve a doboz belső margóján belül.

3. A sorbeli elemek (*inline-level* elements) rendelkezhetnek önálló dobozzal, amely kerettel is ellátható, azonban ez a doboz mindig egy sordobozba kerül.

A középen elhelyezkedő tartalmat (*content*) három réteg veszi körül. Alap esetben a legtöbb nulla értékkel rendelkezik, ilyenkor ezek nem jelennek meg a böngészőben. A különböző elemeknek az alapértelmezett nagysága eltérhet böngészőnként, ennek a kiküszöbölésére használhatjuk az előző fejezetben ismertetett *CSS reset* megoldást. Kifelé haladva közvetlenül ezt veszi körül a belső margó (*padding*) amely a tartalom és a keret közötti távolságot szabályozza, ez a távolság a belső tartalom és az elem széle közti üres hely. A következő maga a keret, vagy szegély (*border*), amely még ténylegesen az elem területén található. Végül a doboz külső felületét a külső margó veszi körbe (*margin*). A külső margó az adott elem és a szomszédos elemek közti üres helyet adja.

A doboz méreteinek meghatározása a régebbi Internet Explorer böngészőkben eltért az ajánlástól, de napjainkban már jóformán minden böngésző megfelelően kezeli ezt. A tartalom szélessége (*width*), csak a belső tartalomra vonatkozik, azonban az objektum teljes szélességét a tartalom, a belső margó, a szegély és a külső margó összeadásával határozható meg.



38. ábra: Doboz modell

Tehát az alábbi példa egy 100 pixel széles objektumot fog eredményezni. Jól látható, hogy itt a tartalom szélessége (*width*) csak 70 pixel, ehhez adódik hozzá a belső margó, ami mindkét oldalon ott van, tehát 5x2 azaz 10 pixel. A külső margó 10 pixel tehát az két oldalra meghatározva 20 pixel. Így a végleges szélesség 70+10+20 = 100 pixel lesz.

```
96 #doboz {
97 margin: 10px;
98 padding: 5px;
99 width: 70px;
100 }
```

Ha hátteret adunk az elemnek az a tartalomra és a belső margó területére lesz érvényes. Mivel a margó az adott elem és a szomszédos elem közti helyközt szabályozza, a háttér beállítás erre a területre már nem terjed ki.

```
101 #szinesdoboz {
102     width: 100px;
103     height: 50px;
104     padding: 20px;
105     margin: 20px;
106     background: #aa44aa;
107 }
```

Értékek beállítása

A doboz minden oldalán külön szabályozható a határoló elemek nagysága. Lehetőségünk van azokat vagy egyszerre módosítani (pl.: margin: 10px), vagy az oldalak angol megnevezésével való kiegészítéssel (margin-left: 10px). A beállítható tulajdonságok tehát a külső margó esetében a *margin-top*, *margin-right*, *margin-bottom*, *margin-left*, valamint ha egyszerre mindet akarjuk állítani a *margin*. A belső margó esetében ez *padding-top*, *padding-right*, *padding-bottom*, *padding-left* és a mindegyik oldalra vonatkozó tulajdonság a *padding*.

Lehetőség van azonban az értékeket rövidített formában megadni, ilyenkor nem kell minden oldal tulajdonságát külön beállítani. A rövidített forma megadásához azonban tudnunk kell, hogy az elemek körüljárási iránya megegyezik az óramutató járásával és felülről kezdődik, tehát: felső, jobb, alsó, bal.

```
108 .blokk {
109     padding-top: 10px;
110     padding-right: 20px;
111     padding-bottom: 10px
112     padding-left: 20px;
113 }
```

Ez a példa lerövidíthető az alábbi formában:

```
114 .blokk { padding: 10px 20px 10px 20px; }
```

A teljes igazsághoz azonban az is hozzátartozik, hogy ha az alsó-felső és a bal-jobb oldalak szélessége megegyezik, akkor tovább lehet rövidíteni a szabályt. Ebben az esetben az első érték az felső-alsó majd a jobb-bal.

```
115 .blokk { padding: 10px 20px; }
```

Margók összevonása

Egy igen egyszerű elképzelés a margók összevonása (margin collapsing). Ezen viselkedés az egymással szomszédos elemek külső margóira vonatkozik. Sok zavart okozhat CSS-el kialakított oldalszerkezetek készítésekor, ha a fejlesztő nincs tisztában ezzel a viselkedéssel.

Egyszerűen, ha két olyan elem kerül egymás alá, melyek rendelkeznek egymással határos margóval (*margin*), akkor azok eggyé olvadnak össze. Az új „összevont" margó mérete megegyezik a nagyobb margó méretével. Tehát ha az egyik elem margója 30 pixel a másiké pedig 20 pixel, akkor az összevont mérete 30 pixel lesz. Ez a viselkedés nem vonatkozik az egymás mellett elhelyezkedő elemekre.

Ugyanezt az eredményt kapjuk elemek egymásba ágyazásakor. Ha a befoglaló elem és a beágyazott elem egyaránt rendelkezik külső margóval, akkor a beágyazó elem margója lesz a mérvadó, a beágyazott elemé pedig eltűnik, azaz beleolvad a külsőbe.

Ez a viselkedés abban az esetben érvényes, ha az elem a normál „folyamban" van. Abban az esetben, ha az lebegtetve van, vagy direkt pozíció van beállítva hozzá, ez a jelenség nem lesz megfigyelhető.

9.2.3 Normál folyam

A CSS-ben három pozícionálási megközelítést különböztetünk meg. Az első a normál folyam. Ebben az esetben az elemek a számukra előre meghatározott módon viselkednek. Ilyenkor a blokk-szintű elemek függőlegesen egymás alá helyezkednek el. Ilyen elemek a *p*, *h1* vagy a *div*. Másrészt a soron belüli elemek, mint a *strong*, *span* egy adott sorban maradnak, ezért szokták azokat sordoboz-nak is nevezni.

Lehetőség van az elem számára generált doboz megváltoztatására. Azaz egy hivatkozást felruházhatunk blokk szintű viselkedéssel, valamint egy *div* elemet soron belüli viselkedésűvé tehetünk. A megjelenés megváltoztatására a *block* tulajdonságot használhatjuk.

A soron belüli elemek, vagyis a sordobozok, hasonlóképpen működnek, megadható ezekhez az oldalsó külső és belső margók valamint a szegély, de ebben az esetben az alsó és felső tulajdonságok nem adhatóak meg. Az ilyen elemek mindig olyan magasak, mint az azt befoglaló elem, tehát a sormagasság növelésével növelhetjük a sordoboz magasságát.

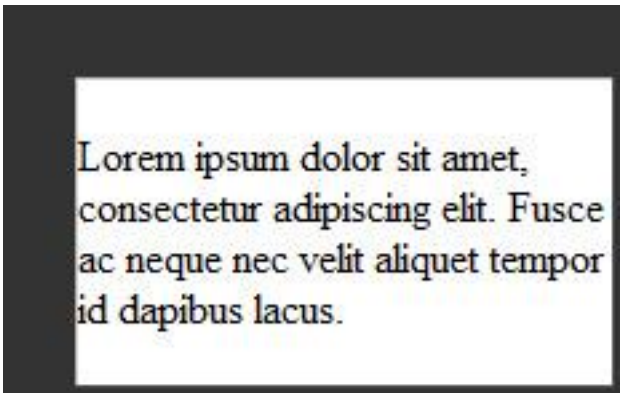
Fontos tehát annak a ténynek az ismerete, hogy minden, ami a képernyőn megjelenik, az valamilyen doboznak a része.

9.2.4 Direkt Pozícionálás

Relatív pozícionálás

A relatív pozícionálás segítségével az elemet egy általunk meghatározott helyre mozgathatjuk. Abban az esetben, ha alkalmazzuk ezt a pozícionálási formát, a kiválasztott elem a helyén marad és látszólag nem is történik változás. A változás az, hogy ezután azt az eredeti helyéről elcsúsztathatjuk az általunk megadott mértékkel. A nevéből adódóan az új pozíciót a régi helyéhez relatívan kell megadni. Megadhatunk felső (*top*), jobb (*right*), alsó (*bottom*) és bal (*left*) pozícionálási értéket. Tehát ha a felső és a bal eltolást 20 pixelre állítjuk egy relatívan pozícionált elemnél az annyival lentebb és jobbra fog tolódni az eredeti állapotához képest. Fontos annak a ténynek az ismerete, hogy az ilyen módon pozícionált elem eredeti helyét ugyanúgy el fogja foglalni.

```
116 #pozicionaltDoboz {
117     position: relative;
118     top: 20px;
119     left: 20px;
120 }
```



39. ábra: Pozícionálás eredménye

Abszolút pozícionálás

A relatívan pozícionált elemtől eltérően az abszolút pozícionálás nem illik bele a normál folyamnál elvárható viselkedésbe. Ebben az esetben a kiválasztott elem kiemelődik az eredeti helyéről, így annak helyére már más elemek

kerülnek, így olyan mintha eltűntettük volna onnan ezt az objektumot. Ennél az esetnél szabadon megválaszthatjuk az elem pozícióját, de ezt mindig a legközelebbi pozícionált szülő elem fogja meghatározni. tehát ha azt akarjuk, hogy az egész böngészőablak helyett egy meghatározott elem legyen a pozícionálási tér, annak relatív pozícionálást kell beállítanunk. Ha nem rendelkezik ilyen szülő elemmel, akkor az a fő dokumentum lesz. Ez nagyobb szabadságot ad, mint az előző módszer.

Előfordulhat, hogy a pozícionálás folyamán bizonyos elemek takarásba kerülnek. Ennek szabályozására használjuk a z-index tulajdonságot, amellyel megadhatjuk egy elem elhelyezésének a mélységét.

Ezzel a módszerrel kialakíthatunk teljes oldalszerkezetet is. Ilyenkor viszont vigyáznunk kell arra, hogy ezek már nem vesznek részt a normál folyamban, így azok határainak módosítása mellett igazítanunk kell ehhez a pozíciójukat is.

Fix pozíció

Tulajdonképpen ez az abszolút pozícionálásnak egy fajtája. A különbséget az adja, hogy ezzel olyan lebegő elemet készíthetünk, ami mindig a kijelző egy adott pontján marad. Ilyenkor hiába görgetünk vagy méretezzük át az ablakot, a kiválasztott objektum ugyanazon a pozíción marad. Gyakran használják ezt a felhasználói élmény javítására, például a visszajelzést biztosító vezérlő elemeket, mindig látható és könnyen elérhető helyen tartják.

9.2.5 Lebegés

A harmadik pozícionálási módszer a lebegés (*float*). Eredetileg a nyomdai megoldásokból származik, amikor egy képelemet helyeztek el a szövegen belül, így az nem a fölé vagy alá került, hanem a szöveg így körbe tudta azt fogni. Ezen megoldás segítségével kialakíthatunk weboldal szerkezeteket, képeket helyezhetünk tartalmak mellé, valamint listákat készíthetünk vele.

```
121 #oldalsav { float: left; }
```

A lebegtetett doboz mozgatható a jobb vagy a baloldalra, melynek az azt befoglaló (szülő) elem, vagy egy másik lebegtetett elem szab határt. Ha egy elemnek lebegést állítunk be, azt azzal együtt része marad a normál folyamnak, de annak helyét átveszik a szomszédos elemei. Mivel az a normáltól eltérően viselkedik, eltakarhatja a vele egy szinten lévő elemeket, azaz föléjük lebeghet. Ez eltérő viselkedés tehát az abszolút pozícionálástól, ugyanis ilyenkor kiemeljük az adott elemet a normál folyamból, így az nincs hatással a többi elemre.

Abban az esetben, ha azonos szinten több elemet lebegtetünk, azok egymás mellé fognak kerülni és az őket befoglaló elem határa szabja meg helyzetü-

ket. Abban az esetben, ha a befoglaló elem kisebb méretű annál, hogy elférjenek benne a lebegtetett elemek, azok egymás alá fognak csúszni, de továbbra is tartják a lebegés szabályait.

A lebegés megszüntetése

Ha azt vesszük, hogy ezen szolgáltatás egyik elsődleges feladata, hogy a környező szöveg azt körbevegye, nem ér meglepetésként, hogy ilyen esetben a szomszédos elemek is hasonlóképpen viselkednek. Ezen hatás megszűntetésére a clear tulajdonságot használjuk. Ennek segítségével az adott elemre alkalmazva felmenthetjük azt a környező lebegtetett elemek hatása alól, és úgy tekintenek arra, mintha a normál módon viselkednének. Az oldalszerkezet kialakításakor a szerkezet összeomlásakor is ezt a tulajdonságot használjuk. Elkülöníthetjük ugyan, hogy mely oldali lebegést szüntesse meg, de a leggyakrabban mindkét oldalra (both) alkalmazzuk.

Az összeomlás

Mivel a lebegéssel megtagadjuk az adott elemnek, hogy normál módon vegyen részt a folyamban, az azt befoglaló elem nem lesz tekintettel arra, tehát tovább nem fog neki helyet fenntartani. Ez akkor lehet problémás, ha a szülő elem nem tartalmaz más normál módon viselkedő elemeket, és az teljes mértékben összecsukódik. Ha ennek háttér van megadva, észrevehető annak összeomlása. Ennek orvoslására több megoldás ismeretes.

Ismerjük a *clear* tulajdonságot, amely itt a megoldást fogja adni. Ilyenkor plusz egy elemet kell beillesztenünk az összes lebegtetett elem után és megadni neki a *clear* tulajdonságot. Azzal, hogy megszünteti a további lebegést, a befoglaló elem tekintettel lesz annak létezésére, és így elkerülhető annak összecsukódása.

```
122 <div style=„clear: both;”></div>
```

Ha szigorúan értelmezzük az XHTML szabályait az nem tartalmazhat üres elemet, így az előző megoldás nem mondható a legelegánsabbnak. Egy alternatívaként olyan technikát is alkalmazhatunk, amely a befoglaló elem tartalmának láthatóságát szabályozza. A befoglaló elemnek a következő beállításokat kell adnunk:

```
123 #tarolo {
124     height: 100%;
125     overflow: hidden;
126 }
```

9.2.6 Oldalszerkezet

Oldalszerkezetek kialakítására több CSS technikát ismerünk. Az egy biztos, hogy a HTML táblázatos megoldás mindenképpen kerülendő és rendkívül idejétmúlt. A konkrét szerkezeti megoldásokat layout-nak nevezzük. Ezek módja rendkívül eltérő és sokféle lehet, a fejlesztők ezeket saját igényeik szerint alakították, így csak irányelveket fogunk áttekinteni.

Az itt ismertetett technikák a *float* tulajdonság használatával valósulnak meg. Klasszikus kettő vagy három oszlopos felosztást valósítanak meg, ami általában bal oldalsáv, fő tartalmi rész, esetleg jobb oldalsáv.

9.2.7 Fix Layout

A legegyszerűbb megoldás talán a fix szélességű elemek lebegtetése. Az egyetlen problémát talán a szülő elemek összeomlása okozhatja. Ahhoz, hogy elkészítsük a megfelelő stílust a szerkezethez, előbb le kell írnunk az XHTML struktúrát.

```
127 <div id=„hatarolo”>
128     <div id=„fejlec”>...</div>
129     <div id=„tartalom”>...</div>
130     <div id=„oldalsav”>...</div>
131     <div id=„lablec”>...</div>
132 </div>
```

Az elkészített struktúra egy ***hatarolo*** azonosítójú elemben kap helyet, ezzel szabályozhatjuk az oldal szélességét, igazítását, valamint a hátterét. Az alapvető elképzelés az, hogy a fejléc a többi elem fölött helyezkedik el, az oldalsáv pedig a tartalom bal oldalán. Ne zavarjon meg, hogy a HTML kódban a tartalom van előbb, erről majd a *float* fog gondoskodni. Ennek felhasználhatósági és akadálymentesítési szerepe van.

Egyszerű megoldás lenne, ha az oldalsávot és a tartalmi részt egyaránt baloldalra lebegtetnénk, így azonban nem lenne köztük szabad hely és az elrendezés zsúfolttá válna. Használhatnánk ugyan erre a célra margókat, de ez a böngészők közti pontatlanság miatt könnyen okozhatná oldalunk szerkezetének szétcsúszását. Ehelyett inkább érdemes mindkét elemet az ellenkező irányban lebegtetni, így egy virtuális rés alakul ki a kettő között.

```
133 #tartalom {
134 width: 520px;
135 float: right;
136 }
137 #oldalsav {
```

```
138     width: 180px;
139     float: left;
140 }
```

Annak elkerülése végett, hogy a lábléc befolyjon a tartalom mellé, meg kell adni annak a clear tulajdonságot. Ezzel együtt elkerülhetjük vele a *hatarolo* elem összeomlását is.

```
141 #lablec {
142     clear: both;
143 }
```

Ahhoz, hogy biztosak legyünk abban, hogy minden böngészőben megfelelően működik az oldal felépítés, kerüljük el a belső margó vízszintes beállításait azoknál az elemeknél, ahol már beállítottuk azok szélességét.

Az oldalt körülhatároló elemnek pedig megadjuk az alapvető beállításokat az igazítására, a méretére és a hátterére vonatkozóan.

```
144 #hatarolo {
145     width: 720px;
146     margin: 0 auto;
147     background: #ffffff;
148 }
```

9.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

9.3.1 Összefoglalás

A fejezetben CSS segítségével való, weboldalak layout-jának kialakításáról olvashattunk. Bemutatásra került a doboz-modell, miszerint a képernyő területén dobozokat helyezünk el, és a weblapon megjelenített elemek mindegyike egy-egy dobozban foglal helyet. Az egyes dobozokhoz paraméterek állíthatóak be, úgy, mint: padding, border, margin. Továbbá a dobozok pozicionálásával kapcsolatos elvekről is esett szó.

9.3.2 Önellenőrző kérdések

1. Milyen direkt pozícionálási módokat használhatunk?
2. Mit jelent a normál folyam?
3. Mi a lényeges különbség az abszolút és a relatív pozíció beállítása között?
4. Milyen tulajdonságok alkotják a doboz modellt?
5. Mi okozhatja a problémát több elem lebegtetésénél?

10. A JAVASCRIPT ALAPJAI

10.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A Web nagy változásokon ment keresztül az elmúlt időkben. Ahogy a világháló fejlődött, úgy fejlődtek az eszközök is. A sima egyszerű HTML nyelvekhez, programozási nyelvek fejlődtek. A JavaScript is egy ilyen programozási nyelv. Nem kell megijedni, mert nagyon egyszerű. Aki még soha nem programozott komolyabban, annak ez lesz a belépés a programozás világába. Manapság a JavaScript elengedhetetlen a weboldalakhoz. Kisebb JavaScriptes programokkal, könnyen feldobhatjuk a honlapunkat. Úgy, mint minden, a világháló is folyamatosan megújul, fejlődik.

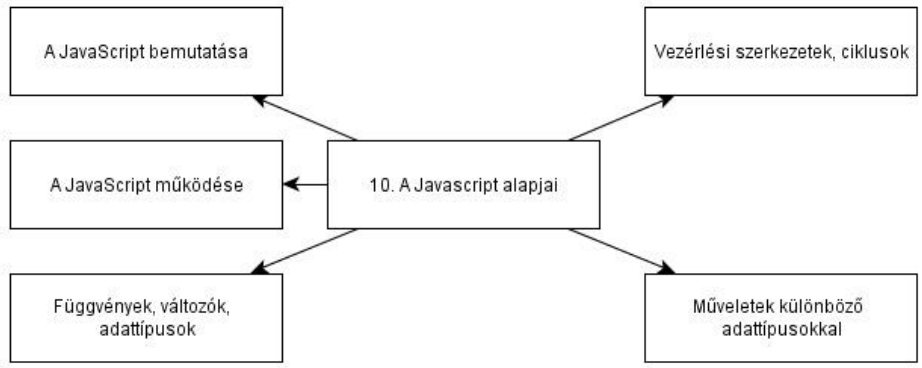
Pár szóban bemutatásra kerül a JavaScript, majd megnézzük, hogyan lehet őket beilleszteni a tartalomba.

Ezek után közelebbről is megvizsgáljuk a függvényeket és szolgáltatásait, az objektumokat, az eseménykezelőket.

Átnézzük a programírás módszereit, melyben szót ejtünk a ciklusokról, változókról. Átnézzük az adattípusokat, az átalakításukat.

Végül átnézzük a vezérlési szerkezeteket.

10.2 TANANYAG



40. ábra: Fogalomtérkép

10.2.1 A JavaScript bemutatása

Webes parancsnyelvek

A webes parancsnyelvek annyiban különböznek a programozási nyelvektől, hogy míg az utóbbit le kell fordítani gépi kódra, úgy a webes parancsnyelvet (mint a JavaScript is), a böngésző sorról sorra olvassa be és hajtja végre. Ennek a JavaScript fájlnak a módosítása mindössze annyi, hogy módosítjuk benne a kódot, elmentjük, majd a böngésző újratöltésével már azonnal életbe lép.

A JavaScript a Netscape böngészőt készítők által született meg. Meg kell említeni, hogy ez volt az első olyan parancsnyelv, amit a weben használtak, és máig is ez a legnépszerűbb. A JavaScript egy kliensoldali programnyelv, azaz a szerveren van letárolva, de a böngészőben, a kliensoldalon fut le. Ezért is nem tud beavatkozni a szerveren lévő dolgokba.

Mit is lehet megvalósítani a JavaScripttel? Íme, egy pár példa:

- Űrlapok tartalmának ellenőrzése, számolások
- Üzenetek, felugró ablakok megjelenítése
- Menüket dobhatjuk fel vele, amikor fölé visszük az egeret
- Kiolvashatunk a böngészőből különböző adatokat

És ez még csak egy pár dolog volt, emellett még sok minden másra is használhatjuk.

Beillesztés a weboldalba

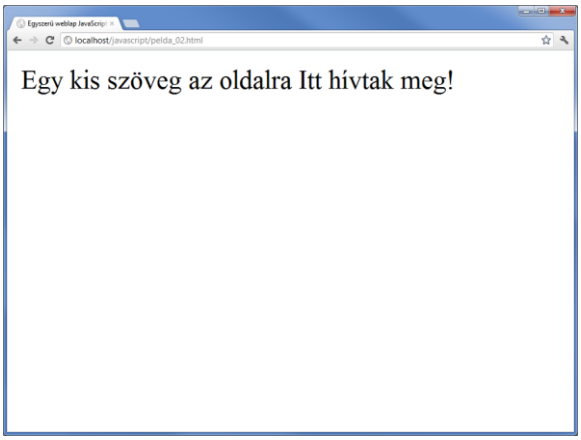
A JavaScript programnyelv egy nagyon egyszerű nyelv, hasonló a HTML nyelvhez. A HTML dokumentumban is elhelyezhetjük a JavaScriptet. Ehhez vizsgáljuk meg az alábbi kódot:

```
149 <html>
150   <head>
151     <title>Egyszerű weblap</title>
152   </head>
153   <body>
154     Egy kis szöveg az oldalra
155   </body>
156 </html>
```

Mint látható, ez egy nagyon egyszerű weboldal forráskódja. Ahhoz hogy ebben a kódban elhelyezünk egy JavaScriptet, a `<script>` elem lesz a segítségünkre. Ennek az elemnek van egy `type` attribútuma, mely a böngészőnek adja át az információt, miszerint JavaScript kódot használunk. Nézzünk erre egy példát:

```
1  <html>
2    <head>
3      <title>Egyszerű weblap JavaScripttel</title>
4    </head>
5    <body>
6      Egy kis szöveg az oldalra
7      <script type=„text/javascript”>
8        document.write(’Itt hívtak meg!’);
9      </script>
10   </body>
11 </html>
```

A `document.write` utasításnak annyi a feladata, hogy a kimenete a böngészőben fog megjelenni. A kimenet látható az alábbi ábrán.



41. ábra: Példakód kimenete

Látható, hogy a kiírás ott jelenik meg, ahol a JavaScript is megtalálható a forráskódban. Ezeket az utasításokat a weboldalban 3 különböző helyre szúrhatjuk be:

- *A weboldal tartalmában.* A <body> és </body> elemek között. Ekkor, mint a példán is látható, ott fog végrehajtódni, ahol található.
- *Az oldal fej részében.* A <head></head> részben. Itt ugyan úgy kell használni, ahogy az előzőekben is. A fejrészben a JavaScript programok viszont nem tartalmazhatnak kimenetet!
- *Elemeken belül.* Ezek az elemek lehetnek a <body> vagy <form> elemek is. Ezeken az elemeken belül már nem kell az utasításokat a <script> elemek közé tenni.

Külső script hívása

Lehetőségünk van a JavaScriptet külön fájlban elhelyezni. Ennek számos előnye van, például, ugyan azt a JavaScriptet több oldalon is fel tudjuk használni, illetve külön tudjuk választani a HTML kódtól, ezzel is elkerülve a forráskód olvashatatlanná válását. A külső JavaScriptes állományok kiterjesztése .js, melyek a <script> elemek közötti utasításokat tartalmazzák.

A külső JavaScriptes állományok használatát az Internet Explorer 4.0-ás változattól, a Netscape Navigator 3.0-ás változattól támogatja. A külső állományok beimportálására nézzünk most egy példát.

```
1  <html>
2    <head>
3      <title>Külső JavaScript importálása</title>
4      <script src=„script.js”
   type=„text/javascript”></script>
5    </head>
6    <body>
7      Egy kis szöveg az oldalra
8    </body>
9  </html>
```

Látható a 4. sorban, hogy a <script> elemek között most nincsen semmi. Mivel külső állományból dolgozunk, így az elem közötti részt figyelmen kívül hagyja a böngésző.

A JavaScriptek feldolgozása, minden esetben a kódban megadott sorrend szerint fognak végrehajtódni. Tehát ha van egy külső script importálva a <head> részbe, majd egy sima <script> elemben az utasítások és végül a body elemben is található JavaScript, akkor ebben a sorrendben fognak az az utasítások is benne végrehajtódni.

10.2.2 A JavaScript működése

Az előző fejezetben megismertük a JavaScript néhány jellemzőjét, de most vizsgáljuk meg közelebbről is.

Függvények

Láthattunk a példában egy sort, ami egy kis magyarázatot igényel.

```
1  document.write('Itt hívtak meg');
```

Ez egy függvény. A JavaScriptben több függvény is szerepel, amelyekből egy párat majd közelebbről is megismerünk a további leckékben.

A függvényeknek lehet paramétere is, ezt zárójelek között kell megadni. Amennyiben több paramétert is fogad, úgy azokat vesszővel választjuk el. A függvény a kapott paraméter alapján dönti el a működését. Vannak olyan függvények is, amelyek valamilyen értéket adnak vissza, ilyen például a következő:

```
1  szam = Math.round(3.14);
```

Ez a függvény visszaadja a 3.14 kerekített értékét a `szam` változóba. Saját magunk is írhatunk függvényeket, amit ha kell, akkor több helyről is meg tudunk hívni, anélkül, hogy a kódot újra és újra begépelnénk.

Objektumok

Mint azt az előző részben láttuk, a változó csak egy értéket tud tárolni. Az objektum ezzel szemben több adat tárolására is alkalmas. Ezeket az adatokat tulajdonságnak nevezzük. Egy objektum lehet egy autó, és ennek lehetnek különböző tulajdonságai. A tulajdonságokat az objektum nevétől egy pont választja el. Ilyen például: `auto.rendszam` vagy `auto.szin`

Ezek mellett nem csak tulajdonságokat tartalmazhatnak, hanem az előzőleg már megismert függvényeket is. Ezek a függvények az objektum adataival dolgoznak. Így lehetőségünk van létrehozni például egy `beindit` függvényt is, aminek a meghívása `auto.beindit()`.

A JavaScriptben 3 féle objektumot használhatunk:

- *Beépített objektumok.* Ezek a JavaScript részei. Ilyen például a `String`, `Date`, `Math` objektumok.
- *Böngésző objektumok.* Ezek a böngésző és a HTML dokumentum részét képezik. Ilyen például az `alert()` vagy a `confirm()` függvény. Mind a kettő a `window` objektum része.
- *Egyedi objektumok.* Ezek pedig olyan objektumok, amiket mi hozunk létre, például az előzőleg emlegetett `auto` objektum.

Az `alert()` függvény a paraméterben megadott szöveget jeleníti meg egy kis ablakban, és egy *OK* gombot.

A `confirm()` pedig hasonló kis ablakot készít mint az előző függvény, csak megjelenik az *OK* mellett egy *Mégsem* gomb. A függvény visszatérési értéke logikai igaz vagy hamis, a gombra kattintástól függően.

Események kezelése

A JavaScript program elhelyezése kapcsán szó esett az eseménykezelőkről is. Az eseménykezelők mindig valamilyen böngészőobjektumhoz kapcsolódnak. Többféle esemény létezik a JavaScriptben. Van például az `onLoad` mely akkor hajtódik végre, amikor az oldal betöltődött. Vannak egérre vonatkozó eseménykezelők: `onMouseOver`, `onMouseOut`, `onClick`. Lehet alkalmazni eseménykezelőt beviteli mezőre is. Belépéskor az `onEnter`, elhagyáskor az `onLeave` esemény fog bekövetkezni.

A következő kódrészlet feldob egy kis ablakot, ha az oldal betöltődött.

```
1  <body onLoad=„alert('Az oldal betöltése
   kész!');”>
```

Ha több utasítást is szeretnénk végrehajtani egy adott eseményre, akkor célszerűbb rá függvényt írni, mivel annak a neve rövidebb, és jobban átlátható kódot eredményez, nem utolsó sorban több helyen is alkalmazható.

Feltételes utasítások

A JavaScriptben lehetőségünk van egy bizonyos programrészletet végrehajtani attól függően, hogy egy bizonyos feltétel teljesül-e vagy sem. Ezt a feltételes utasítások használatával tudjuk megoldani. Ehhez a JavaScriptben az `if` kulcsszót kell használnunk. Később erről még lesz szó. Itt látható egy egyszerű példa:

```
1  if (szam > 2) alert('A szám nagyobb, mint ket-
   tő');
```

A fenti kód feldob egy ablakot, ha a `szam` változó értéke nagyobb, mint kettő.

Ciklusok

Mint ahogy a legtöbb programozási nyelvben, itt is van lehetőségünk ciklusokat használni. Ez nagyon hasznos, ha valamit többször szeretnénk végrehajtani. Példánkban kiírjuk az első 10 pozitív egész számot.

```
1  for (i=1;i<=10;i++) {
2    document.write(i + '<br />');
3  }
```

Megjegyzések

Mint minden programozási nyelvben, a JavaScriptben is használhatunk megjegyzéseket. Aki már valamennyire járatosak a C nyelvben, azoknak ez nem lesz új. A kódban a megjegyzésekkel egyes kódrészleteket tudunk hatástalanítani, vagy megjegyzéseket tudunk fűzi a kódhoz. A megjegyzések elhelyezéséhez kezdjük a sort két / jellel. Ez annyit jelent, hogy a perjelek utáni részt a JavaScript nem veszi figyelembe. Ennek értelmében akár a sor végére is tehetjük őket.

```
1  // Ez a sor soha nem fog végrehajtódni
2  document.write('Szia!'); // Ez kiírja hogy Szia!
```

Ha egy kódrészletet szeretnék elrejteni, akkor használjuk a /* */ jeleket. Ezekkel több sort is át tudunk rakni megjegyzésbe.

```
1  /*
2  A következő programkód üdvözli a felhasználót
3  alert('Ez nem hajtódik végre!');
4  */
5  nev = 'Ádám';
6  document.write('Szia '+nev);
```

10.2.3 Függvények, változók, adattípusok

Függvények

Az eddig itt bemutatott kódokban csak utasítások szerepeltek. A forráskód könnyebb olvashatósága érdekében rendszerezhetjük az utasításainkat különböző függvényekbe. A függvények, olyan utasítások csoportja, amit egy egységként kezelhetünk. Az egyes függvényeket a meghívásuk előtt meg kell határoznunk. Ez a `function` kulcsszóból, a függvény nevéből, esetleges paraméterlistájából és a törzséből áll.

```
1  function Udvozles() {
2    alert('Szia!');
3  }
```

Láthatjuk a függvény nevét, mely az `Udvozles` nevet kapta. A JavaScript érzékeny kis- és nagybetűkre, ezért ha meghívjuk később ezt a függvényt, akkor ugyan ebben a formában (`Udvozles`) kell rá hivatkoznunk. A függvény neve utáni zárójel kötelező. Amennyiben olyan függvényt írunk, amelynek van paramétere úgy az egyes paramétereket a két zárójel között kell felsorolnunk. A függvény törzsét, azaz az utasítások sorozatát kapcsos zárójelek között kell megadni.

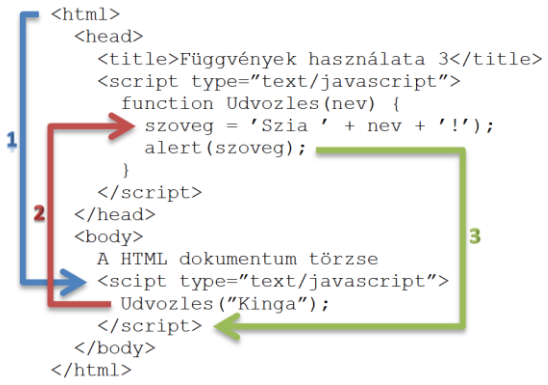
Célszerű ezeket a függvény meghatározásokat a HTML dokumentum `<head>` részében elhelyezni, így amikor használni akarjuk a dokumentumban, akkor már biztos, hogy léteznek.

```
1  <html>
2    <head>
3      <title>Függvények használata</title>
4      <script type=„text/javascript”>
5        function Udvozles(nev) {
6          alert('Szia ' + nev + '!');
7        }
8      </script>
9    </head>
10   <body>
11     A HTML dokumentum törzse
12   </body>
13 </html>
```

A fenti HTML dokumentumot betöltve a böngészőbe, semmi nem fog történni. Ahhoz hogy a függvényünk produkáljon eredményt, meg kell hívni valahol. A függvény meghívása úgy történik, hogy valahol a dokumentumban utasításként elhelyezzük a függvény nevét és paraméterét, ha van.

```
1  <html>
2    <head>
3      <title>Függvények használata</title>
4      <script type=„text/javascript”>
5        function Udvozles(nev) {
6          alert('Szia ' + nev + '!');
7        }
8      </script>
9    </head>
10   <body>
11     <scipt type=„text/javascript”>
12       Udvozles(„Kinga”);
13       Udvozles(„Béla”);
14     </script>
15   </body>
16 </html>
```

Megfigyelhetjük a példában, hogy egyszer írtuk meg a függvényt, de azt kétszer is használtuk más paraméterrel.



42. ábra: JavaScript feldolgozása

Ha olyan függvényt szeretnénk írni, amely valamilyen értéket visszaad, akkor egy újabb kulcsszót kell majd használnunk a függvény törzsében, ez pedig nem más, mint a `return`.

```
1  function terulet(a, b) {
2    return (a*b);
3  }
```

A függvény visszatérési értéke a paraméterben megkapott két szám szorzata lesz. Nézzük meg a gyakorlatban is hogy néz ez ki.

```
4  <html>
5    <head>
6      <title>Függvények használata 2</title>
7      <script type=„text/javascript”>
8        function terulet(a ,b) {
9          return (a + b);
10       }
11     </script>
12   </head>
13   <body>
14     <scipt type=„text/javascript”>
15      document.write('Terület: ' + terulet(2,5));
16     </script>
17   </body>
18 </html>
```

Változók

Az előző fejezetekben már találkozhattunk a változó szóval, de egy- két fontos dolgot még meg kell említeni róla. A változók neveinél meg van határozva, hogy miből épülhet fel. Ezek a következők, amire jó hogyha odafigyelünk:

- A változók neveiben csak az angol abc kis- és nagybetűit, a számjegyeket 0-9-ig, és az _ jelet. Ezen kívül semmi mást!
- A változónevek első karaktere nem lehet szám!
- A változónevek úgy, mint a függvények nevei is kis- nagybetű érzékenyek, azaz a `szam` változó nem ugyan az, mint a `Szam` vagy a `SZAM`!

Lehetőségünk van létrehozni különböző hatókörű változókat. Azaz meg tudjuk határozni, hogy egy változó globálisan elérhető legyen a dokumentumon belül, vagy csak lokálisan például egy függvény belsejében.

A lokális változók neveit a programban `var` kulcsszó előzi meg. Ha a változót függvényeken kívül használjuk, akkor elhagyható a kulcsszó. Ha a függvény törzsében hozunk létre egy változót a `var` kulcsszóval, akkor az csak ott használható, a függvényen kívül már nem. A függvények paraméterében átadott változók is lokálisak.

Ha szeretnénk értéket adni egy változónak, akkor a sima egyenlőség jelet kell használnunk. Tehát az egyenlőség jel bal oldalán áll a változó neve, a bal oldalán pedig az érték vagy kifejezés. Ezeket persze lehet variálni is.

```
1  szam = 12;
2  szam = szam + 1; // Itt növeljük az értékét
3  szam += 1; // Ez ugyan az, mint az előző
4  szam -= 1; // Itt 1-el csökkentjük az értéket
5  szam++; // Ez ugyan az, mint ami 3. sorban van
6  szam--; // Ez a 4. sorral egyezik meg
```

Adattípusok

Vannak olyan programozási nyelvek, ahol meg kell határoznunk előre, hogy milyen adattípussal szeretnénk dolgozni. Ilyen például a C# vagy a Pascal. A JavaScriptben nem kell ilyenekkel foglalkoznunk, de azért nézzük át, milyen adattípusok léteznek a JavaScriptben.

- *Számok*: Ide tartoznak, az egész számok, és a lebegőpontos számok is. Például: `1`, `6` vagy a `3.1415`
- *Logikai változók*: Ide a `true` és `false` logikai értékek tartoznak.
- Karakterláncok: Ezek valójában a szövegek, amik több karakterből tevődnek össze. Például „`alma a fa alatt`”
- *`null`*: ez egy speciális érték, ez jelenti a még nem meghatározott változókat.

A JavaScript az egyes adattípusok közötti átalakításról megpróbál önállóan gondolkodni. Ez némely esetben probléma nélkül fog működni.

```
1  document.write('Összesen: '+total);
```

Itt a `total` változóban tárolt számot automatikusan szöveggé alakítja. Előfordulhatnak olyan esetek is, amikor nem bízunk meg a kapott változó típusában és szeretnénk az átalakítani. Erre a célra szolgál két függvény a `parseInt()` és a `parseFloat()`. Míg az első egy szöveget alakít át számmá, az utóbbi szövegből készít lebegőpontos értéket.

Mind a két függvény addig olvassa be a karaktereket a szöveg elejétől, amíg az szám vagy pont. Utóbbi a csak a `parseFloat()` függvényre igaz.

10.2.4 Műveletek különböző adattípusokkal

String objektum

A String objektum valójában karakterek halmaza, más néven karakterlánc. Ennek az objektumnak vannak különböző beépített függvényei, amivel manipulálhatjuk a szöveget.

Ahhoz hogy létrehozzunk egy karakterláncot, nem kell mást tennünk, mint egy változónak értékül adni a szöveget.

```
1  szoveg = „Ez egy szöveg”;
```

Ha szeretnénk összefűzni két szöveget, akkor használjuk az összeadásból már jól ismert + jelet. Ezt két érték összeadására is használjuk, és arra is, hogy szövegeket fűzzünk össze vele.

```
1  mind = egyik + masik;
```

Ha szeretnénk megtudni, hogy egy szöveg hány darab karakterből áll, akkor használjuk a `length` jellemzőt.

```
1  hossz = szoveg.length;
```

Ha a `szoveg` változó az „`Alma a fa alatt`” szöveget tartalmazza, akkor a `hossz` változó értéke 15 lesz.

Előfordulhat olyan speciális eset, amikor egy szöveget csupa kisbetűssé vagy nagybetűssé kell alakítanunk. Erre a problémára is vannak beépített függ-

vények. A nagybetűs átalakításra használható a `toUpperCase()`, a kisbetűs konverzióra pedig a `toLowerCase()` függvény.

```
1  szoveg = 'Alma';
2  document.write(szoveg.toUpperCase()); // ALMA
3  document.write(szoveg.toLowerCase()); // alma
```

Az előző példában a `szoveg` változó tartalma megmarad, nem íródik felül. A felülíráshoz a következő parancsot kell kiadnunk.

```
1  szoveg = szoveg.toUpperCase();
```

Ha megfigyeljük, akkor itt a már függvényeket használunk, azaz megjelennek a zárójelek. A szöveg hosszának lekérdezésekor pedig csak a változó egy tulajdonságát kértük le.

Szövegeknek egy részét is ki tudjuk nyerni a `substring()` függvénnyel. Két paramétert kell neki megadni, hogy hányadik karaktertől, hányadik karakterig szeretnénk kimásolni a szöveget.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	**l**	**m**	**a**		**a**		**f**	**a**		**a**	**l**	**a**	**t**	**t**

43. ábra: Egy karakterlánc indexelése

Figyeljük meg, hogy a karakterek számozása 0-tól kezdődik!

```
1  szoveg = „Alma a fa alatt”;
2  document.write(szoveg.substring(0,4)); // Alma
3  document.write(szoveg.substring(7,9)); // fa
```

Amennyiben csak egy karakterre van szükségünk, akkor használhatjuk a `charAt()` metódust is. Itt paraméterben meg kell adni, hogy hányadik karaktert szeretnénk visszakapni.

```
1  szoveg = „Alma a fa alatt”;
2  document.write(szoveg.charAt(2)); // m
```

Arra is van lehetőségünk, hogy a szövegben megkeressünk egy bizonyos szövegrészletet. Erre az `indexOf()` függvényt használjuk. Ennek a függvénynek van 1 és 2 paramétert váró változata is. Ha csak egy paraméterrel hívjuk meg, akkor visszakapjuk a megadott szövegrészlet első előfordulási helyét a szövegben. Ha például a karakterlánc első 10 karakterében nem akarunk keresni, akkor célszerű kitölteni a második paramétert, amivel elérhetjük, hogy a keresés a 9. karaktertől induljon.

```
1  szoveg = „Alma a fa alatt”;
2  document.write(szoveg.indexOf('a')); // 3
3  document.write(szoveg.indexOf('a',5)); // 7
```

Ha valamilyen okból az utolsó találat helyét szeretnénk megtudni, akkor használjuk a `lastIndexOf()` függvényt. Használata hasonló az `indexOf()` függvényéhez, ugyan úgy létezik 1 és 2 paraméteres változata.

```
1  szoveg = „Alma a fa alatt”;
2  document.write(szoveg.lastIndexOf('l')); // 11
3  document.write(szoveg.lastIndexOf('l',10)); // 1
```

Az utóbbi két függvénynél figyeljünk oda a helyesírásra, mivel a JavaScript kis- és nagybetű érzékeny, ezért ha nem megfelelően írjuk be a függvényneveket, akkor hibajelzést kapunk!

Tömbök

A tömb egy olyan csoport, ahol minden egyes elem számozva van, és ezek az elemek több különböző típusúak is lehetnek. Tartalmazhatnak számokat vagy szövegeket is. Egy új tömböt a `new` és az `Array()` kulcsszóval tudunk létrehozni.

```
1  tomb = new Array(4);
```

Mint látható az `Array()` paramétereként meg lett adva a tömb hossza. Ha nem tudjuk előre, hogy hány elemből fog állni, akkor ez elhagyható.

A tömb elemeinek a feltöltése, 3 féle képen is történhet.

```
1  tomb[1] = 23;
2  tomb[2] = 45;
3  tomb[3] = 67;
4  tomb[4] = 89;
5
6  tomb = new Array(23,45,67,89);
7
8  tomb = [23,45,67,89];
```

A tömb hosszát a karakterláncoknál már megismert `length` tulajdonsággal kérdezhetjük le.

```
1  tomb = new Array(4);
2  document.write(tomb.length); // 4
```

A tömb egyes elemeinek értéke a tömb nevének és index számának megadásával kérdezhető le.

```
1  document.write(tomb[3]); // 67
```

Ha a tömbben nem számot, hanem szöveget szeretnénk tárolni, akkor ennek semmi akadálya. Létrehozás, feltöltés, lekérdezés ugyan úgy működik, sőt még a karakterlánc kezelő függvényeket is használhatjuk.

```
1  sz = new Array(2)
2
3  sz[1] = „Béla”;
4  sz[2] = „Kinga”;
5
6  sz = new Array(„Béla”, „Kinga”);
7
8  sz = [„Béla”, „Kinga”];
```

```
9
10 document.write(sz[1].substring(2,3); // ng
```

Ha van egy olyan szövegünk, ami mondjuk az Excelből kimentve CSV (pontosvesszővel tagolt) fájlba, akkor azt is könnyen fel tudjuk dolgozni. Ehhez a `split()` függvényt kell használnunk.

```
1  szoveg = „Kis Béla;Eger;12.025 Ft”;
2  darab = szoveg.split(„;”);
```

Ezen utasítások végrehajtása utána a `darab` tömb a következő értékeket fogja felvenni.

```
1  darab[0] = „Kis Béla”;
2  darab[1] = „Eger”;
3  darab[2] = „12.025 Ft”;
```

Ennek a fordítottja is elérhető a `join()` függvénnyel, azaz a 3 tömbelemből tudunk létrehozni egy új karakterláncot.

```
1  szoveg = darab.join(„;”);
```

Mind a két függvény paraméterében meg kell adni a szétválasztó, vagy összefűző karaktert.

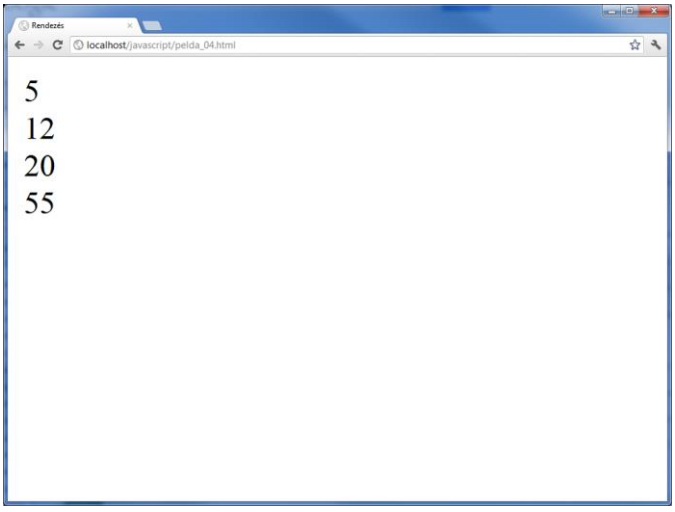
A JavaScript `sort()` függvényével abc sorba tudjuk rendezni a tömb elemeit.

```
1  tomb[1] = „Banán”;
2  tomb[2] = „Alma”;
3  tomb[3] = „Citrom”;
4  sorba = tomb.sort();
5  // A végeredmény: Alma, Banán, Citrom
```

Azt gondolnánk, hogy ez ugyan úgy igaz a számokat tartalmazó tömbre is. Pedig nem!

Sajnos a számokat is szövegként érzékeli. Ezért a 20, 5, 55, 12 elemű tömböt a következő képen rendezi: 12, 20, 5, 55. A helyes működéshez egy függvényt kell írnunk. A függvényben egy műveletet kell elvégeznünk, amit a sorba rendezéskor a függvény végrehajt. Ezzel akár módosítani is tudjuk a sorba rendezési eljárást.

```
1  function kivon(a, b) {
2    return a-b;
3  }
4  tomb = new Array(20,5,55,12);
5  rendezve = tomb.sort(kivon);
```



44. ábra: Számokból álló tömb rendezése

A kivon függvény 0, negatív vagy pozitív számmal tér vissza. Ebből már tudja a rendező függvény, hogy a két szám hogy viszonyul egymáshoz.

10.2.5 Vezérlési szerkezetek, ciklusok

Feltétel vizsgálat

A feltétel vizsgálat minden fontosabb programozási nyelvben megtalálható. Ezzel a programozási szerkezettel tudjuk szabályozni, hogy egy adott feltétel teljesülése vagy nem teljesülése esetén hogy viselkedjen a program. A két kulcsszó, az `if` és az `else`. Felépítése a következő:

```
1  if (feltétel) {
2    // Ha teljesül a feltétel
3  } else {
4    // Ha nem teljesül a feltétel
5  }
```

A feltétel bármelyik oldalán állhat változó, állandó vagy kifejezés. Összehasonlíthatunk akár két változót, vagy egy értéket és egy állandót is.

```
1  if (a == 1)
2    alert('Az a változó értéke 1');
3
4  if (num > num2) {
5    num2 = num;
6    alert('Az érték átadva');
7  }
```

Ha csak egy utasítás szerepel az `if` vagy `else` részben, akkor a kapcsos zárójelek elhagyhatóak.

Ne keverjük össze az értékadó és egyenlőség vizsgálat műveleti jeleit. Az értékadó műveleti jel egy darab egyenlőségjel, míg az egyenlőség vizsgálat jele kettő darab egyenlőség jel!

A két érték között, amit szeretnénk összehasonlítani többféle műveleti jel is állhat:

- == (egyenlő)
- != (nem egyenlő)
- < (kisebb mint)

- > (nagyobb mint)
- < (kisebb vagy egyenlő)
- > (nagyobb vagy egyenlő)

Az egyes feltételeket logikai műveleti jelekkel is összefűzhetjük. Erre használhatjuk az és (`&&`) műveleti jelet és a vagy (`||`) műveleti jelet.

```
1  if (nev != „„) && (cím != „„)
2    alert('Töltsön ki minden mezőt');
```

Ha nem akarunk sokat gépelni, programozás közben, akkor van lehetőségünk a feltételeket röviden megfogalmazni.

```
1  szam = 12;
2  valtozo = (szam % 2 == 0)?”paros”:”paratlan”;
```

A fenti példában látható a „rövid if” használata. Itt azt fogjuk vizsgálni, hogy a `szam` változó 2-vel való osztási maradéka egyenlő-e nullával. Ha igen, akkor a `valtozo` felveszi a `páros` értéket, egyébként a `páratlan`-t. A könnyebb érthetőség kedvéért lássuk ugyan ezt a példát a hagyományos módon.

```
1  szam = 12;
2  if (szam % 2 == 0)
3    valtozo = „paros”;
4  else
5    valtozo = „paratlan”;
```

Amennyiben több feltételt is kellene vizsgálnunk, használhatjuk a `switch` szerkezetet. Erre lássunk egy példát:

```
1  switch (ertekeles) {
2    case „1”: jegy = „elégtelen”; break;
3    case „2”: jegy = „elégséges”; break;
4    case „3”: jegy = „közepes”; break;
5    case „4”: jegy = „jó”; break;
6    case „5”: jegy = „jeles”; break;
```

```
7    default: jegy = „Ilyen jegy nincs!”
8  }
```

A `switch` kulcsszó után zárójelben kell megadnunk a vizsgálni kívánt értéket. Az egyes elágazásokat a `case` kulcsszó előzi meg, majd utána az érték, amit összehasonlítunk a `switch`ben megadott értékkel. Ha teljesül valamelyik ág, akkor végrehajtódnak az adott `case` ágban található utasítások. A legutolsó elágazás egy `default` ág. Ez akkor fog végrehajtódni, ha egyik ág sem fog teljesülni. Ennek használata nem kötelező.

Az `case` egyes ágai mindaddig végrehajtódnak, míg vége nincs a `switch` blokknak, vagy `break` kulcsszót nem találnak.

Ciklus

Ha többször szeretnénk megismételni egy utasítást, akkor ciklusba kell szerveznünk őket. Ennek kulcsszava a `for`. Az egyes paramétereket pontosvesszővel kell elválasztanunk. Ezek a következőek:

- kezdőérték: `i = 1`
- feltétel: `i < 10`
- léptetés: `i++`

A paramétereket zárójelekben kell megadni, majd utána kapcsos zárójelekben soroljuk fel az utasításokat, amit szeretnénk megismételni.

A léptetés nem csak növekményes lehet, hanem csökkenő is.

```
1  for (i = 0; i < 10; i++)
2    document.write(i);
3
4  for (i=10; i > 0; i--) {
5   document.write(i);
6  }
```

Léteznek a JavaScriptben olyan ciklusok is, amelyek addig futnak, míg a feltétel nem teljesül. Ez a `while` ciklus. Ha már a ciklus elején nem teljesül a feltétel, akkor a ciklus egyszer sem fog végrehajtódni.

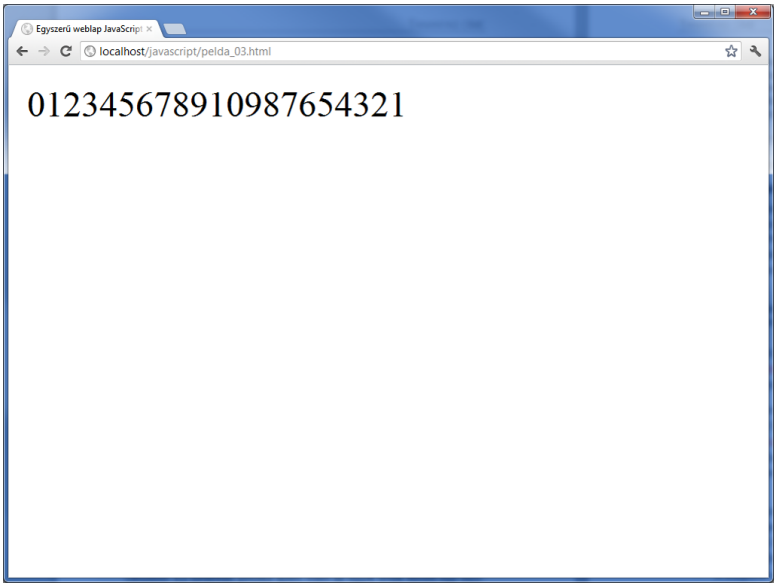
```
1  while (i < 10) {
2    document.write(i);
3    i++;
4  }
```

Amennyiben elfelejtjük beleírni, a `i++` növekményt, úgy végtelen ciklusba fogunk futni.

Ha szeretnénk, hogy a ciklus minimum egyszer lefusson, úgy használjuk a `do..while` ciklust. Ennek felépítése ugyan az, mint a sima `while` ciklus esetében.

```
1  do {
2    document.write('.');
3    total ++;
4  } while (total == 1);
```

Arra is van lehetőségünk, hogy valamilyen okból kifolyólag, kilépjünk a ciklusból. Ez történhet például azért, mert az egyik érték felvett egy bizonyos értéket. Ilyenkor a `break` kulcsszóval tudjuk megszakítani a ciklust, és a ciklus utáni utasításnál folytatni a futtatást.



45. ábra: A „for” ciklus kimenet

Ha nem akarunk kilépni a ciklusból, csak át akarjuk ugorni a ciklusban lévő további utasításokat, akkor a `continue` kulcsszót kell használni. Ez nem lép ki a ciklusból, hanem folytatja a ciklust az elejétől.

10.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

10.3.1 Összefoglalás

A fejezetben áttekintésre kerültek a JavaScript parancsnyelvvel kapcsolatos alapvető tudnivalók, bemutatásra került annak működése, elhelyezésének módja a weboldalon belül és külső scriptként. A tanuló megismerkedhetett a függvények, objektumok és események kezelésével, az adattípusokkal és az azokon végezhető műveletekkel, továbbá a vezérlési szerkezetek leírásával.

10.3.2 Önellenőrző kérdések

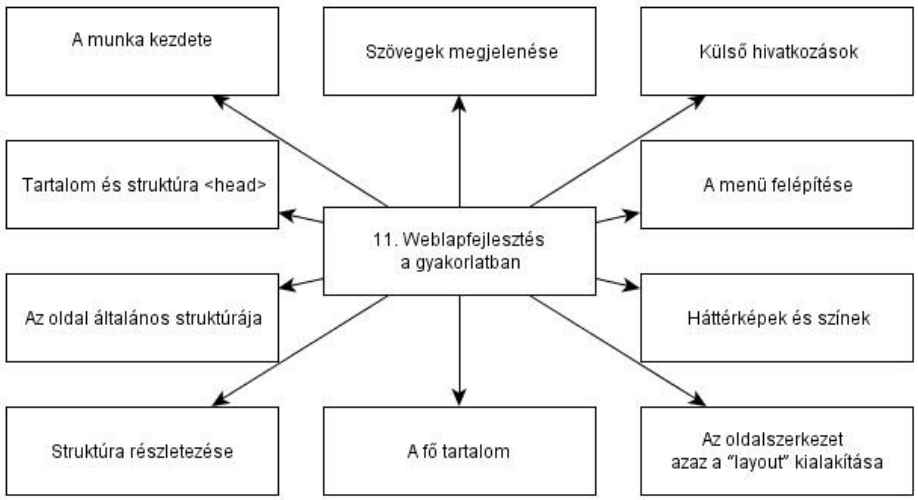
1. Tartalmazhat-e a weboldal több <script> elemet?
2. A JavaScript, `document.write` utasításában használhatok HTML elemeket?
3. Mire használható a `var` kulcsszó?
4. Miben különböznek a különböző ciklusok a JavaScriptben?
5. Mondja el a különféle elágazási lehetőségek szintaxisait!

11. WEBLAPFEJLESZTÉS A GYAKORLATBAN

11.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Cél, hogy a tanulók folyamatában lássanak egy weboldal elkészítését a tervezési folyamattól kezdve a kivitelezésig.

11.2 TANANYAG



46. ábra: Fogalomtérkép

11.2.1 A munka kezdete

A következőkben egy weboldalt fogunk elkészíteni az eddig elsajátított ismeretekre támaszkodva. Ez egy statikus oldalból egy lap. Az oldalszerkezet kialakításánál feltételezzük, hogy az összes többi lap is erre a sémára épül. Az oldal tartalma kizárólag szöveges jellegű, de természetesen jelentésében megkülönböztetett. A tartalom kialakításához XHTML-t használunk, amely napjainkban is elfogadott jól bevált forma.

A gyakorlatban, persze ideális esetben az oldal megjelenésének azaz dizájnjának megtervezése egy külön fázis amit vagy a webfejlesztő, vagy egy dizájner készít, így a fejlesztő már a kész nyersanyaggal dolgozhat. Ez a grafikai

terv optimális esetben valamely rasztergrafikus szerkesztőprogram forrásállománya, rosszabb esetben egy jó minőségű kép.

Az oldal másik összetevője maga a tartalom amelyet a megrendelő bocsájt a rendelkezésünkre. A tartalom megtervezése meghatározó jelentőséggel bír, így ennek a folyamatában a fejlesztőnek is részt kell vennie és szakmai tanácsokkal kell ellátnia a tartalom szervezőt. Természetesen ez alapján valósulhat meg az oldal struktúrájának megtervezése is ami a laikusok számára többnyire a menüpontok és az azok alatt található tartalmak elkülönítéséből áll.

Miután ezeket a fejlesztő rendelkezésére bocsájtották, csak ekkor indulhat meg az oldal konkrét készítése. Jelen esetben a tartalom XHTML struktúrába szervezése, és a megjelenés kialakítása CSS és a rendelkezésre álló grafikai állományok felhasználásával.

Mivel a kezdőoldalt fogjuk elkészíteni, a weblapfejlesztésben alapértelmezett kezdőoldalnak számító index elnevezést adjuk ennek a fájlnak (a webszerverek alapértelmezésként az index.html elnevezésű fájlokat keresik egy könyvtárban, ha a webcímben nem szerepel konkrét fájlnév).

A gyakorlat forrásanyaga letölthető az alábbi címről:
www.ektf.hu/letoltheto-dokumentumok/weblap_gyakorlat.zip

47. ábra: A kész oldal

11.2.2 Tartalom és struktúra <head>

A fejlesztéshez az XHTML 1.0 verziójának Strict típusát használjuk. Ez a verzió kifejezetten csak az XHTML elemeit engedi meg, így a régi HTML szabványból használt elemek esetén ez hibát feltételez. Hasznos lehet ugyanis a fejlesztés alatt így ponosabban és fegyelmezettebben alkothatjuk meg az oldal alapjait. Így a dokumentumunk legelején az alábbi kód közli az oldalt megjelenítő eszközzel, hogy milyen tartalomra számíthat:

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

A html dokumentum kezdetén a html-hez megadjuk az ehhez megfelelő attribútumot:

```
<html xmlns=„http://www.w3.org/1999/xhtml” lang=„hu” xml:lang=„hu”>
```

A dokumentumunk fejrészében csak a legszükségesebb információkat adjuk meg.

Karakterkódolás

Elsőként a megfelelő karakterkódolás beállításához az ezt szabályozó meta jelölőt használjuk. A dokumentumunk a jelenkori normáknak megfelelően az UTF-8 kódolást használja ezzel megfelelő átjárást biztosít esetleges további fejlesztésnél használt más technológiák közt, valamint biztosítja, hogy más nyelvterületeken élők is helyes képet kapjanak az oldalon használt szövegről. Nagyon lényeges, hogy nem elegendő csak a fejrészben jelölni az UTF-8 használatát, hanem magát a dokumentumot is ebben kell elkészíteni. Ezt bármely manapság használatos modern fejlesztőeszköz biztosítani tudja.

```
<meta http-equiv=„Content-Type” content=„text/html; charset=utf-8” />
```

Az oldal címe

A megfelelő cím megválasztása keresőmarketing szempontjából kiemelt jelentőséggel bír. A cím megadásának elmulasztása nemhogy rossz gyakorlat, hanem az XHTML-ben végrehajtott komoly hiba. Oldalunk az internetmarketing témakörével foglalkozik, így ez lesz az oldal címe is.

```
<title>Internetmarketing</title>
```

CSS hivatkozás

A megjelenés-tartalom-működés hármas elkülönítéséhez természetesen a CSS kód külön állományban helyezkedik el. Ennek használatához az XHTML dokumentumban el kell helyeznünk a css dokumentumra történő hivatkozást. A böngészők ezt fogják használni az oldal megjelenítésekor.

```
    <link      href=„stilus.css”      rel=„stylesheet”
type=„text/css” />
```

11.2.3 Az oldal általános struktúrája

Az oldal struktúráját megvizsgálva négy részre oszthatjuk azt fel. Az oldal megjelenése áttekinthető és viszonylag könnyen megfigyelhetők ezek az elkülönülő részek.

Az első a fejléc, ami tartalmazza az oldal felhasználók számára megjelenített címét. Ez más oldalakon lehet egy logó vagy bármely más szimbólum, esetünkben azonban egy egyszerű szöveg. Alatta helyezkedik el a menüsáv melyben az oldalon történő navigációhoz elengedhetetlen menüpontokat találjuk.

Az oldal jelentős részét a tartalmi terület teszi ki, amely feladata a közlendők és az oldalon található információk megjelenítése. Esetünkben elmondható, hogy az oldal egyedüli olyan része, amely menüpontonként változik, a többi változatlan, így egységes képet kapunk az oldalon történő böngészés során. A tartalmi terület automatikusan igazodik a benne megjelenített adatok mennyiségéhez.

Végül a tartalmi terület alatt az oldal legalsó részében találjuk a láblécet. Az oldalunk esetében nem tartalmaz lényeges információt, illetve annak mennyisége is elenyésző. Általánosan azokat a kevésbé fontos közleményeket érdemes itt elhelyezni, ami nem kiemelt jelentőségű, ugyanis a weboldalak felépítéséből fakadóan látogató mindig az oldal tetejéről kezdi a tájékozódást és bizonyos képernyőfelbontás esetén ez kilóghat a képernyőből is.

A lényegesebb része k után tovább kell bontanuk a tartalmi részt két oszlopra, ahogy az a grafikai terven látható. A bal oldali nagyobb rész az ami ténylegesen a lényegi információt hordozza. A jobb oldali keskenyebb oldalsávban külső hivatkozásokat találhatunk. Az oldalsáv tartalma is eltérhető oldalanként vagy oldaltípusonként.

Az oldal felosztása

A különböző részek kialakítását div elemekkel oldjuk meg. Az XHTML-ben ez az elem kifejezetten erre a célra szolgál. A div bármely más típusú elemet tartalmazhatja magában. Alapvetően blokk szintű elemnek számít, így a doboz

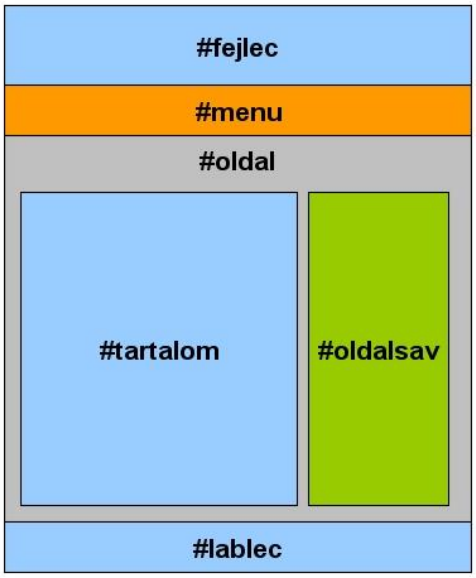
modell szabályai ekképpen vonatkoznak rá. A különböző blokkokat egyéni azonosítóval látjuk el, ezzel jelölve a struktúrában az adott blokk betöltött szerepét. Másik, talán még ennél is lényegesebb szerepe az, hogy ez alapján hivatkozhatunk rá a CSS-ből. Egyéni azonosítókat használunk, hiszen az adott blokkok csak egyszer fordulnak elő az oldalon, így ezzel egyértelműen meghatározhatók. Erre a célra az id attribútumot alkalmazzuk, és egy általunk választott beszédes elnevezéssel látjuk el. Az azonosító megadásánál ügyeljünk az XHTML szabályaira, miszerint az attribútumokat csupa kisbetűvel írjuk és azok értékeit idézőjelek közt adjuk meg. Az azonosítók nevét is csupa kis betűvel, kizárólag az angol ABC betűit használva válasszuk meg.

A blokkok elnevezése az alábbiak szerint alakul:

1. Fejléc - „fejlec”
2. Menüsáv - „menu”
3. A tartalmat magában foglaló blokk - „oldal”
    3.1. A fő tartalémi rész - „tartalom”
    3.2. Jobb oldali oldalsáv - „oldalsav”
4. Lábléc - „lablec”

Az XHTML kód csak vázszerűen tartalom nélkül:

```
<div id=„fejlec”>...</div>
<div id=„menu”>...</div>
<div id=„oldal”>
  <div id=„tartalom”>...</div>
  <div id=„oldalsav”>...</div>
</div>
<div id=„lablec”>...</div>
```



48. ábra: Oldalszerkezet

11.2.4 Struktúra részletezése

Fejléc

A példafeladatban a fejléc két dologra hivatott. Egyrészt tartalmazza az oldal címét a látogatók számára esztétikus módon, illetve még egy rövid üzenetet az oldal jellegére utalva. Ez más oldalakon lehet mottó vagy valamilyen jelmondat is. Az oldal fejlécének kiemelt szerepe van a megjelenés szempontjából, ugyanis ez az egyik leg meghatározóbb momentum. Ez helyezkedik el az oldal tetején, így a látogató ezzel találkozik legelőször. Sokszor törekednek rá, hogy ez megkapó legyen, és ezzel jelentős módon hozzájáruljon az élményhez.

A keresőmarketing szempontjából már nem alkalmazzuk azt a gyakorlatot, hogy ez a cím legyen az elsődleges címsor, ezt ugyanis meghagyjuk a tartalomra való súlyozott utalásnak. Egyszerűen egy bekezdést használunk erre a címre utaló osztálynév kiválasztásával.

```
<p class=„cim”>internetmarketing</p>
```

Az üzenetet is hasonlóképpen helyezzük el a tartalomban, természetesen másik osztálynevet alkalmazva, ebben az esetben „uzenet”.

```
<p class=„uzenet”>marketing, online marketing,
keresőmarketing</p>
```

Menüsor

A menüsornak szintén kitüntetett szerepe van az oldalon, ugyanis a navigáció ezzel valósul meg. Az elhelyezésénél figyelnünk kell rá, hogy egyértelmű és feltűnő helyen szerepeljen, ne kelljen keresgélni az oldalon. Használata is egyértelmű kell, hogy legyen, lehetőség szerint kerüljük a bonyolult egérmozgásokat és kézügyességet feltételező megoldásokat. Annál felhasználóbarátabb egy menü megoldás minél átláthatóbb, így próbáljuk minimális szinten tartani a menüszintek számát. A példafeladatban egy egyszerű egyszintes menüt alkalmazunk, ugyanis a tartalom nem indokolja ennek bonyolítását.

A menü tartalom jellegéből fakadóan nem csak hivatkozások egymás mellé helyezve. Ennek megfelelő struktúrája a lista. Általánosan ezt a megoldást alkalmazzák menü felépítésére, pontosabban a számozatlan listát. A felhasználók számára persze ez nem lesz észrevehető, erről majd a CSS kódban gondoskodunk. Tekintve, hogy a példa csak egy oldal elkészítését indokolja, nem létező oldalakra fogunk hivatkozni a menüben.

```
    <ul>
        <li><a href=„index.html”>Kezdőoldal</a></li>
        <li><a    href=„keresok.html”>Kereső    marke-
ting</a></li>
        <li><a
href=„webergonomia.html”>Webergonómia</a> </li>
        <li><a
href=„kapcsolat.html”>Kapcsolat</a></li>
    </ul>
```

Azért, hogy átláthatóbb forráskódra törekedjünk elhagyhatnánk a menüt befoglaló elemet. Erre a célra ugyanis megfelel a számozatlan lista jelölője és ehhez is hozzáadhatnánk a menü azonosítóját, a CSS szempontjából ez nem jelentene változást. Azonban a feladat gyakorló jellegére való tekintettel, külön div elemt használunk erre a célra.

Tartalmat befoglaló blokk

A fő tartalmat magában foglaló blokk elsődleges feladata a benne elhelyezkedő elemek pozícionálása és egységes megjelenésének biztosítása. Szükséges továbbá az oldal középre helyezésének megvalósításához is. A „tartalom” azonosítót vezettük be hozzá. Ez lényegi tartalmat közvetlenül nem foglal magában csak a „tartalom” és az „oldalsav” blokkok helyezkednek el benne.

Lábléc

A lábléc esetünkben egyszerű, kevésbé fontos információt foglal magában. Más oldalakon esetleg ide kerül a jogi nyilatkozat vagy impresszum, esetleg másodlagos menü, vagy a használhatóság szempontjából előnyös oldal tetejére

görgetési lehetőség. A példafeladat terjedelme nem indokolja más tartalom ide helyezését.

```
    <p>A „internetmarketing” példaoldal XHTM és CSS
felhasználásával készült</p>
```

11.2.5 A fő tartalom

„tartalom”

A példában ez a rész szolgál a lényegi információ közreadására, itt jelennek meg azon tartalmak, ami a weboldal magvát alkotja. Tekintve a lényegességére ez kap a legnagyobb helyet is. A weblapok jellegéből fakadóan ez a rész képes korlátlan mennyiségű adat fogadására, függőlegesen automatikusan igazodni fog a benne foglalt adatok mennyiségéhez.

Az adott oldal címét helyezzük el először. A dokumentum szerkezetében ez meghatározó szereppel bír, ugyanis ez írja le, hogy az adott oldal milyen témát ad közre. Ezen okból ez lesz az elsődleges címsor. Fontos, hogy keresőmarketinges szempontból is ezt válasszuk az elsődleges címsornak, és gondoskodjunk róla, hoyg az adott lapon ne forduljon elő több belőle.

```
    <h1>Online marketing</h1>
```

A szövegek nem helyezkedhetnek el a dokumentumban jelentés nélkül ezért gondoskodjunk róla, hogy a különböző bekezdésekhez használjuk a p jelölőt.

```
    <p>Nem mondunk vele … szinte mindegy is.</p>
    <p>Az online marketing ... weboldal léte.
Sőt.</p>
```

A tartalom igényli, hogy logikailag tovább bontsuk, ugyanis az Online marketing témakörön belül felmerül az „eladási tölcsér” és a „suttogó marketing” fogalma. Ezen részek jelöléséhez a tartalom hierarchiájában a következő jelölés a másodlagos címsor lesz.

```
    <h2>Eladási tölcsér</h2>
```

A szövegben előforduló kiemelt fontosságú részeket az arra szánt megfelelő elembe helyezzük, ez esetben használhatjuk az erős kiemelést.

```
    <p>Egy <strong>eladási tölcsér</strong> felépíté-
se … online tevékenységét.</p>
```

Hasonlóképpen járjunk el a többi előforduló esetnél is.

```
    <p>A <strong>suttogó marketing</strong> általában
… az alkalmazáskor.</p>
```

„oldalsav”

Esetünkben az oldalsáv más oldalakra vonatkozó hivatkozásokat tartalmaz. Ez is egy meghatározott szereppel bíró rész a tartalomban, így használjuk a megfelelő jelölést a dokumentum hierarchiájában. Ez a fő tartalomtól kisebb prioritással bír, így a harmadlagos címsort jelölő elemet alkalmazzuk a kódban.

```
    <h3>külső hivatkozások</h3>
```

Az itt elhelyezkedő hivatkozások egyértelműen lista struktúrába szerveződnek, így a számozatlan listára megfelelő XHTML struktúrát használjuk. Mivel a feladat szempontjából nem lényeges, hogy ezek a hivatkozások tényleges oldalakra mutassanak a href attribútum értékéül egy # jelet használunk. Ebben az esetben a hivatkozás megfelelőképpen viselkedik „élő hivatkozás lesz”.

```
    <ul>
        <li><a   href=„#”>Online   marketing   akadé-
mia</a></li>
        <li><a href=„#”>Marketingblog</a></li>
        <li><a href=„#”>Technikai elemzés</a></li>
        <li><a        href=„#”>Online        marketing
blog</a></li>
        <li><a href=„#”>SEO online oktatás</a></li>
    </ul>
```

11.2.6 Az oldalszerkezet azaz a „layout” kialakítása

Alapbeállítás

Első lépésként stílusszabályokkal kialakítjuk az oldal szerkezetét, amely nagyvonalakban már meghatározza az oldal megjelenését. Ide tartozik a különböző elemek pozícionálása és méretezése. Ez a munkálat régen HTML táblázatokkal történt, de ez már összeegyeztethetetlen a weboldalkészítés irányelveivel. Az XHTML struktúra már nagy segítséget ad abban, hogy mely elemekkel kell ennél az állomásnál foglalkoznunk.

Mindenekelőtt biztosítanunk kell, hogy az oldal a böngészők közti különbség ellenére is ugyanúgy jelenjen meg. Ehhez az úgynevezett CSS reset technikát kell alkalmazni. A feladatban csak egy nagyon egyszerű formáját használjuk ennek, hogy a „dobozok” külső és belső margóit eltüntessük, így magunk adhatjuk meg a konkrét értékeket. Ennél a megoldásnál a speciális „*” kiválasztót használjuk, így a szabály az összes oldalon előforduló elemre vonat-

kozni fog. (A gyakorlatban ennél összetettebb CSS reset technikákat alkalmazunk)

```
* {
    margin: 0px;
    padding: 0px;
}
```

Fejléc

A fejléc elem magasságát és szélességét a grafikai oldaltervből határozhatjuk meg. Beállításához a height és width tulajdonságokat állítjuk be.

```
#fejlec {
   height: 80px;
   width: 713px;
}
```

A megfelelő pozícionálást a margók állításával érhetjük el. Előzőekben a CSS technikáknál már megismertük elemek középre igazításának a módját. Ehhez a két szélső margó értékét auto-ra állítjuk, így a rendelkezésre álló területet a böngésző automatikusan elosztja két oldal közt. Az alapbeállításnál ugyan meghatároztuk az alsó és felső margók értékét 0-ra, viszont a példa teljessége miatt itt is szerepel.

```
#fejlec {
   margin-top: 0px;
   margin-right: auto;
   margin-bottom: 0px;
   margin-left: auto;
   height: 80px;
   width: 713px;
}
```

Menü

A menü méreteit szintén a grafikai tervből állapíthatjuk meg. Ebben az esetben a tervben egy elkülönülő zöld blokk szerepel a menü hátteréül így ennek vesszük a méreteit.

```
#menu {
   height: 50px;
   width: 713px;
}
```

Ezt az elemet szintén a böngészőablak közepére igazítjuk igazodva az oldal többi részéhez. Megoldható lenne, hogy bevezetünk egy új mindent körbefogó blokkot és így globálisan állíthatnánk a szélső margók értékét. Ez gyakori megoldás az oldalszerkezetek kialakításánál, viszont a példában most az egyszerű-

ségre törekszünk. Itt már nem írjuk ki mind a négy oldalra vonatkozó margó értéket, hanem a rövidített írási formát alkalmazzuk. Az első érték az alsó és felső a második a bal és jobb oldalakra vonatkozik.

```
#menu {
   height: 50px;
   width: 713px;
   margin: 0 auto;
}
```

Oldal

Ez a rész szélességében igazodik az eddigiekhez, viszont ebben az esetben nem adunk meg magasság értéket, ugyanis azt szeretnénk, hogy a benne foglalt tartalomhoz ez automatikusan igazodjon. Ebből látszik tehát, hogy a szélességnél és magasságnál, ha nem adunk meg konkrét értéket ez az alapértelmezett viselkedés.

```
#oldal {
   width: 713px;
}
```

A tervezethez igazodva a benne foglalt tartalom nem ér hozzá közvetlenül a terület aljához és tetejéhez. Ezt a hatást a belső margók beállításával érhetjük el, amit egységesen 20 pixelnyire állítunk. Továbbá biztosítjuk a blokk középre igazítását a margók beállításával.

```
#oldal {
   width: 713px;
   padding-top: 20px;
   padding-bottom: 20px;
   margin: 0 auto;
}
```

Tartalom

A tartalom és az oldalsáv egymás mellé helyezéséhez, szükséges, hogy először adjuk meg mindkettő szélességét, ugyanis addig nem fognak beférni egymás mellé az oldal blokkon belül. Ezt az értéket nem érdemes pixelre pontosan kiszámolni, ugyanis előfordulhat, hogy régebbi böngészők kevésnek találják a fennmaradó területet, és ilyenkor a két blokk egymás alá csúszik.

```
#tartalom {
   width: 410px;
}
```

A továbbiakban arról gondoskodunk, hogy a tartalom ténylegesen az oldalsáv bal oldalára kerüljön ezt az elem lebegtetésével vagy úsztatásával határozzuk meg. Az elem tehát a bal oldalon lebeg.

```
#tartalom {
   width: 410px;
   float: left;
}
```

Végül biztosítjuk, hogy legyen távolság az oldal széle és az elem között, erre a bal margónak adunk 25 pixeles értéket.

```
#tartalom {
   float: left;
   width: 410px;
   margin-left: 25px;
}
```

Oldalsáv

Az oldalsáv elhelyezése nagyban hasonlít a tartalméhoz azzal a különbséggel, hogy a jobb oldalra vonatkozó beállításokat adjuk meg. Tehát először szélesség, majd lebegés jobbra, végül pedig a jobb margó beállítása.

```
#oldalsav {
   width: 210px;
   float: right;
   margin-right: 25px;
}
```

Lábléc

A lábléc az oldal talán legegyszerűbb eleme. Mivel itt csak egy rövid szöveg van, elegendő azt középre igazítani, így nem kell ezt a blokkot méretezni és margókkal pozícionálni. A szöveg megfelelően esztétikus elhelyezéséhez azonban a belső margók használatával megadjuk a felső és alsó helyközt.

```
#lablec {
   padding-top: 20px;
   padding-bottom: 20px;
}
```

Lebegési probléma kiküszöbölése

Sajnálatos módon az eddigi beállítások még nem elegendőek, ugyanis a tartalom és az oldalsáv lebegtetése miatt azok kicsúsznak az oldal blokkból, és ezzel szétcsúszik az oldalunk szerkezete. Ezen probléma feloldására több tech-

nika is létezik. Talán a legelegánsabb, ha az oldal, azaz a befoglaló elem tulajdonságait állítjuk be, hogy az ne engedje kicsúsznia két elemet a területéről.

```
#oldal {
   width: 713px;
   padding-top: 20px;
   padding-bottom: 20px;
   margin: 0 auto;
   height: 100%;
   overflow: hidden;
}
```

11.2.7 Háttérképek és színek

A szerkezet kialakításához jelentősen hozzájárul a blokkok háttereinek megadása, ugyanis ezzel különülnek el azok egymástól. A grafikai tervet weblapfejlesztési irányelvek alapján érdemes elkészíteni, így egyszerűbb lesz a különböző elemek előállítása.

Az oldal háttere

Az oldal háttere mindaz a terület ami a a weboldal két szélén és alatt helyezkedik el. Többnyire homogén fekete egyszerű háttér, ferdén csíkozott felső sávval. A felső sáv mérete igazodik a böngésző ablak szélességéhez, így mindegy milyen felbontásban tekintik meg az oldalt, a sávnak sosem lesz vége. Ezt a hatást úgy érjük el, hogy a sávból kimetszünk egy darabot, amit ha vízszintesen ismétlünk egész hatást kelt. Figyelnünk kell ennek a felső pozícióra történő beállítására is.

49. ábra: fejlec.png

```
body {
   background-color: #000000;
   background-image: url(img/fejlec.png);
   background-repeat: repeat-x;
   background-position: top;
}
```

Menü háttere

Ennek a blokknak a hátterét egyben emeltük ki a grafikai tervből, így azt egyszerűen beilleszthetjük. Háttérszínként a háttérképnek megfelelő színt választunk arra az esetre, ha valamilyen okból kifolyólag a kép nem töltődne be, úgy is olvashatók legyenek a menüfeliratok. Másrészt ha az oldal lassan töltődik be a menü háttere már hasonló megjelenést kapjon.

50. ábra: menu.png

```
#menu {
   height: 50px;
   width: 713px;
   margin: 0 auto;
   background-color: #a7e218;
   background-image: url(img/menu.png);
   background-repeat: no-repeat;
}
```

Tartalom háttere

Ennek a blokknak a hátterét fölösleges képként beilleszteni, ugyanis egy egyszerű szürke háttér. Egyetlen momentum, amit meg kell oldani, az az oldal alján található sarkok lekerekítése. Ezt ugyan a CSS3 alkalmazásával egyszerűen megoldhatjuk, de továbbra is maradunk a CSS2 megoldásainál. A grafikai tervben egyszerűen lemetsszük a tartalmi rész legalsó sávját olyan magasságban ameddig a lekerekítés kanyarulata tart. Ezt a képet alsó háttérként beillesztve elérjük a kívánt hatást.

```
#oldal {
   width: 713px;
   padding-top: 20px;
   padding-bottom: 20px;
```

```
    margin: 0 auto;
    height: 100%;
    overflow: hidden;
    background-color: #4c4c4c;
    background-image: url(img/lablec.png);
    background-repeat: no-repeat;
    background-position: left bottom;
}
```

51. ábra: lablec.png

11.2.8 A menü felépítése

Igazodva a szemantikai irányelvekhez a menüt számozatlan listaként definiáltuk az XHTML kódban. Ennek az alapértelmezett megjelenítési formája egymás alatti felsorolás. Esetünkben azonban vízszintes elrendezés szükséges. Ennek eléréséhez először meg kell szüntetnünk a lista formázási jeleit. Valamint a korrekt megjelenítéshez beállítjuk a bal oldali margót is. A felső belső margó segítségével pozícionáljuk a menüpontok függőleges helyzetét.

A szabálynál kombináljuk a kiválasztókat, ugyanis több lista helyezkedik el az oldalon de mi csak a menu azonosítóval ellátott blokkban lévő számozatlan listára akarunk hivatkozni. Ezért a #menu azonosító kiválasztó után írjuk a számozatlan lista kiválasztóját.

```
#menu ul {
   list-style-type: none;
   margin-left: 40px;
   padding-top: 14px;

}
```

A menüpontok egyenlőre még mindig egymás alatt helyezkednek el. Ez a viselkedés azért történik, mert a listaelemek alapértelmezésben blokkszintű elemek, így ekképpen is viselkednek. A CSS lehetőséget ad ezen viselkedés megváltoztatására a display tulajdonság használatával. Ha ennek inline értéket adunk onnantól az elem sorközi viselkedéssel rendelkezik.

A menüelemek közti helyközökről az oldalsó belső margók beállításaival gondoskodunk.

```
#menu li {
   display: inline;
   padding-right: 10px;
```

```
   padding-left: 10px;
}
```

11.2.9 Külső hivatkozások

Hasonlóképpen a menühöz először eltűntetjük a lista elemek jelölését.

```
#oldalsav ul {
   list-style-type: none;
}
```

A listaelemek tagolt megjelenítéséhez egyéni sormagasságot adunk meg, továbbá itt beállítjuk az általánostól eltérő betűméretet is amit 10 pixelesre állítunk.

```
#oldalsav li {
   line-height: 26px;
   font-size: 10px;
```

11.2.10 Szövegek megjelenése

Globális szövegstílus

Az általános szövegstílus beállításainál figyelembe vesszük, hogy ez az elemek közt öröklődik, így ha a mindenkori legmagasabb szinten álló szülő elemnek állítjuk be a tulajdonságait, azt az abban foglaltak örökölni fogják néhány kivétellel. A betűtípus a betűméret és a betűszín beállításait adjuk itt meg.

```
body {
   background-color: #000000;
   background-image: url(img/fejlec.png);
   background-repeat: repeat-x;
   background-position: top;
   font-family: Verdana, Arial, Helvetica, sans-
serif;
   font-size: 14px;
   color: #FFFFFF;
}
```

11.2.11 A fejléc

A fejlécben két bekezdés található, az egyik az oldal címét a másik az üzenetet tartalmazza. Először a cím pozícióját állítjuk be, de mivel ezek blokk szintű elemek az egymás mellé helyezésüket lebegtetéssel tudjuk elérni. A további pozícionálást a belső margók értékeinek beállításaival érjük el.

```
.cim {
   float: left;
   padding-top: 20px;
```

```
}
```

A betűtípus a betűszín és a betűméret megadása a következő lépés.

```
.cim {
   float: left;
   padding-top: 20px;
   font-family: Georgia, „Times New Roman”, Times,
serif;
   font-size: 40px;
   color: #4c4c4c;
}
```

Az üzenetnél is hasonlóképpen járunk el. Balra lebegtetjük és a megfelelő pozícionálást a bal oldali és a felső belső margó beállításaival kalibráljuk.

```
.uzenet {
   float: left;
   padding-top: 40px;
   padding-left: 18px;
}
```

A szövegstílus meghatározásához beállítjuk a betűtípust, a betűméretet és betűszínt is. A betűtípus beállítására csak ennél a két elemnél kell kitérnünk ugyanis ez eltér a globális beállításoktól.

```
.uzenet {
   float: left;
   padding-top: 40px;
   padding-left: 18px;
   color: #8bd80e;
   font-family: Georgia, „Times New Roman”, Times,
serif;
   font-size: 18px;
}
```

Bekezdések

A lényegi tartalom jelentős részét bekezdésekbe rendezett szövegek teszik ki. Ezek esztétikus tagolt megjelenítéséhez a felső és alsó margóját 12 pixel nagyságúra állítjuk.

```
#tartalom p {
    margin: 12px 0;
}
```

A sormagasság és a betűméret is az általános képet szabályozza. Ezeken felül a szöveg sorkizárt igazítását állítjuk be, hogy rendezettebbnek tűnjön a tartalom.

```
#tartalom p {
    margin: 12px 0;
```

```
    line-height: 22px;
    font-size: 12px;
    text-align: justify;
}
```

Címsorok

Háromszintű címsort használunk az oldalon. Ezek megjelenését a margók és betűméretek beállításaival szabályozzuk. Továbbá az egységesebb megjelenés érdekében megszüntetjük az elsődleges és másodlagos címsor félkövér megjelenítését.

```
h1 {
   margin: 18px 0;
   font-weight: normal;
}
h2 {
   margin: 12px 0;
   font-size: 20px;
   font-weight: normal;
}
h3 {
   margin: 12px 0;
}
```

Hivatkozások

Hiába állítjuk be a globális betűszínt, a hivatkozások másképpen viselkednek. Az általános hivatkozásszín meghatározásához egyszerűen a hivatkozás kijelölőjéhez állítjuk be az alapértelmezett színt.

```
a {
   color: #8bd80e;
}
```

A menüelemek hivatkozásai eltérnek az oldal többi részén használtaktól. Elsősorban betűszínük és betűméretük különbözik.

```
#menu a {
   color: #000000;
   font-size: 12px;
}
```

Másik jellemző különbség, hogy ezek a hivatkozások nincsenek aláhúzva, ezt a text-decoration tulajdonsággal szabályozhatjuk.

```
#menu a {
   color: #000000;
   font-size: 12px;
   text-decoration: none;
```

```
}
```

11.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

11.3.1 Összefoglalás

A fejezet segítségével betekintést nyerhetett az olvasó a weboldal-készítési folyamat egészébe.

11.3.2 Önellenőrző kérdések

1. Milyen előkészületi munkákat kellett megtennünk?
2. Mire törekedtünk a struktúra megtervezésekor?
3. Mire törekedtünk a címsorok meghatározásakor?
4. Miért div elemeket használtunk az oldal tartalmának megírásakor?
5. Mire kell ügyelnünk az oldal elemeinek lebegtetésekor?

12. ÖSSZEFOGLALÁS

12.1 TARTALMI ÖSSZEFOGLALÁS

A tananyag a statikus weboldalak elkészítéséhez szükséges tudnivalókról szól. *A Weblapfejlesztés eszközei* című, első témakörben a tanuló átfogó képet kap a web kialakulásának okáról és körülményeiről, a weblapok letöltéséről, működéséről, felépítéséről, a fájlszerkezettel kapcsolatos alapszabályokról és a weblapok elkészítésének eszközeiről. A *Webes szabványok és a W3C* című, második témakörben megvilágításra kerül az, hogy miért érdemes szabványokat használni a weblapszerkesztés során, továbbá a böngészők fejlődéséről és a weblapszerkesztéshez kapcsolódó szabványok kialakulásáról lehet olvasni. Az említett két témakört három olyan fejezet követ, amely a HTML leírónyelv jelölőivel, attribútumaival és szabályaival foglalkozik. Újabb három fejezetben a CSS stílusfájlok felépítése, szabályai és HTML-hez való kapcsolódása kerül bemutatásra. Az utolsó előtti témakör a weblapok működését segítő JavaScript szkriptnyelvvel foglalkozik. Végül egy teljes fejezet egy konkrét példát bemutatva szól arról, hogy hogyan érdemes egy weboldalt felépíteni a tananyag korábbi fejezeteiben leírt eszközöket használva.

12.2 ZÁRÁS

Remélhetőleg a tananyag feldolgozását követően a tanulók egy **globális látásmóddal fognak rendelkezni a weblapfejlesztéssel kapcsolatosan**. Ismerni fogják a weblapszerkesztéshez szükséges szoftverek széles tárházát és azok **webes vonatkozású alkalmazási területeit**. Ismerni fogják, és jól el tudják majd különíteni a weblapfejlesztés egyes folyamatait, ezzel párhuzamosan felismerik és körvonalazódik bennük a weblapfejlesztő teamekben dolgozó munkatársak szerepei és a hozzájuk tartozó egyes feladatkörök.

Azoknak a szakembernek, akik egyedül készítenek weboldalakat, minden egyes webes munkakört (**tervező, grafikus, HTML szerkesztő, programozó**) egyedül kell ellátniuk és minden terület esetében magas fokú tudással kell rendelkezniük ahhoz, hogy fel tudják venni a versenyt a piacon lévő weblapfejlesztő cégekkel.

Egy személy jellemzően azért a weblapfejlesztés **egyetlen területének a tudója és szakembere**, de ahhoz, hogy egy weblapfejlesztő team csapatmunkája eredményes legyen elengedhetetlen az, hogy a fejlesztésben résztvevő személyeknek **legyen rálátásuk a többi szakember tevékenységeire** és a weblapok

készítésének a teljes folyamatára annak érdekében, hogy képesek legyenek jól összedolgozni a többi terület szakembereivel.

13. KIEGÉSZÍTÉSEK

13.1 IRODALOMJEGYZÉK

TERRY FELKE-MORRIS: Web Development and Design Foundations with XHTML, Pearson Education, 2011.

TERRY FELKE-MORRIS: Web Development & Design Foundation with HTML5 (6th Edition). USA, Pearson Education, Inc. 2013. ISBN-13: 978-0-13-278339-2.

ETHAN MARCOTTE - JEFFREY ZELDMAN: Szabványkövető webtervezés. Kiskapu Kiadó, Bp., 2011.

VIRGINIA DEBOLT: HTML és CSS - Webszerkesztés stílusosan. Kiskapu Kiadó, Bp., 2005.

SIKOS LÁSZLÓ: XHTM - A HTML megújulása XML alapokon. BBS-INFO Kiadó, Bp., 2004.

SIKOS LÁSZLÓ: Stíluslapok a weben - CSS kézikönyv. BBS-INFO Kiadó, Bp., 2005.

MICHAEL MONCUR: Tanuljuk meg a JavaScript használatát 24 óra alatt. Kiskapu Kiadó, Bp., 2006.

MARK PILGRIM: HTML 5 - Az új szabvány, Ugorjunk fejest a webfejlesztés jövőjébe! Kiskapu Kiadó, Bp., 2011.

13.1.1 Hivatkozások

Elektronikus dokumentumok / források

http://www.w3schools.com

http://www.tutorial.hu/webszerkesztes/html5-css3-osszefoglalo/html5-css3-osszefoglalo-v10.pdf

http://weblabor.hu/cikkek/cssalapjai1