

WEBTERVEZÉS – GYAKORLATI JEGYZET

1. gyakorlat

Követelmények, HTML alapok I.

Készítették:

Cservenák Bence
Farkas Anikó
Savanya Sándor

FIGYELEM!

A jegyzet folyamatosan készül, így előfordulhatnak benne apróbb hibák, hiányosságok, elírások. Ha valaki esetleg ilyet találna, kérem, írjon a Cservenak.Bence@stud.u-szeged.hu címre, hogy mihamarabb javíthassuk.

1. Közérdekű információk

- A gyakorlat tematikája, illetve követelményei elérhetők a gyakorlat CooSpace színterén
- Néhány fontosabb segédanyag
 - Vancsics Béla weboldala: www.inf.u-szeged.hu/~vancsics/webterv
 - Abonyi-Tóth Andor: A weblapkészítés technikája (HTML5, CSS3) és ergonómiaja
 - Angol nyelvű referencia: <https://www.w3schools.com/>

2. Áttekintés: A HTML, CSS és PHP nyelvek

	HTML	CSS	PHP
Micsoda?	jelölőnyelv	stílusleíró nyelv	szerveroldali szkriptnyelv
Szerepe	weboldalak tartalmának kialakítása, a tartalom strukturálása	a tartalom formázása	úrlapfeldolgozás, adatbáziskezelés, menetkövetés, fájlkezelés, OOP, ...

3. A HTML alapjai (1. rész)

A. Bevezetés

A **HTML** (HyperText Markup Language) segítségével mondjuk meg, hogy mi az, amit egy weboldalon látni szeretnénk (pl. szövegek, képek, hivatkozások, táblázatok, multimédia, ...).

A weboldal tartalmát lehetőségünk van strukturálni is, különböző szakaszok, tartalmi egységek kialakításával (pl. fejléc, menüsor, cím, indexkép, fő tartalom, lábléc, ...).

A HTML kód írása roppant egyszerű:

- A kódot tetszőleges szerkesztőprogramba írjuk (pl. Gedit, Notepad++, Webstorm, ...)
- Az állományt **.html** kiterjesztéssel elmentjük
- Végül a fájlt megnyitjuk egy tetszőleges böngészővel, és már láthatjuk is az eredményt

Megjegyzés: A HTML kódot más-más böngészők (és böngészőverziók) eltérő módon jeleníthetik meg

B. Szabványok

A weboldalak készítésére vonatkozóan a **W3C** nevű szervezet ad ki előírásokat, szabványokat.

Fontosabb szabványok:

- XHTML 1.0 Strict (2000)
 - a korábbi HTML 4 szabvány “újraírása” XML alapokon
 - szigorú szabályok
 - HTML dokumentumok egységessé tétele, egyszerűbb feldolgozás programmal
- HTML5 (2014)
 - kevésbé szigorú, mint az XHTML 1.0 Strict
 - programmal nehezebb feldolgozni
 - számos újdonság az elődjeihez képest (pl. szemantikus elemek, multimédia, grafikus elemek, új űrlapmező típusok)

A gyakorlaton a **HTML5** és **CSS3** szabványokkal fogunk foglalkozni. A webfejlesztés során érdemes törekednünk a választott szabvány előírásainak betartására!

A weboldalunk szabványosságát legegyszerűbben böngésző plug-inek vagy az alábbi online validator szoftverek segítségével ellenőrizhetjük:

- HTML validator: <https://validator.w3.org/>
- CSS validator: <https://jigsaw.w3.org/css-validator/>

Tipp: A beadandó projekt szabványosságát ezekkel egyszerűen tudjátok ellenőrizni

C. Alapfogalmak

- **Tagek:** < és > közé írt, speciális jelentésű HTML objektumok
 - a legtöbb tag **páros tag** – egy nyitótagból és egy zárótagból áll
 - pl. <p>Ide jön valami szöveg</p>
 - vannak viszont **páratlan (üres) tagok** is – csak nyitótagból állnak
 - pl.

 - opcionálisan a tag végére perjelet is lehetünk (pl.
)
 - MEGJEGYZÉS: a jegyzetben minden üres tag esetén kiírjuk a perjelet
 - tagok egymásba ágyazásakor a tagokat belülről kifelé haladva zárjuk le
 - pl. <div><p><a></p></div>
- **Attribútumok:** tagokhoz tartozó speciális tulajdonságok
 - speciális értékek tartoznak hozzájuk
 - az attribútum-érték párok a nyitótag neve után írjuk, szóközzel elválasztva
 - pl. Google
 - pl.

D. A HTML dokumentum felépítése

Minden HTML dokumentum megírása az alábbi kóddal kezdődik. Nézzük meg, hogy mi mit jelent!

```
<!DOCTYPE html>
<html>
<head>

</head>
<body>

</body>
</html>
```

- `<!DOCTYPE html>`: jelezük, hogy a HTML5 szabvány szerint írtuk a kódot
- `<html>...</html>`: ez írja le a weboldalt
- `<head>...</head>`: **fejrész** – az oldalon nem megjelenő tartalom (metainformációk) fájlcsatolások, CSS- és szkriptbeágyazások
- `<body>...</body>`: **törzsrész** – a megjelenő tartalom, strukturális elemek, szkriptek

E. Kommentek

A HTML fájlokba lehetőségünk van kommenteket (megjegyzések) írni az alábbi szintaxissal:

```
<!-- Ez egy komment a HTML kódban -->
```

F. A fejrész (`head`) fontosabb tagjai

- `<title>...</title>`: a böngésző fülén megjelenő cím
- `<meta charset="UTF-8"/>`: UTF-8 karakterkódolás beállítása
- `<meta name="description" content="Józsi blogoldala"/>`: a weboldal leírása
- `<meta name="author" content="Józsi"/>`: a weboldal szerzője
- `<meta name="keywords" content="webtervezés, webterv, HTML, CSS, PHP"/>`: kulcsszavak megadása (célja a találati esélyeink javítása)
- `<link rel="stylesheet" type="text/css" href="foo.css"/>`: külső CSS fájl beágyazása
- `<link rel="icon" type="image/png" href="icon.png"/>`: böngészőfülön megjelenő ikon
- `<style>...</style>`: CSS kód beszúrása
- `<script>...</script>`: szkript (pl. JavaScript) beszúrása

G. A törzsrész (**body**) fontosabb tagjei

a. Bekezdések, címsorok

- `<p>...</p>`: bekezdés
- `<h1>...</h1>`: 1. szintű címsor (legfontosabb)
- `<h2>...</h2>`: 2. szintű címsor
- `<h3>...</h3>`: 3. szintű címsor
- `<h4>...</h4>`: 4. szintű címsor
- `<h5>...</h5>`: 5. szintű címsor
- `<h6>...</h6>`: 6. szintű címsor (legkevésbé fontos)

```
<!DOCTYPE html>
<html>
<head>
    <title>Bekezdések, címsorok</title>
    <meta charset="UTF-8">
</head>
<body>
    <h1>1. szintű címsor</h1>
    <h2>2. szintű címsor</h2>
    <h3>3. szintű címsor</h3>
    <h4>4. szintű címsor</h4>
    <h5>5. szintű címsor</h5>
    <h6>6. szintű címsor</h6>
    <p>Ez egy bekezdés...</p>
</body>
</html>
```

1. szintű címsor

2. szintű címsor

3. szintű címsor

4. szintű címsor

5. szintű címsor

6. szintű címsor

Ez egy bekezdés...

b. Karakterentitások

Mivel a HTML szintaxisából adódóan bizonyos karakterek (pl. a < és > karakterek) le vannak foglalva speciális célokra, ezért ezek helyett célszerű ún. **karakterentitásokat** használni.

Néhány gyakori karakterentitás (teljes lista [itt](#)):

Karakterentitás	Megjelenítés
<code>&ampnbsp</code>	(szóköz)
<code>&lt;</code>	<
<code>&gt;</code>	>
<code>&amp;</code>	&
<code>&quot;</code>	"
<code>&copy;</code>	©

c. Képek

Képet az `` páratlan taggel tudunk beszúrni a weboldalunkra.

Fontosabb attribútumai:

- `src`: a kép elérési útvonala (kötelező megadni)
- `alt`: helyettesítő szöveg, ha a kép nem jelenik meg (kötelező megadni)
- `title`: a kép címe, ami akkor jelenik meg, ha a kurzor a kép fölött áll
- `width`: szélesség (pixelben)
- `height`: magasság (pixelben)

Megjegyzés: Ha a `width` és `height` attribútumok közül csak az egyiket adjuk meg, akkor a másik a kép eredeti méretarányaiból számolódik. Például ha egy 400x300-as (400 px széles, 300 px magas) kép szélességét 800 px-re állítjuk (megduplázzuk), akkor a magassága automatikusan 600 px lesz.

Megjegyzés: A képek szélességét és magasságát célszerűbb CSS-ben megadni.

Példa: A HTML fájllal egy mappában van egy `img` mappa, amiben található a `tree.jpg` képfájl. Szűrjuk be a képet a weboldalunkra!

```

```



d. Az `id` és `class` attribútumok

Minden HTML objektumnak adható egyedi azonosító, és minden HTML objektum csoporthoz köthető osztályokba. Ezek CSS-ben lesznek majd hasznosak, amikor egy-egy objektumra vagy objektumok csoportjára szeretnénk hivatkozni.

Az **egyedi azonosító** (`id`) értéke a dokumentumon belül egyedi kell, hogy legyen. Emellett az attribútum értéke nem tartalmazhat szóközöt.

Az **osztály** (`class`) értéke nem kell, hogy egyedi legyen (sőt általában nem az). Értékéül szóközzel elválasztva megadható több osztály neve is, amihez az adott elem tartozik.

```
<!-- Egy "asd" egyedi azonosítóval rendelkező bekezdés -->
<p id="asd">Ide jön valami szöveg...</p>

<!-- A "foo" és "bar" osztályokhoz tartozó bekezdés -->
<p class="foo bar">Ide jön valami szöveg...</p>
```

e. Hivatkozások (Linkek)

Hivatkozásokat az `<a>` és `` tagpár segítségével szúrhatunk be a weboldalunkra.

Fontosabb attribútumok:

- **`href`**: cél, ahova a hivatkozásra kattintva jutunk
 - lehet egy másik weboldal
 - pl. `Google`
 - lehet egy másik dokumentum
 - pl. `ZH pontszámok`
 - lehet lapon belüli hivatkozás
 - pl. `Vissza a tetejére`
 - ez az `id="top"` attribútummal rendelkező objektumra mutat
- **`target`**: hol legyen megnyitva a hivatkozott cél
 - ugyanott, ahol rákattintottunk (alapértelmezett): `target="_self"`
 - új lapon vagy ablakban: `target="_blank"`
 - az ablak teljes méretében: `target="_top"`
- **`title`**: hivatkozás címe (ha a kurzor a hivatkozás fölött áll, akkor jelenik meg)

```
<!-- Egy hivatkozás, ami egy képet nyit meg egy új lapon -->
<a href="cukicicak.jpg" target="_blank">Stressz ellen</a>

<!-- Egy hivatkozás, ami egy másik weboldalt nyit meg ugyanott, ahol a linkre
     kattintottunk, és az oldal id="Behavior" attribútumú elemére mutat -->
<a href="https://en.wikipedia.org/wiki/Goat#Behavior">Kecske</a>
```

f. Szakaszok

HTML-ben lehetőségünk van a weboldal tartalmának strukturálására. Ezt szakaszok, tartalmi egységek kialakításával tudjuk megtenni a weboldalon belül.

- `<div>...</div>`: blokkszintű szakasz
 - minden sorban kezdődik, és kihasználja a rendelkezésre álló szélességet
- `...`: sorszintű szakasz
 - nem kezdődik új sorban, szélessége csak akkora, mint amekkora helyre szüksége van

A `span tag` és `<div>div tag</div>` szakaszok kialakítására alkalmas.

A `span tag` és
`div tag`
szakaszok kialakítására alkalmas.

WEBTERVEZÉS – GYAKORLATI JEGYZET

2. gyakorlat

HTML alapok II., a CSS alapjai

Készítették:

Cservesnák Bence
Farkas Anikó
Savanya Sándor

FIGYELEM!

A jegyzet folyamatosan készül, így előfordulhatnak benne apróbb hibák, hiányosságok, elírások. Ha valaki esetleg ilyet találna, kérem, írjon a Cservesnak.Bence@stud.u-szeged.hu címre, hogy mihamarabb javíthassuk.

1. A HTML alapjai (2. rész)

Folytassuk a HTML dokumentum törzsrészében (`body`) használható tagokkal való ismerkedést!

A. Néhány fontosabb szövegelem

- `<p>...</p>`: bekezdés
 - előtte és utána térköz található
- `<pre>...</pre>`: előre formázott szöveg
 - alakhű megjelenítést biztosít
 - akkor használjuk, ha a szöveg jellegéből szemantikusan következik a tartalom szóközökkel, térközökkel való formázása
- `
`: sortörés (páratlan tag)
- `<hr/>`: elválasztó vonal (páratlan tag)

```
<h1>Piramis</h1>
<hr/>
<pre>
    0
    0 0
    0 0 0
    0 0 0 0
</pre>
A piramis egy négyzet alapú <br/> gúla
alakú építmény.
```

Piramis

```
    0
    0 0
    0 0 0
    0 0 0 0
```

A piramis egy négyzet alapú
gúla alakú építmény.

- `...`: tartalom hangsúlyozása (szemantikus jelentés, általában dőlt)
- `...`: erős kiemelés (szemantikus jelentés, általában félkövér)

```
<em>Ricsi</em> ma vizsgázik prog2-ből. <br/>      <!-- Ki vizsgázik? -->
Ricsi <em>ma</em> vizsgázik prog2-ből. <br/>      <!-- Mikor vizsgázik? -->
Ricsi ma vizsgázik <em>prog2-ból</em>. <br/>      <!-- Miből vizsgázik? -->

<p><strong>Figyelem!</strong> A ketrecbe benyúlni veszélyes!</p>
```

Ricsi ma vizsgázik prog2-ből.
Ricsi ma vizsgázik prog2-ből.
Ricsi ma vizsgázik prog2-ből.

Figyelem! A ketrecbe benyúlni veszélyes!

Megjegyzés: A HTML tagokat **ne** használjuk formázási célokra! Ha egy szövegrészt félkövérré vagy dőlötté szeretnénk tenni, használunk CSS-t!

- `_{...}`: alsó index
- `^{...}`: felső index
- `<q>...</q>`: idézet (az idézőjeleket automatikusan kiteszi)
- `<blockquote>...</blockquote>`: idézetblokk
- `<code>...</code>`: kódrészlet
- `<mark>...</mark>`: vizuális szövegkiemelés

```
Alsó index: x<sub>2</sub>, felső index: x<sup>2</sup> <br/>
<q>A Kalkulus még könnyű tárgynak számít</q>
<blockquote>Never gonna give you up <br/>
Never gonna let you down <br/>
Never gonna run around <br/>
And desert you</blockquote>

<code>int main(int argc, char **argv)</code> <br>
rizsa rizsa <mark>lényeg</mark> rizsa
```

Alsó index: x_2 , felső index: x^2
"A Kalkulus még könnyű tárgynak számít"

Never gonna give you up
Never gonna let you down
Never gonna run around
And desert you

```
int main(int argc, char **argv)
rizsa rizsa lényeg rizsa
```

B. Az **iframe** tag

- `<iframe>...</iframe>`: inline frame, más weboldalt lehet megnyitni benne
 - **src**: a megnyitni kívánt weboldal URL-je
 - **width**: iframe szélessége (érdemesebb CSS-ben megadni)
 - **height**: iframe magassága (érdemesebb CSS-ben megadni)

```
<iframe src="http://ttik.hu" width="600" height="600"></iframe>
```

C. Listák, felsorolások

- `...`: rendezetlen (számozatlan) lista
 - `...`: listaelém
- `...`: rendezett (számozott) lista
 - `...`: listaelém
 - fontosabb attribútumok:
 - `reversed`: fordított számozás
 - `start`: számozás kezdősorszáma
 - `type`: felsorolásjel típusa
 - lehetséges értékek: 1 (arab szám), i (“kisbetűs” római szám), I (“nagybetűs” római szám), a (kisbetű), A (nagybetű)

```

<!-- Rendezetlen lista -->
<ul>
    <li>Kenyér</li>
    <li>Tej</li>
    <li>Felvágott</li>
</ul>

<!-- Rendezett lista -->
<ol>
    <li>Nyisd ki a szemed!</li>
    <li>Nyomd ki az ébresztőt!</li>
    <li>Aludj vissza!</li>
</ol>

```

- Kenyér
- Tej
- Felvágott

1. Nyisd ki a szemed!
2. Nyomd ki az ébresztőt!
3. Aludj vissza!

- a fentiekből készíthetünk többszintű listát is

```

<ol>
    <li>Rendezett listaelém</li>
    <li>
        Rendezett listaelém
        <ul>
            <li>Rendezetlen listaelém</li>
            <li>Rendezetlen listaelém</li>
        </ul>
    </li>
    <li>Rendezett listaelém</li>
</ol>

```

1. Rendezett listaelém
2. Rendezett listaelém
 - Rendezetlen listaelém
 - Rendezetlen listaelém
3. Rendezett listaelém

D. Táblázatok

- <table>...</table>: táblázat beszúrása
 - <tr>...</tr>: sor
 - <th>...</th>: fejléc cella
 - <td>...</td>: oszlopcella
 - **colspan**: oszlopok összevonása
 - **rowspan**: sorok összevonása
 - **headers**: cella társítása adott id-jű fejléc cellá(k)hoz
 - <caption>...</caption>: táblázat címe
 - közvetlenül a <table> tag után írjuk
 - <thead>...</thead>: táblázat fejlécsorainak csoportja (elhagyható)
 - <tbody>...</tbody>: táblázat adatsorainak csoportja (elhagyható)
 - <tfoot>...</tfoot>: táblázat láblécsorainak csoportja (elhagyható)
 - <colgroup>...</colgroup>: oszlopok csoportosítása
 - közvetlenül a <caption> után (ha van) és a <thead> előtt (ha van)
 - a <col /> páratlan tag egy oszlopot jelöl a csoportosításban
 - a <col **span="n"** /> n darab oszlopot jelöl a csoportosításban

```
<table>
  <caption>Valami klassz cím</caption>
  <colgroup>
    <col style="background-color: lightblue;" />      <!-- világoskék -->
    <col style="background-color: lightgray;" span="2" /> <!-- világosszürke -->
  </colgroup>
  <thead>
    <tr>
      <th id="a">Fejléc</th>
      <th id="b" colspan="2">Oszlopösszevonás</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th id="c" rowspan="2">Sorösszevonás</th>
      <td headers="b c">adat</td>
      <td>még több adat</td>
    </tr>
    <tr>
      <td>ez is adat</td>
      <td>meg ez is</td>
    </tr>
  </tbody>
</table>
```

Valami klassz cím

Fejléc	Oszlopösszevonás	
Sorösszevonás	adat	még több adat
	ez is adat	meg ez is

2. A CSS alapjai

- A CSS (Cascading Style Sheets) egy stílusleíró nyelv
 - a weboldal tartalmának formázására, elrendezésére használjuk
 - a CSS fájlok kiterjesztése: `.css`
 - utasítások szintaxisa: kijelölő (amit formázok) + meghatározásblokk (ahogyan formázok)
 - a meghatározásblokk tulajdonság-érték párokat tartalmaz
 - például: `h1 { color: blue; font-size: 24px; font-weight: bold; }`
 - kommentek szintaxisa: `/* ez egy komment a CSS kódban */`

A. CSS kód elhelyezése

- **Beágyazott (inline) CSS:** HTML objektumok `style` attribútumaként adjuk meg
 - például: `<p style="color: red;">Ez egy bekezdés.</p>`
 - az adott objektumra lesz érvényes
 - nem túl hatékony – ha lehet, ne használjuk
- **Lapon belüli (internal) CSS:** a `head`-ben adjuk meg `<style>` és `</style>` tagok között
 - például: `<style> p { color: red; } </style>`
 - az aktuális dokumentumra lesz érvényes
- **Külső (external) CSS:** a `head`-ben adjuk meg a `<link>` páratlan tag segítségével
 - például: `<link rel="stylesheet" type="text/css" href="foo.css"/>`
 - minden csatoló dokumentumra érvényes lesz a `foo.css` fájlban lévő CSS kód
 - ha több HTML oldalt szeretnénk hasonlóan formázni, ezt érdemes használni

B. Alapvető kijelölők

- `p`: az összes `<p>...</p>` HTML objektumot kijelöli (element selector)
- `.foo`: az összes `class="foo"` attribútummal rendelkező objektumot kijelöli (class selector)
- `#foo`: az `id="foo"` attribútummal rendelkező objektumot jelöli ki (id selector)
- `*`: minden HTML objektumra vonatkozik (universal selector)

C. Értékek, mértékegységek

- Hosszúság megadása
 - abszolút hosszúság: `mm` (milliméter), `cm` (centiméret), `in` (inch), `pt` (pont), `pc` (pica)
 - relatív hosszúság: `%` (más értékekből számolódik), `px` (pixel), `em` (betűmérethez viszonyít), `rem` (gyökérelem betűméretéhez viszonyít) `vw` (ablakszélesség 1%-ához viszonyít), `vh` (ablakmagasság 1%-ához viszonyít)
- Szögek megadása: `deg` (fok), `rad` (radián), `grad` (gradián)

- Színek megadása: névvel (pl. white), decimális RGB-kóddal (pl. `rgb(255, 255, 255)`), hexadecimális RGB-kóddal (pl. #FFFFFF vagy röviden #FFF)

D. Objektumok méretezése

- `width`: objektum szélessége
 - megadható: px, %, auto (a böngésző számolja)
 - legkisebb szélesség: `min-width`, legnagyobb szélesség: `max-width`
- `height`: objektum magassága
 - megadható: px, %, auto (a böngésző számolja)
 - legkisebb magasság: `min-height`, legnagyobb magasság: `max-height`

E. Szövegek formázása

- `font-family`: betűtípus
 - a több szóból álló betűtípusok (pl. Courier New) nevét ''-ok között adjuk meg
 - több betűtípus felsorolható (vesszővel elválasztva) – az első telepített fog megjelenni
- `font-size`: betűméret
- `font-weight`: betűvastagság
 - értékek: `normal` (alapértelmezett vastagság), `bold` (vastag), `bolder` (vastagabb), `lighter` (vékonyabb), szám (100 és 900 között)
- `font-style`: italic; – dőlt betűssé tételezés
- `color`: betűszín
- `text-align`: szöveg vízszintes igazítása
 - értékek: `left` (balra), `right` (jobbra), `center` (középre), `justify` (sorkizárt)

```
<p>Ez itt egy <span>formázandó</span> szöveg.</p>

<!-- CSS utasítások a head-ben --&gt;
&lt;style&gt;
    p {
        font-family: Arial, Verdana, sans-serif;
        font-size: 18px;
        font-weight: bold;
        color: darkblue;
    }

    span {
        font-family: 'Courier New';
        font-style: italic;
        color: red;           /* vörös */
    }
&lt;/style&gt;</pre>
```

Ez itt egy *formázandó* szöveg.

F. Háttér formázása

- **background-color**: háttérszín
 - megadható szín vagy `transparent` (átlátszó, alapértelmezett érték)
- **background-image**: háttérkép
 - `url('foo.png')`: kép elérési útvonala
- **background-repeat**: háttérkép ismétlődése
 - `repeat` (vízszintes és függőleges ismétlődés, alapértelmezett), `repeat-x` (vízszintes ismétlődés), `repeat-y` (függőleges ismétlődés), `no-repeat` (nem ismétli)
- **background-position**: háttérkép pozíciója (x, y)
 - x: vízszintes pozíció – `left` (bal), `right` (jobb), `center` (középre)
 - y: függőleges pozíció – `top` (fent), `bottom` (lent), `center` (középre)
 - nem kötelező minden értéket megadni (pl. `background-position: center;`)
- **background-size**: háttérkép mérete (x, y)
 - x: vízszintes méret, y: függőleges méret
 - nem kötelező minden értéket megadni
 - `auto`: automatikus (alapértelmezett)
 - `cover`: kitölți a hátteret az eredeti méretarányok megtartásával
 - `contain`: a legnagyobb méretre skáláz, az eredeti méretarányok megtartásával
- a háttér tulajdonságai összevonva is megadhatók a **background** CSS tulajdonsággal
 - pl. `background: lightblue url('bg.jpg') no-repeat center;`
 - színátmenetes kitöltés: `linear-gradient(irány, szín1, szín2);`

```
<h1>Üdvözöllek a weboldalamon!</h1>

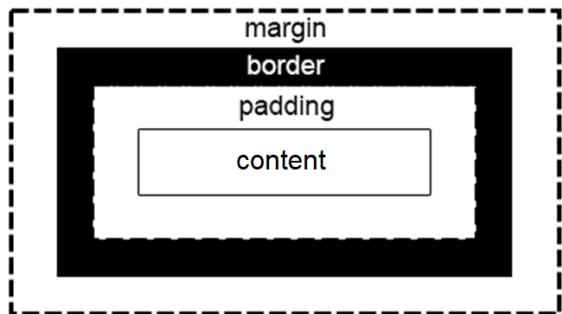
<!-- CSS utasítások a head-ben -->
<style>
  html, body {
    height: 100%;
  }

  body {
    background: url('bg.jpg') no-repeat center;
  }

  h1 {
    background-color: darkblue; /* régebbi böngészők */
    background: linear-gradient(90deg, darkblue, blue);
    color: #fff;
    text-align: center;
  }
</style>
```



G. A dobozmodell



Minden HTML objektum felfogható egy téglalap alakú dobozként. Ezt nevezzük **dobozmodellnek**.

A dobozon belül szerepel a tartalom, ami körül térköz (padding) található. Ezen kívül az objektumok rendelkezhetnek szegéllyel (border) is. A szegély körül található a külső margó (margin).

a. Térközök megadása

- **padding:** [padding-top] [padding-right] [padding-bottom] [padding-left];
 - a paraméterek rendre: felső, jobb, alsó, bal térköz
 - ezek a paraméterek külön-külön is megadhatók
 - nem megadott paraméter esetén az értéket a szemközti oldaltól örökli

b. Margók megadása

- **margin:** [margin-top] [margin-right] [margin-bottom] [margin-left];
 - a paraméterek rendre: felső, jobb, alsó, bal margó
 - ezek a paraméterek külön-külön is megadhatók
 - nem megadott paraméter esetén az értéket a szemközti oldaltól örökli

```
/* elem vízszintesen középre igazítása */
div {
    margin-left: auto;
    margin-right: auto;
    text-align: center;
}
```

c. Szegély megadása

- **border:** [border-width] [border-style] [border-color];
 - a paraméterek külön-külön is megadhatók
 - **border-width:** a szegély vastagsága
 - px, **thin** (vékony), **medium** (közepes), **thick** (vastag)
 - **border-style:** a szegély stílusa
 - **solid** (folytonos), **dashed** (szaggatott), **dotted** (pontozott), **double** (dupla), **groove** (faragott), **ridge** (kidomborodó) **none** (nincs), **hidden** (rejtett), ...
 - **border-color:** a szegély színe

```
<div><h2>Gratulálok! Nyertél egy iPhone-t</h2></div>

<!-- CSS utasítások a head-ben -->
<style>
    div {
        border: 5px double red;
        background-color: blue;
        color: yellow;
        text-align: center;
    }
</style>
```

Gratulálok! Nyertél egy iPhone-t

Példa: Szegély megadása CSS-ben

H. Elem megjelenítése

- **display:** elem megjelenítése
 - **inline:** sorszintű – vízszintesen ismétlődik, amíg kifér, utána új sorban ugyanúgy
 - **block:** blokkszintű – előtte és utána térköz található
 - **inline-block:** inline, de megadhatók a méretei (szélesség, magasság)
 - **none:** elem eltüntetése
 - nagy népszerűségnek örvend a [flexbox](#) és a [grid](#)

```

<p class="magical">Sitty-sutty, eltűntem</p>
<p>Én pedig itt maradtam</p>

<!-- CSS utasítások a head-ben -->
<style>
    img { /* középre igazítás */
        display: block;
        margin: auto;
    }

    .magical { display: none; }
</style>
```



Én pedig itt maradtam

WEBTERVEZÉS – GYAKORLATI JEGYZET

3. gyakorlat

HTML5 szemantikus tagek,
CSS kijelölők, formázás és pozícionálás

Készítették:

Cservenák Bence

Farkas Anikó

Savanya Sándor

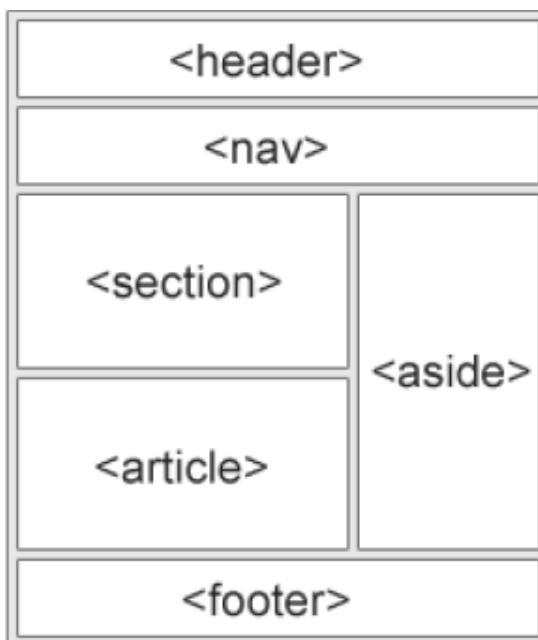
FIGYELEM!

A jegyzet folyamatosan készül, így előfordulhatnak benne apróbb hibák, hiányosságok, elírások. Ha valaki esetleg ilyet találna, kérem, írjon a Cservenak.Bence@stud.u-szeged.hu címre, hogy mihamarabb javíthassuk.

1. A HTML5 szemantikus tagok

A HTML5 szabványban bevezettek új tagokat a tartalom strukturálására. Ezek a tagok főként szemantikai jelentéssel bíró, egyszerű div elemeket helyettesítő blokkszintű (block) objektumok.

- <header>...</header>: fejléc
- <footer>...</footer>: lábléc
- <nav>...</nav>: navigációs menü
- <aside>...</aside>: oldalsáv
- <main>...</main>: az oldal lényegi részei
- <section>...</section>: logikai egység
- <article>...</article>: önálló tartalom



Példa: Egy lehetséges “layout” a fenti tagok használatával

2. CSS kijelölők (Folytatás)

A múlt órán megismertük a legalapvetőbb CSS kijelölőkkel (elemkijelölő, osztálykijelölő, azonosítókijelölő, univerzális kijelölő). Nézzünk néhány példát egyéb kijelölőkre!

- Amikor több kijelölőre is ugyanaz a stílus vonatkozik, vesszővel elválasztva felsoroljuk a kijelölőket (group selector)

```
h1, h2, .warning { color: red; }
```

WEBTERVEZÉS – 3. GYAKORLAT

- **Element1 Element2:** az Element1-en belüli összes Element2-t kijelöli
- **Element1>Element2:** minden olyan Element2-t kijelöl, amelynek közvetlen szülője Element1

```
<body>
  <p>első</p>
  <div>
    <p>második</p>
  </div>
  <p>harmadik</p>
</body>
```

```
body p {
  /* első, második, harmadik */
}

body>p {
  /* első, harmadik */
}
```

- **Element[a]:** minden olyan Element elemet kijelöl, ami rendelkezik az a attribútummal
- **Element[a="val"]:** minden olyan Element elemet kijelöl, amelynek a attribútumnak az értéke val

```
<p id="első">első</p>
<p id="foo">második</p>
<p id="bar">harmadik</p>
<p>negyedik</p>
```

```
p[id] { /* első, második, harmadik */ }
p[id="foo"] { /* második */ }
```

- Pseudo-class segítségével történő kijelölés:

- a hivatkozások 4 állapota

```
a:link { /* ha még nem kerestük fel... */ }
a:visited { /* ha már felkerestük... */ }
a:hover { /* ha a kurzor fölötté áll... */ }
a:active { /* ha rákattintunk a hivatkozásra... */ }
```

- néhány egyéb pseudo-class kijelölő

- **:first-child:** a beágyazó objektumnak legelső gyermeke
- **:nth-child(n):** a beágyazó objektumnak n-edik gyermeke (n pozitív egész)
- **:last-child:** a beágyazó objektumnak utolsó gyermeke
- **:first-of-type:** az azonos típusú, vele egy szinten lévő elemek közül a legelső
- **:nth-of-type(n):** az azonos típusú, vele egy szinten lévő elemek közül a n-edik (n pozitív egész)
- **:last-of-type:** az azonos típusú, vele egy szinten lévő elemek közül az utolsó

```
/* a legelső listaelem */
li:first-child { ... }

/* a harmadik listaelem */
li:nth-child(3) { ... }
```

Az összes kijelölő: https://www.w3schools.com/cssref/css_selectors.asp

3. Hivatkozások, listák és táblázatok formázása

A. Hivatkozások formázása

- A hivatkozásokra is használhatók a múlt órán tanult szöveg- és háttérformázások
- A hivatkozásokat állapot alapján is formázhatszuk (*lásd: előző pont, pseudo-class*)
- A `text-decoration` tulajdonsággal megadhatjuk a szövegdekorációt
 - `underline`: a hivatkozás alatt egy vonal jelenik meg (alapértelmezett)
 - `none`: általában a hivatkozás alatti vonal eltüntetésére használjuk

```
<a href="https://www.google.com/">Első link</a> <br/>
<a href="https://www.google.com/">Második link</a>


a:nth-of-type(2) { text-decoration: none; }
```

Első link
Második link

B. Listák formázása

- A listák felsorolásjele a `list-style-type` tulajdonsággal módosítható
 - rendezetlen lista (`ul`) esetén: `disc` (teli karika), `circle` (üres karika), `square` (teli négyzet), `none` (nincs felsorolásjel)
 - rendezett lista (`ol`) esetén: `decimal` (arab szám), `lower-roman/upper-roman` (római szám), `lower-alpha/upper-alpha` (kisbetű/nagybetű), `none` (nincs felsorolásjel)
- A `list-style-image` tulajdonsággal kép is megadható listaelemként
 - a kép elérési útvonalát az `url('elérési útvonal')` segítségével adjuk meg
- A `list-style-position` tulajdonsággal beállítható a felsorolásjelek pozíciója
 - `outside`: a felsorolás függőleges vonalába a felsorolt tartalom került
 - `inside`: a felsorolás függőleges vonalába a listajelek kerülnek

C. Táblázatok formázása

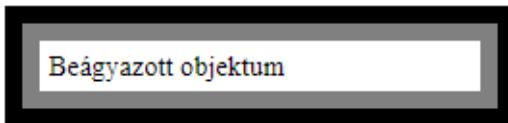
- A táblázatra, annak soraira, celláira is használhatók a tanult szöveg- és háttérformázások, valamint a térközre, margóra és szegélyre vonatkozó utasítások
- A `td:hover` kijelölő kijelöli azt a táblázatcellát, ami fölött a kurzor áll
- A `vertical-align` utasítással megadhatjuk a cella tartalmának függőleges igazítását (`middle`: középre igazítás)
- A `caption-side` tulajdonsággal megadhatjuk a táblázat címének (`caption`) pozíóját
 - lehetséges értékei: `top` (a táblázat fölött, alap), `bottom` (a táblázat alatt)
- A `border-collapse: collapse;` utasítással összevonhatjuk a táblázat szegélyeit
- A `table { margin: auto; }` utasítással a táblázat vízszintesen középre igazítható

D. A **box-sizing** tulajdonság

- Segítségével megadhatjuk, hogy az elem méretébe (szélességébe és magasságába) beleszámítanak-e a belső térközök (padding) és a szegélyek (border)
- Gyakori értékei:
 - content-box**: a méretbe csak a tartalom tartozik bele (alapértelmezett)
 - border-box**: a méretbe a tartalom, a padding és a border is beletartozik
- Ha a méreteket %-ban adjuk meg, akkor kiemelten fontos lehet, hogy a padding és border is beletartozzon a méretbe

```
<div id="parent">
  <div id="child">Beágyazott objektum</div>
</div>

<!-- CSS formázás --&gt;
#parent { border: 10px solid black; }
#child { box-sizing: border-box; width: 100%;
          border: 10px solid gray; padding: 5px; }</pre>
```



Beágyazott objektum

Megjegyzés: A példában box-sizing: content-box; esetén a beágyazott objektum "kilógná"

4. A CSS rangsor

Kérdés: Több, egymásnak ellentmondó CSS formázás közül melyik lesz érvényes az objektumra?

Válasz: Egy előre definiált rangsor alapján dől el

- !important** CSS utasítások (**legmagasabb prioritás**)
 - inline CSS (a tagek **style** attribútumaként adjuk meg)
 - azonosítókijelölő (**id** alapján)
 - osztálykijelölő (**class** alapján)
 - elemkijelölő (**legalacsonyabb prioritás**)
- A felhasználó által definiált **!important** stílusok erősebbek, mint a programozó által definiáltak
 - A felhasználó által definiált egyéb stílusok viszont önmagukban **nem** erősebbek, mint a programozó által definiáltak
 - Az összetett kijelölő esetén a részkijelölők prioritásai összeadódnak

Példa:

Legyenek adottak a következő prioritások: p(id selector) = 100, p(class selector) = 10, p(element selector) = 1. Ekkor a #main .container p összetett kijelölő prioritása: 100 + 10 + 1 = 111.

- Ha az eddigiek alapján nem dőlt el a rangsor, akkor a kódban későbbi formázás lesz érvényes

5. Pozícionálás, helyzetmegadás

A. A `position` CSS tulajdonság

- A HTML objektumok pozícióját a `position` CSS tulajdonsággal adhatjuk meg, értékei:
 - `static` (alap): a normál szövegfolyamban lévő hely, a pozícionálásnak nincs hatása
 - `relative`: a normál szövegfolyambeli helyéhez képest elmozdítja a kért mértékben
 - `absolute`: a legközelebbi, **nem static** helyzetű beágyazó objektum belső széléhez (annak hiányában a viewport széléhez) képest pozícionál
 - `fixed`: az elem gördítéskor is rögzített helyen marad

A **sticky** kulcsszó: https://www.w3schools.com/howto/howto_css_sticky_element.asp

- **Nem static** módon pozícionált HTML objektumok esetén megadhatók:
 - `top, bottom`: függőleges igazítás
 - `left, right`: vízszintes igazítás

```
<div class="container">
  <b>Pozícionálás</b>
  <div class="first box">Első</div>
  <div class="second box">Második</div>
  <div class="third box">Harmadik</div>
</div>

<!-- CSS utasítások --&gt;
.box { padding: 20px 10px; }
.container { background-color: lightblue; position: relative; height: 800px; }
.first { background-color: tomato; position: absolute; top: 0; right: 0; }
.second { background-color: mediumaquamarine; /* top: 500px; */ }
.third { background-color: lightgreen; position: fixed; }</pre>
```

Kódpélda: A pozícionálással kapcsolatos CSS utasítások működése

A fenti példa magyarázata:

- Az `absolute` módon pozícionált "Első" feliratú doboz az őt beágyazó nem `static` (hanem jelen esetben `relative`) helyzetű doboz (`container`) jobb felső sarkához igazodik.
- A "Második" feliratú doboz `static`, így a pozícionálásnak nincs hatása.
- A `fixed` helyzetű "Harmadik" feliratú doboz gördítéskor is rögzített helyen marad.

B. Úsztatás

- A `float` tulajdonság úsztatja az elemet, amelyet a másik irányból a tartalom körbefolyhat
 - értékei: `left` (balra úsztatás), `right` (jobbra úsztatás), `none` (nincs úsztatás)
- A `clear` tulajdonság megakadályozza másik, úsztatott elem elhelyezését
 - értékei: `left` (balról nem lehet úsztatott elem), `right` (jobbról nem lehet úsztatott elem), `both` (sem balról, sem jobbról nem lehet úsztatott elem), `none` (balról és jobbról is lehet úsztatott elem)

```



A zsiráf (Giraffa camelopardalis) Afrikában élő párosujjú patás emlős, a legmagasabb és leghosszabb nyakú szárazföldi élőlény. A szavannák lakója az ókortól kedvelt attrakció volt – Rómában először Julius Caesar mutatott be zsiráfokat az amfiteátrumi játékokon – és ma is népszerű szafarikon és állatkertekben. Neve arab eredetű, olasz közvetítéssel jutott el a magyarba a késő középkorban. A tudományos nevében szereplő camelopardalis a faj addig használt nevére, a görög kamélopardaliszra („tevepárdus”) utal.</p>

<!-- CSS utasítások -->
img {
    height: 140px;
    float: left;
    margin-right: 10px;
}


```



A zsiráf (Giraffa camelopardalis) Afrikában élő párosujjú patás emlős, a legmagasabb és leghosszabb nyakú szárazföldi élőlény. A szavannák lakója az ókortól kedvelt attrakció volt – Rómában először Julius Caesar mutatott be zsiráfokat az amfiteátrumi játékokon – és ma is népszerű szafarikon és állatkertekben. Neve arab eredetű, olasz közvetítéssel jutott el a magyarba a késő középkorban. A tudományos nevében szereplő camelopardalis a faj addig használt nevére, a görög kamélopardaliszra („tevepárdus”) utal.

WEBTERVEZÉS – GYAKORLATI JEGYZET

4. gyakorlat

A HTML5 úrlapok

Készítették:

Cservesnák Bence
Farkas Anikó
Savanya Sándor

FIGYELEM!

A jegyzet folyamatosan készül, így előfordulhatnak benne apróbb hibák, hiányosságok, elírások. Ha valaki esetleg ilyet találna, kérem, írjon a Cservesnák.Bence@stud.u-szeged.hu címre, hogy mihamarabb javíthassuk.

1. Úrlapok létrehozása

Eddig főként a weboldal megjelenésén volt a hangsúly. Amennyiben viszont felhasználói bemenetet szeretnénk fogadni későbbi feldolgozás céljából, használjuk a HTML úrlapokat.

- A HTML úrlapokat a `<form>` és `</form>` tagok között adjuk meg (a body-ban)
 - `action` attribútum: az úrlap elküldésekor végrehajtandó feladat (pl. PHP szkript)
 - `method` attribútum: az úrlapadatok továbbításának módja
 - `method="GET"`: az adatok az URL-ben kerülnek továbbításra (alapértelmezett)
 - `method="POST"`: az adatok a háttérben továbbítódnak
 - `enctype` attribútum: úrlapadatok kódolása
 - `autocomplete` attribútum: automatikus kitöltési javaslatok
 - `autocomplete="on"`: kitöltési javaslatok engedélyezése (alapértelmezett)
 - `autocomplete="off"`: kitöltési javaslatok letiltása
 - `novalidate` attribútum: tiltja az úrlap elküldésekor végrehajtandó ellenőrzéseket

2. Fontosabb úrlapelemek

- `<input/>`: egysoros beviteli mező (páratlan tag)
 - `name` attribútum: mezőnév – ez alapján tudjuk lekérni a beírt értéket
 - `size` attribútum: szélesség
 - `maxlength` attribútum: a beírható karakterek maximális száma
 - `value` attribútum: alapértelmezett érték
 - `placeholder` attribútum: mintaérték a felhasználó számára
 - `autocomplete` attribútum: ugyanaz, mint a `form`-nál (`on/off`)
 - `tabindex` attribútum: Tab hatására milyen sorrendben kéri be az adatokat
 - `type` attribútum: típus
 - `<input type="text"/>`: rövid szöveg
 - `<input type="password"/>`: jelszó
 - `<input type="number"/>`: szám
 - `<input type="range"/>`: intervallum
 - `min`: a legkisebb választható érték
 - `max`: a legnagyobb választható érték
 - `step`: lépésköz
 - `<input type="tel"/>`: telefonszám
 - `<input type="email"/>`: e-mail cím (ellenőrzi a formátumát)
 - `<input type="url"/>`: URL (ellenőrzi a formátumát)
 - `<input type="color"/>`: szín

- <input type="date"/>: dátum (`min` és `max` opcionális attribútumok)
 - <input type="time"/>: időpont
 - <input type="datetime"/>: dátum és időpont
 - <input type="file"/>: fájlcsatolás
 - <input type="search"/>: keresendő szöveg
 - <input type="radio"/>: választógomb
 - egy opció jelölhető meg
 - a választógombok csoportosítása a `name` attribútum alapján történik
 - <input type="checkbox"/>: jelölőnégyzet
 - több opció is megjelölhető
 - <input type="button"/>: nyomógomb
 - <input type="reset"/>: visszaállítja az űrlapmezők eredeti értékét
 - <input type="submit"/>: elküldés gomb
 - <input type="hidden"/>: rejtett mező (technikai célok)
 - `required` attribútum: kötelező kitölteni
 - `disabled` attribútum: letiltott (halvány)
- <textarea>...</textarea>: több soros beviteli mező
 - fontosabb attribútumok: `name` (mezőnév), `rows` (sorok száma), `cols` (oszlopok száma)
 - <select>...</select>: választólista
 - <option>...</option>: egy kiválasztható opció
 - lehetséges attribútumok: `value` (elküldendő érték), `selected` (alapból kiválasztott)
 - a `multiple` attribútummal egyszerre több opció is kiválasztható
 - <fieldset>...</fieldset>: mezőcsoportosítás
 - <legend>...</legend>: csoporthoz köthető felirat
 - <label>...</label>: elemfelirat
 - `for` attribútumának értékeként megadjuk annak a mezőnek az `id`-jét, amihez tartozik
 - ha közrefogja az űrlapelemet, akkor nem kell a `for` attribútum

3. Néhány CSS pseudo-class űrlapokhoz

- `:required`: a required attribútummal rendelkező mezők
- `:optional`: a required attribútummal nem rendelkező mezők
- `:focus`: a fókuszon lévő elem
- `:checked`: bejelölt (radio, checkbox, option)
- `:disabled`: a disabled attribútummal rendelkező (letiltott) mezők
- `:enabled`: a disabled attribútummal nem rendelkező (engedélyezett) mezők

4. HTML5 űrlap – Példa

HTML kód (body) :

```
<form action="process.php" method="POST">
<fieldset>
    <legend>Regisztrációs adatok</legend>
    <label>Név: <input type="text" name="full-name" size="30"/></label> <br/>
    <label>Felhasználónév: <input type="text" name="uname" required/></label> <br/>
    <label>Jelszó: <input type="password" name="pwd" placeholder="Jelszó..."/></label> <br/>
    <label>Születési dátum: <input type="date" name="birth-date"/></label> <br/>
    <label>E-mail: <input type="email" name="email"/></label> <br/>
    <label>Azonosító: <input type="number" name="uid" value="123" disabled/></label> <br/>
</fieldset>

<label for="education">Legmagasabb iskolai végzettség:</label>
<select id="education">
    <option value="elementary">8 általános</option>
    <option value="highschool" selected>Érettségi</option>
    <option value="uni">Felsőfokú végzettség</option>
</select> <br/>

Nem:
<label for="foo1">Férfin: </label>
<input type="radio" id="foo1" name="sex" value="m"/>
<label for="foo2">Nő: </label>
<input type="radio" id="foo2" name="sex" value="f"/>
<label for="foo3">Egyéb: </label>
<input type="radio" id="foo3" name="sex" value="other" checked/> <br/>

Hobbik:
<label for="bar1">Fotózás: </label>
<input type="checkbox" id="bar1" name="hobbies" value="photo"/>
<label for="bar2">Főzés: </label>
<input type="checkbox" id="bar2" name="hobbies" value="cooking"/>
<label for="bar3">Soroztnézés: </label>
<input type="checkbox" id="bar3" name="hobbies" value="series"/> <br/>

<label>Bemutatkozás: <textarea name="intro" maxlength="200"></textarea></label> <br/>
<input type="reset" name="reset-btn" value="Törlés"/>
<input type="submit" name="submit-btn" value="Elküld"/>
</form>
```

CSS-kód:

```
<style>
    input, select, textarea, fieldset { margin-bottom: 10px; }
    textarea { display: block; }
    input[type="reset"], input[type="submit"] {
        outline: none;
        border: none;
        background-color: #4CAF50;
        color: #fff;
        font-size: 110%;
        width: 120px;
        height: 50px;
    }
    input:focus { background-color: #BFFFC3; }
</style>
```

WEBTERVEZÉS – 4. GYAKORLAT

Regisztrációs adatok

Név:

Felhasználónév:

Jelszó: Jelszó...

Születési dátum: éééé. hh. nn.

E-mail:

Azonosító: 123

Legmagasabb iskolai végzettség: Érettségi ▾

Nem: Férfi: Nő: Egyéb:

Hobbik: Fotózás: Főzés: Soroztnézés:

Bemutatkozás:

/

[Törlés](#)

[Elküld](#)

Ábra: A fenti kód eredménye

WEBTERVEZÉS – GYAKORLATI JEGYZET

5. gyakorlat

A HTML és CSS további lehetőségei

Készítették:

Cservenák Bence
Farkas Anikó
Savanya Sándor

FIGYELEM!

A jegyzet folyamatosan készül, így előfordulhatnak benne apróbb hibák, hiányosságok, elírások. Ha valaki esetleg ilyet találna, kérem, írjon a Cservenak.Bence@stud.u-szeged.hu címre, hogy mihamarabb javíthassuk.

1. A HTML további használati lehetőségei

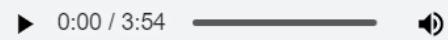
A. Multimédia elemek

A HTML5-ös szabvány támogatja multimédia (audió, videó) beágyazását a weboldalunkra.

- `<audio>...</audio>`: hangállomány beszúrása
 - `controls` attribútum: vezérlőgombok megjelenítése
 - `<source/>`: több alternatív fájl megadása (páratlan tag)
 - a böngésző az első általa felismerhetőt fogja beágyazni
 - `src`: a fájl elérési útvonala
 - `type`: a fájl MIME-típusa
- `<video>...</video>`: videó beszúrása
 - `controls` attribútum és `<source/>` tag ugyanúgy, mint hangállomány esetén
 - `autoplay` attribútum: automatikus lejátszás
 - `width`: videó szélessége, `height`: videó magassága

```
<h2>A kedvenc zeném</h2>
<audio controls>
  <source src="powerwolf.mp3" type="audio/mpeg"/>
  <source src="greenday.wav" type="audio/wav"/>
  Ez akkor jelenik meg, ha a böngésző nem támogatja
  hangállomány beágyazását
</audio>
```

A kedvenc zeném



```
<h2>A kedvenc videóm</h2>
<video controls width="360" height="280">
  <source src="allstar.mp4" type="video/mp4"/>
  <source src="rickroll.ogg" type="video/ogg"/>
  Ez akkor jelenik meg, ha a böngésző nem támogatja
  videóállomány beágyazását
</video>
```

A kedvenc videóm



B. Egyéb beágyazások

Az `<object>`, illetve `<embed>` tagokkal

C. A `canvas` tag

- `<canvas>...</canvas>`: JavaScript alapú rajzolásra használjuk
 - grafika, animációk, böngészős játékok
 - `width`: vászon szélessége
 - `height`: vászon magassága

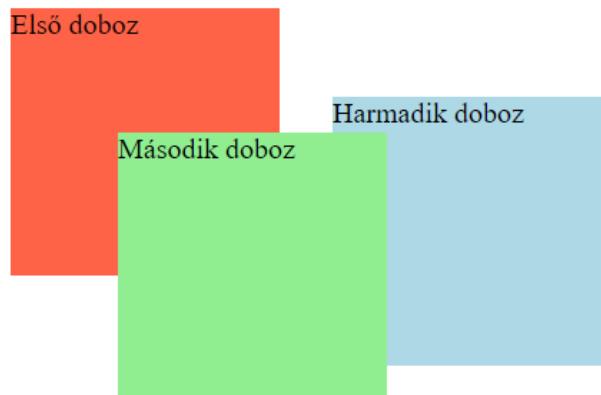
2. A CSS további használati lehetőségei

A. A **z-index** tulajdonság

- Értékéül megadható egy egész szám (akár negatív szám is)
- Az egymást takaró elemek közül az kerül előtérbe, amelyiknek nagyobb a **z-index** értéke

```
<div class="first">Első doboz</div>
<div class="second">Második doboz</div>
<div class="third">Harmadik doboz</div>

<!-- CSS kód a head-ben -->
<style>
    div { width: 150px; height: 150px; }
    .first {
        background-color: tomato;
        position: absolute;
        top: 50px; left: 20px;
        z-index: 1;
    }
    .second {
        background-color: lightgreen;
        position: absolute;
        top: 120px; left: 80px;
        z-index: 200;
    }
    .third {
        background-color: lightblue;
        position: absolute;
        top: 100px; left: 200px;
        z-index: -1;
    }
</style>
```



B. Lekerekített sarkok

- **border-radius:** az elem sarkainak sugara
 - paraméterezése: [bal felső sarok] [jobb felső sarok] [jobb alsó sarok] [bal alsó sarok]
 - nem megadott paraméterérték esetén az értéket szemben lévő saroktól öröklí

```
<div class="rounded">Ez egy lekerekített sarkú téglalap</div>

<!-- CSS kód a head-ben -->
<style>
    .rounded { border: 2px solid red; border-radius: 0 10px 0 25px; height: 50px;
               width: 300px; padding: 10px; }
</style>
```

Ez egy lekerekített sarkú téglalap

C. Árnyék

- **text-shadow:** szövegárnyék
 - paraméterezés: [vízszintes árnyék] [függőleges árnyék] [mérték] [szín]
 - az utolsó két paraméter elhagyható
- **box-shadow:** dobozszerű elem körüli árnyék
 - paraméterezés: [vízszintes árnyék] [függőleges árnyék] [mérték] [kiterjedés] [szín]
 - az utolsó három paraméter elhagyható

```
<p>Ez egy árnyékkal rendelkező szöveg</p>
<div>Ez egy árnyékkal rendelkező doboz</div>

<!-- CSS kód a head-ben -->
<style>
    p { text-shadow: 2px 0 lightgray; }
    div { border: 1px solid black; box-shadow: 0 3px lightblue, 0 2px 25px yellow; }
</style>
```

Ez egy árnyékkal rendelkező szöveg

Ez egy árnyékkal rendelkező doboz

D. Pseudo-elemek

A CSS pszeudo-elemeket HTML objektumok specifikus részeinek formázására használjuk.

- **::before:** generált tartalom közvetlenül az elem előtt
- **::after:** generált tartalom közvetlenül az elem után
- **::selection:** a kijelölt szövegrész
- **::first-letter:** a szöveg első karaktere
- **::first-line:** a szöveg első sora

```
<p>Első bekezdés</p>
<p>Második bekezdés</p>
<p>Harmadik bekezdés</p>

<!-- CSS kód a head-ben -->
<style>
    p::selection { background-color: lightgreen; }
    p::first-letter { font-size: 120%; color: red; }
    p::after { content: " -- bekezdés vége"; }
</style>
```

Első bekezdés -- bekezdés vége

Második bekezdés -- bekezdés vége

Harmadik bekezdés -- bekezdés vége

Példa: A kijelölés, első betű és az elem utáni tartalom megadása pseudo-elemekek segítségével

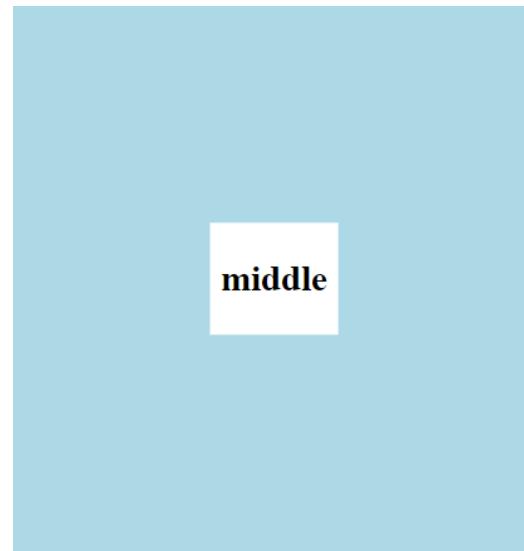
E. Transzformálás

- **transform:** HTML elem transzformálása
 - A transzformálás régebbi böngészőkben nem érhető el
 - `translate(x, y)`: eltolja az elemet a jelenlegi helyéhez képest (x: jobbra, y: le)
 - `rotate(xdeg)`: elforgatja az elemet x fokkal (x negatív is lehet)
 - `scale(w, h)`: nyújtás (w: szélesség, h: magasság)
 - `scaleX(n)`: eredeti szélesség megnyújtása n-szeresére
 - `scaleY(n)`: eredeti magasság megnyújtása n-szeresére
 - `skew(xdeg, ydeg)`: elferdítés (x: vízszintes, y: függőleges)
 - `skewX(ndeg)`: vízszintes elferdítés n fokkal
 - `skewY(ndeg)`: függőleges elferdítés n fokkal
 - `matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())`: a fentiek összevonása
 - `rotateX(ndeg)`: elforgatja az elemet az x-tengely körül n fokkal
 - `rotateY(ndeg)`: elforgatja az elemet az y-tengely körül n fokkal
 - `rotateZ(ndeg)`: elforgatja az elemet a z-tengely körül n fokkal

```
<div><h1>middle</h1></div>

<!-- CSS kód a head-ben --&gt;
&lt;style&gt;
    body { background-color: lightblue; }

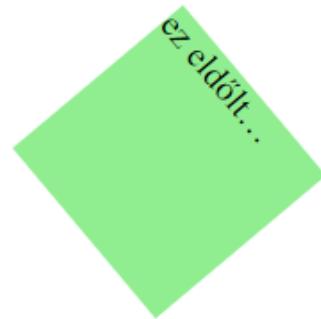
    div {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        margin: 0 auto;
        text-align: center;
        background-color: white;
        padding: 10px;
    }
&lt;/style&gt;</pre>
```



Példa: HTML objektum középre igazítása, vízszintesen és függőlegesen

```
<div>ez eldőlt...</div>

<!-- CSS kód a head-ben -->
<style>
  div {
    transform: translate(50px, 20px) rotate(50deg);
    width: 100px;
    height: 100px;
    background-color: lightgreen;
  }
</style>
```



Példa: HTML objektum eltolása, és 50 fokkal való elforgatása

F. A **transition** tulajdonság (áttűnések)

- Időben dinamikusan változtathatjuk meg elemeink stílusát azok állapotváltásai között
- Az áttűnések a régebbi böngészőkben nem érhetők el
- Megadjuk a CSS tulajdonságot, amit az áttűnés megváltoztat, valamint az áttűnéshez szükséges időt (másodpercben)
- **transition-timing-function:** az átmenet lefolyásának típusa
 - **linear:** egyenletes állapotváltás
 - **ease:** fokozatosan lassuló állapotváltás
 - **ease-in:** gyorsuló állapotváltás
 - **ease-out:** lassuló állapotváltás
 - **ease-in-out:** gyorsuló, majd lassuló állapotváltás
- Vesszővel elválasztva több átmenet is megadható

```
<div></div>

<!-- CSS kód a head-ben -->
<style>
  div {
    width: 100px;
    height: 100px;
    background: tomato;
    transition: width 2s, height 2s;
    transition-timing-function: ease-in;
  }

  div:hover { width: 300px; height: 300px; }
</style>
```

Példa: Egy téglalap alakú doboz méreteinek megnövelése áttűnésekkel

G. Animációk

- A CSS lehetőséget biztosít animációk készítésére JavaScript vagy Flash használata nélkül
- Az animációk régebbi böngészőkben nem érhetők el
- `@keyframes animacio_neve { ... }`: az animáció kódja
 - `from { ... }`: az animáció kezdetén fennálló stílus
 - `to { ... }`: az animáció befejezése után fennálló stílus
 - megadhatók százalékértékek is (az animáció hány százalékban hajtódott végre)
- Az animálandó elem esetén megadható CSS tulajdonságok:
 - `animation-name`: az animáció neve (ami a `@keyframes` után áll)
 - `animation-duration`: az animáció hossza
 - `animation-delay`: animáció kezdete előtti késleltetés
 - `animation-iteration-count`: hányszor játszódjon le az animáció
 - megadható szám vagy `infinite` (végtelen)
 - `animation-timing-function`: az animáció lefolyásának típusa
 - `linear, ease, ease-in, ease-out, ease-in-out` - mint a `transition`-nél
 - `animation-direction`: az animáció iránya
 - `normal`: az animáció előrefelé lesz lejátszva (alap)
 - `reverse`: az animáció visszafelé lesz lejátszva
 - `alternate`: az animáció először előrefelé, majd visszafelé lesz lejátszva
 - ezek összevonva is megadhatók: `animation: name duration timing-function delay iteration-count direction;`

```
@keyframes elseo {
  0% { background-color: yellow; }
  25% { background-color: orange; }
  50% { background-color: tomato; }
  75% { background-color: lightgray; }
  100% { background-color: lightgreen; }
}

body {
  animation-name: elseo;
  animation-duration: 10s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

Példa: Váltakozó háttérszín animáció

```
<div class="float-in">CSS</div>

<!-- CSS kód a head-ben --&gt;
&lt;style&gt;
  @keyframes masodik {
    from { left: -50%; }
    to { left: 50%; }
  }
  .float-in {
    width: 100px; height: 100px;
    background-color: lightgreen;
    position: relative;
    text-align: center;
    animation: masodik 3s forwards;
  }
&lt;/style&gt;</code>
```

Példa: Beúszás effektus animáció

H. Media query-k, nyomtatási stíluslap

- Lényegük, hogy a megadott CSS formázást csak a meghatározott médiatípusok és feltételek esetén végzi el a böngésző → pl. különböző eszközökre és felbontásokra való optimalizálás
- Szintaxis: `@media [not|only] mediatype and (mediafeature) { CSS formázás }`
 - `not` (opcionális): az egész media query jelentését negálja
 - `only` (opcionális): régi, CSS3 media query-keket nem ismerő böngészőknél nem formáz
 - `mediatype` lehetséges értékei:
 - `screen`: számítógépek, táblagépek, okostelefonok, stb. képernyője
 - `print`: nyomtató (**nyomtatási stíluslap**)
 - `@page` direktívával megadhatjuk a nyomtatási margókat is
 - `speech`: felolvásóprogram
 - `all`: minden médiatípus eszköz (alapértelmezett)
 - `mediafeature` helyén megadjuk a feltételt, ami mellett elvégezzük a formázást

```


<p>A zsiráf (Giraffa camelopardalis) Afrikában
élfő párosujjú patás emlős...</p>

<!-- CSS kód a head-ben --&gt;
&lt;style&gt;
  img {
    height: 140px;
    float: left;
    margin-right: 10px;
  }

  @media only screen and (max-width: 768px) {
    img {
      float: none;
      display: block;
      margin: auto;
    }
  }
&lt;/style&gt;
</pre>

```

Példa: Egy media query, ami a 768px-nél nem szélesebb képernyők esetén megszűnteti a szöveg kép körüli körbe-futtatását, és középre igazítja a képet

```

@media print {
  p {
    font-size: 12pt;
  }

  img {
    display: none;
  }

  h1, h2, h3, h4, h5, h6 {
    page-break-after: avoid;
  }
}

@page :left {
  margin: 0.5cm;
}

```

Példa: Nyomtatási stíluslap készítése

I. Egyéb hasznos CSS ismeretek

- [Google Web Fonts betűtípusok használata](#)
- [Lenyiló menü készítése](#)
- [Hasábolás](#)
- [Elemek számozása](#)