Honlapkészítés

2. Előadás – CSS

(Cascading Stlye Sheets –
Egymásba ágyazott stíluslapok)

Paksy Patrik

HTML nem elég jó!

Minden szöveg formázása külön történik, így egy jól formázott oldalnál ugyanazt a stílust (pl: Verdana, 5-ös méret, zöld, félkövér) többször használjuk. Ekkor ahány helyen használjuk annyi helyen be kell írni ezeket a tulajdonságokat.

Negatívumok:

Fölösleges időtöltés! Még akkor is ha csak ctrl-c + crtl-v-vel megy
Sok a fölösleges kód → Lehetne sokkal átláthatóbb is!
Sok a fölösleges kód → Nagyobb a fájlméret!

Eddig valamelyest összemosódott a szerkezet kialakítása és a formázás, most először kialakítjuk a szerkezetet, majd utána formázunk!

Képek

A képek .jpg formátumúak. Kattints a linkekre, hogy láthasd a képeket

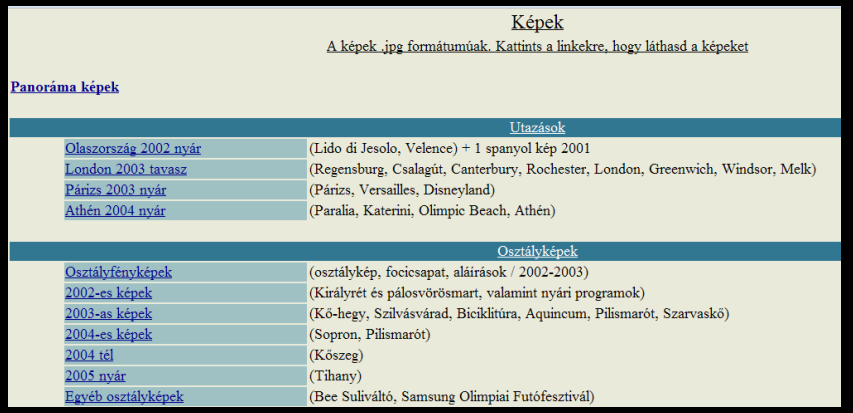
Panoráma képek

Utazások

Olaszország 2002 nyár	(Lido di Jesolo, Velence) + 1 spanyol kép 2001
London 2003 tavasz	(Regensburg, Csalagút, Canterbury, Rochester, London, Greenwich, Windsor, Melk)
Párizs 2003 nyár	(Párizs, Versailles, Disneyland)
Athén 2004 nyár	(Paralia, Katerini, Olimpic Beach, Athén)

Osztályképek

Osztályfényképek	(osztálykép, focicsapat, aláírások / 2002-2003)
2002-es képek	(Királyrét és pálosvörösmart, valamint nyári programok)
2003-as képek	(Kő-hegy, Szilvásvárad, Biciklitúra, Aquincum, Pilismarót, Szarvaskő)
2004-es képek	(Sopron, Pilismarót)
2004 tél	(Kőszeg)
2005 nyár	(Tihany)
Egyéb osztályképek	(Bee Suliváltó, Samsung Olimpiai Futófesztivál)



Képek

A képek .jpg formátumúak. Kattints a linkekre, hogy láthasd a képeket

Panoráma képek

Utazások	
Olaszország 2002 nyár	(Lido di Jesolo, Velence) + 1 spanyol kép 2001
London 2003 tavasz	(Regensburg, Csalagút, Canterbury, Rochester, London, Greenwich, Windsor, Melk)
Párizs 2003 nyár	(Párizs, Versailles, Disneyland)
Athén 2004 nyár	(Paralia, Katerini, Olimpic Beach, Athén)

Osztályképek	
Osztályfényképek	(osztálykép, focicsapat, aláírások / 2002-2003)
2002-es képek	(Királyrét és pálosvörösmart, valamint nyári programok)
2003-as képek	(Kő-hegy, Szilvásvárad, Biciklitúra, Aquincum, Pilismarót, Szarvaskő)
2004-es képek	(Sopron, Pilismarót)
2004 tél	(Kőszeg)
2005 nyár	(Tihany)
Egyéb osztályképek	(Bee Suliváltó, Samsung Olimpiai Futófesztivál)

CSS miért jó?

- Mert rendezettebb lesz a honlap
- Mert egy külön fájlba (*.css) tárolod az általad használt sablonokat
- Egy sablont hozol létre, majd ha változtatni kell, akkor elég csak a css fájlt megváltoztatni
- Jegyzettömbben lehet megírni a fájlt, és pl.: "proba.css" néven kell elmenteni
- A html kódban csak hivatkozni kell rá
- Leegyszerűsíti az oldalak forráskódjának írását
- Eddig nem használt formázási lehetőségek (amik nincsenek HTML-ben)

CSS esetében is az egyszerűbb és jobban szerkeszthető megoldást fogom bemutatni, vagyis táblázatban készítjük a szerkezetét az oldalnak, nem -ekkel.

Megtanulunk egy új nyelvet, amivel HTML-hez hasonlóan formázhatunk, de sokkal egyszerűbben! Hova írhatjuk ezeket az új nyelvi részeket? → köv.dia

Hogyan kezdjünk hozzá?

Két lehetőség a CSS kód írására:

HTML oldal <head> részébe

külön valami.css fájlba (szöveges állomány, jegyzettömb kell csak)

HTML <head>-be:

```
<html><head><title></title>
<style type="text/css">
<!--                          - komment kezdete, ez nem fog látszani az oldalon
        Ide kerül a css kód
-->                           - komment vége
</style>
</head><body></body></html>
```

Külön fájlba, ekkor csak hivatkozni fogunk rá:

```
<html><head><title></title>
<link href="fájlnév.css" rel="stylesheet" type="text/css">
</head><body></body></html>
```

Vagy:

```
<html><head><title></title>
<style type="text/css">
@import url(fájlnév.css);
</style>
</head><body></body></html>
```

Melyik jobb?

Egyértelműen, ha külön fájlba írjuk!

Ekkor bármennyi oldalban tudunk rá hivatkozni, így a „közös” fájl megváltoztatásával változik minden oldalon is a kinézet ←→ eddig: minden oldalon, minden helyen át kellett írni

Lehet olyan helyzet, hogy kivételesen egy-egy fájlban felülbírálnánk az eredeti, külön fájban lévő formázást, ekkor lehet jó, ha az egész CSS kódot a <head>-be írjuk. Hiszen ez csak arra az egyetlen oldalra érvényes, aminek a <head>-jében van.

CSS szintaxis – avagy mit írunk a fájlba

```
szelektor {
        tulajdonság1: érték1;
        tul2: érték2;
}
```

Konkrét példa:

```
h1 {
        font-size: 20px;
        color: black;
        font-weight: bold;
}
```

Először az elem nevét, majd kapcsos zárójel között a tulajdonságát, és a tulajdonság után kettőspontot írva a definícióját (értékét) adhatjuk meg, melyet pontosvesszővel választunk el a következő tulajdonságától.
Egy CSS utasítás két részből áll, a szelektor (vagyis megnevező),
és a deklaráció (formázása = tulajdonság-érték párok).

Az olyan tulajdonságoknál, ahol több szóból áll a tulajdonság neve, kötőjelet teszünk közé (pl.: font-size)

Az egyes tulajdonságok sorrendje lényegtelen.

Ékezetes neveket ne használjunk sehol se!

CSS szintaxis – avagy mit írunk a fájlba

```
szelektor {
        tulajdonság1: érték1;
        tul2: érték2;
}
```

Konkrét példa:

```
h1 {
        font-size: 20px;
        color: black;
        font-weight: bold;
}
```

Emlékszünk még a HTML-nél használt színezésre? Arra gondolhatunk, hogy a fenti kód ez lenne:
`<h1><font size="5" color="black"><b>Szöveg</b></font></h1>`
Sokat nem is tévedünk, hiszen nagyjából ugyanazt írja le, hogy nekünk kell egy bizonyos méretű és színű h1-es címsor.
DE! Figyeljük meg, hogy CSS-ben más a size mértékegysége: px (pixel)
→ nem csak néhány adott méret van, hanem mi választhatjuk tetszőlegesen!
DE! A <b> tagból attribútum lett!
→ CSS-ben a félkövér az egy tulajdonság: font-weight!
DE! HTML kódban egy adott h1-es címsorra érvényes amit odaírtunk! A CSS külön fájlban van
→ Mi van, ha több h1-es címsor van? Ekkor a CSS mindet ugyanolyanra formázza!!!

TEHÁT EGY HTML TAGOT HATÁROZTUNK MEG! (itt: <h1>)
AZ OLDALON LÉVŐ ÖSSZES ILYEN TAGRA UGYANAZ LESZ A FORMÁZÁS!!!

Mi a teendő, ha mi azt szeretnénk, hogy csak az egyik h1-es címsor legyen ilyen (több esetén)?

CSS szintaxis – avagy mit írunk a fájlba

```
szelektor {
        tulajdonság1: érték1;
        tul2: érték2;
}
```

Konkrét példa:

```
.sablon1 {
        font-size: 20px;
        color: black;
        font-weight: bold;
}
```

Előző példakód:

```
<h1><font size="5" color="black"><b>Szöveg</b></font></h1>
```

Mi van, ha mi azt szeretnénk, hogy csak az egyik h1-es címsor legyen ilyen (több esetén)?
Fent a szelektornak nem egy HTML tagot írunk, hanem egy tetszőleges nevet PONT után (pl: .nevem)

De ettől még honnan tudjuk, hogy melyik h1-es címsor kapja ezt a stílust?
→ Hivatkozni kell a fenti .sablon1-re a HTML kódban!

```
<h1 class="sablon1">Szöveg</h1>
```

A hivatkozás a class attribútummal történik, értékként a szelektor nevét kell megadni pont nélkül.
Ekkor ez a szöveg a külön fájlban megírt stílus szerint lesz formázva.
Látható: kevesebb a HTML kód, de ott a külön fájl, így van CSS is → Jó ez?
Igen, mert ezt a sablon1-et akárhány helyen megadhatom, és mindenhol csak ennyit kell beírni.

CSS szintaxis – avagy mit írunk a fájlba

```
szelektor {
        tulajdonság1: érték1;
        tul2: érték2;
}
```

Összefoglalás:

Ha a szelektor HTML tag:

Minden oldal összes olyan tagja a megadott szerint lesz formázva (ahol a stíluslapot használjuk)
Nem kell rá külön hivatkozni
Mi lehet szelektor? (nem az összes van itt felsorolva, csak ami nekünk fontos)
<body>,<table>,<tr>,<td>,<p>,címsorok (pl.<h1>), ,,
<img>,<input>,<ul>,<ol>,

Ha a szelektort mi adjuk meg:

Csak az lesz úgy formázva, amire külön hivatkozunk
Ilyenből sokkal több lesz, mint a másik féle sablonból

A kettő ötvözése – példa:

CSS:

```
h1 {font-size: 20px; color: #a00000;}
.fontos {color: #ff0000;}
h1.fontos {background: #ffcccc;}
```

HTML:

```
<h1 class=fontos>Ez a szöveg mindkét CSS sablon tulajdonságait felveszi</h1>
```

Kérdés: látható, hogy betűszín (color) 2x is meg van adva. Melyik lesz érvényes?

Válasz: ilyen esetben az egyéni sabloné lesz, tehát a .fontos színe

CSS szintaxis – avagy mit írunk a fájlba

```
szelektor {
        tulajdonság1: érték1;
        tul2: érték2;
}
```

A kettő ötvözése – példa folytatása:

CSS:

```
h1 {font-size: 20px; color: #a00000;}
.fontos {color: #ff0000;}
h1.fontos {background: #ffcccc;}
```

HTML:

```
<h1 class=fontos>Ez a szöveg mindkét CSS sablon tulajdonságait felveszi</h1>
```

Azt szeretnénk, hogy a h1-nél megadott szín legyen érvényes ilyen esetekben

→ `h1 {font-size: 20px; color: #a00000!important;}`

!important paranccsal kijelölhetünk egy fontossági sorrendet.

Azonosító alapú kiválasztás

Lényegében ugyanaz, mint ahol mi adunk nevet a szelektornak, de:

- Név elé nem PONT, hanem # jel kerül
- Ismerős lehet már ez: HTML – oldalon belüli hivatkozásnál (ugrás) volt már # jel!
  - Oda ugrott ahol id attribútumnak meg volt adva a # utáni érték (max. 1 hely/oldal)
- Ezért itt is csak 1x lehet használni, nem tetszőleges mennyiségben → kevesebbet tud
- Class attribútum helyett id-nél hivatkozunk rá (tagok pl: <p>, , <table>, <tr>, <td>…)
- Használata elkerülhető → mindig a tetszőleges nevűt használjuk

```
CSS:  #azonosito {color: green;}
HTML:<p id="azonosito">Szöveg</p>
```

CSS kitérő – mértékegységek

Láttuk, hogy pl. betűméret megadáshoz pixelt használunk.

A betűméret alapértelmezése 16px.

4-es betűméretnek kb. 16px-es vastag / 18px-es normál betűtípus felel meg

5-ös betűméretnek kb. 24px-es normál betűtipus felel meg

Lehetséges mértékegységek továbbá:

cm		
mm		
in	(inch	– 1 in = 2.54 cm)
pt	(point	– 1 pt = 1/72 inch)

CSS szintaxis – csoportosítás

Triviális csoportosítás:

```
h1 {font-size: 20px;}
h1 {color: white;}
h1 {font-style: normal;}
```

Ekvivalens: `h1 {font-size: 20px; color: white; font-style: normal;}`

Típusösszevonás:

```
h1 {font-size: 20px; color: #00000;}
h2 {font-size: 20px; color: #00000;}
h3 {font-size: 20px; color: #00000;}
```

Ekvivalens: `h1, h2, h3 {font-size: 20px; color: #00000;}`

Eltérő tulajdonságok csoportosítása
(nem mindenre jó, inkább ne használjunk ilyet, írjunk „tisztán” = írjunk ki mindent soronként)

```
h1 {font: bold 12pt helvetica}
```

Itt a betűre vonatkozó információkat írtuk le tömörebben: félkövér, 12pt méretű, helvetica betűtípus

Több tagú szelektorok csoportosítása

```
h1 em, h1 b, h2 b, h2 em {color: red}
```

Ez nem azt jelenti, hogy egy h2-es címsor pl félkövér lesz, hanem a piros szín ott lesz érvényes, ahol pl. h2-es címsor van, de a címsor szövege ráadásul HTML-ben formázottan vastag, tehát megtalálható a <b> tag

→ `Pl: <h2>Nem lesz piros <b>Piros lesz</b> Nem lesz</h2>`

Kódpéldák – egy css fájl részlete

Egyenlőre ne érdekeljen minket, hogy mit jelent,
ez csak szemléltetésként szolgál, hogy szokjuk a CSS fájlok leírását:

```
/* CSS Document */
Body {background-color: #eaeada;
          margin: 0px;}
a:link {color: darkblue;}
a:visited {color: darkblue;}
a:active {color: darkblue;}
a:hover {color: #cc0000;
          text-decoration: none;}

.cim {font-size: 24px;
          font-family: "Times New Roman", Times, serif;
          text-decoration: underline;
          text-align: center;}
.oldalcim {font-size: 18px;
          font-family: "Times New Roman", Times, serif;
          text-decoration: underline;}
.oldalcim_piros {font-size: 18px;
          font-family: "Times New Roman", Times, serif;
          color: #cc0000;
          text-decoration: underline;}
.table {text-align: center;
          width: 90%;
          border: 1px;
          border-color: #eaeada;}
.text_nagy {font-size: 18px;
          font-family: "Times New Roman", Times, serif;}
```

Kódpéldák – html részlet hozzá

```
<body>
<table width="100%" border="0">
  <tr>
    <td colspan="4" class="cim">CSS stíluslapok használata<br>
      <span align="center" class="oldalcim">
        Cascading Style Sheets = Egymásba ágyazott stíluslapok</span></td>
  </tr>
  <tr>
    <td colspan="4"> </td>
  </tr>
  <tr>
    <td colspan="4" class="text_nagy">FONTOS! Ezt csak akkor kezd tanulni, ha
      tisztában vagy a html nyelvvel (sztech/Html alapok menüpont)</td>
  </tr>
  <tr>
    <td colspan="4" class="text_nagy">Egy példa fájl-nak töltsd le az én egyik
      css-em: <a href="../../style.css">style.css</a></td>
  </tr>
  <tr>
    <td colspan="4"> </td>
  </tr>
  <tr>
    <td colspan="4" class="oldalcim">Mi is ez?</td>
  </tr>
```

Kódpéldák – body

```
body {
        background-color: skyblue;              - háttérszín szövegesen megadva
        background-image: url(../kep.jpg);      - háttérkép relatív útvonallal
        background-repeat: no-repeat;           - háttérkép ismétlés
        background-attachment: fixed;           - háttérkép rögzítés
        background-position: center center;      - háttérkép pozíció
}
```

Ekvivalens: `body { background: skyblue url(hatter.jpg) no-repeat fixed center center; }`

Részletesebben:

```
background-repeat: no-repeat / repeat / repeat-x / repeat-y
        no-repeat - egyszer teszi ki a háttérképet,
                        amit nem tud lefedni, ott a háttérszín lesz látható
        repeat   – vízszintesen és függőlegesen is ismétli a képet,
                        így az egész látható területet lefedi
        repeat-x – csak vízszintes ismétlés
        repeat-y – csak függőleges ismétlés

background-attachment: fixed / scroll
        fixed    – hátteret rögzítjük, ahogy görgetünk le, más részét látjuk
        scroll   – háttér a görgetéssel „jön le”, mindig ua. a részét látjuk

background-position: ____  ____
        első tagja lehet:         top / center / bottom
        második tagja:            left / center / right
```

Megjegyzés: háttere lehet pl td-nek (táblacella) is, oda is megadhatók, illetve általunk elnevezett sablonnak.

Kódpéldák – szöveg

A következő szelektorokba írhatók pl.: h1{}, p{}, table{}, tr{}, td{}, .valami{}…

```
color: #C0B0A0;                    - betűszín
letter-spacing: 10px;              - betűtávolság
word-spacing: 20px;                - szavak közti távolság
line-height: 35px;                 - sormagasság
vertical-align: top;               - függőleges igazítás
        sub/super/baseline/text-top/text-bottom/middle/bottom is lehet
text-align: left;                  - vízszintes igazítás
        center/right/justify is lehet
text-decoration: underline;        - dekoráció
        none (nincs) / overline (felülhúzás) / underline (aláhúzás)
        line-through (keresztül) / blink (villog)
text-indent: 40px;                 - bekezdés (<p> tagra)
text-transform: uppercase;         - szöveg tanszformáció
        none (nincs) / capitazize (kezdőbetű nagy) /
        uppercase (csupa nagy betű) / lowercase (csupa kicsi)
```

Egyéb pl.:

```
p: first-line {font-variant: small-caps;}          - nagybetűs első sor
p: first-letter {font-size: 200%; float: left;}    - nagy kezdőbetű
```

Kódpéldák – keretek

A következő szelektorokba írhatók pl.: p{}, table{}, tr{}, td{}, .valami{}…

```
border: 2px solid blue;            - szegély (méret, stílus, szín)
        Méret:            px-ben, tetszőlegesen
        Stílusok:         solid    (sima vonal)
                          dotted   (pontozott vonal)
                          dashed   (szaggatott vonal)
                          double   (dupla vonal)
                          groove   (süllyesztett vonal, 3D hatással)
                          ridge    (domború vonal, 3D hatással)
                          inset    (beékelt stílusú vonal)
                          outset   (kiemelt stílusú vonal)
        Szín:             hexadecimálisan vagy szövegesen

border-width: 2px;                 - szegélyvastagság

border-left-width: 2px;            - csak a baloldali szegély változtatása
        többi oldalra is lehet: left, right, top, bottom

border-style: solid;               - szegélystílus

border-color: green;               - szegély színe
```

Megjegyzés: a legelső border: 2px solid blue igazából a border-width, border-style és border-color összevonása.

Egyéb pl.:

```
p {     border-left: 1px solid red;                - külön minden oldalra
        border-top: 3px double blue;
        border-right: 3px dotted green;
        border-bottom: 3px dashed black; }
```

Kódpéldák – margó, kitöltés

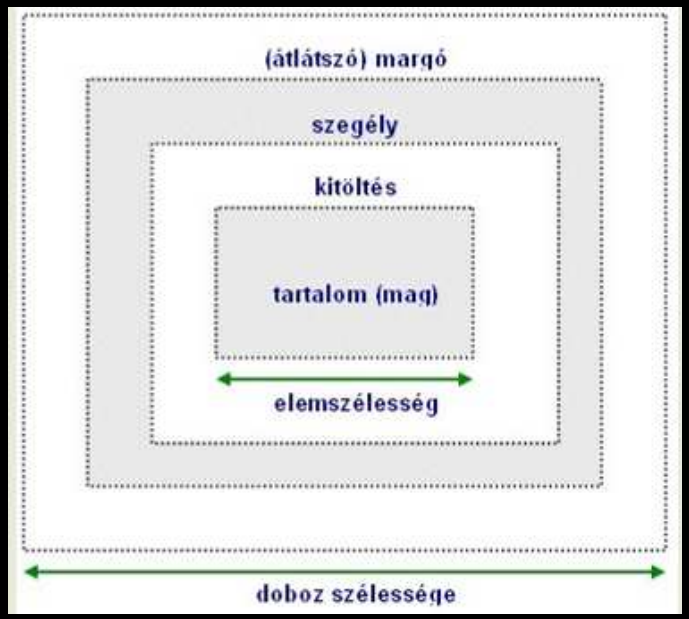
Margók – következő szelektorokba írhatók pl.: body{}, p{}…

```
margin: 3px;                - margó (minden oldal)
margin: 3px 5px;            - margó (függőleges / vízszintes külön megadása)
        nem biztos, hogy minden böngésző támogatja
margin-left: 3px;           - margó (oldalanként megadható – lásd keretek)
```

Kitöltés – a következő szelektorokba írhatók pl.: body{}, p{}, table{}, tr{}, td{}…

```
padding: 3px;               - kitöltés, hézag (minden oldal)
margin-left: 3px;           - margó (oldalanként megadható – lásd keretek)

padding: 2px; padding-left: 3px;
                            - minden oldal 2px, kivéve a bal, ami 3px
```



A CSS egy egyszerű, dobozszerű formázási modellt használ, ahol minden elemformázás eredménye egy, vagy több négyszögletes dobozként képzelhető el. Minden doboznak van egy 'magja', az őt körülvevő 'kitöltéssel' (padding), szegéllyel (border) és margóval (margin).

A kitöltés területének a háttérszíne mindig megegyezik a mag (elem) háttérszínével.

A margó mindig átlátszó.

Kódpéldák – linkek, látszólagos osztályok

Látszólagos elemekkel már találkozhattunk a szövegformázásnál is. (first-letter / first-line)
Ezeknél a tag szelektor után KETTŐSPONTot írunk:

```
a {color: #0000FF; text-decoration: underline;}     - link alapstílus
        emlékezzük… link <a href=>link</a>
a:link {text-decoration: none;}                     - még nem látott link
a:active {font-size: 20px;}                         - aktív link
a:visited {font-style: italic;}                     - már látott link
a:hover {color: cc0000;}
        Ha az egeret fölévisszük, ilyenre változik
        Itt csak a módosulást kell megadni, vagyis ez pl aláhúzott is lesz
```

Lehet használni más tagokkal együtt:

```
.minta a {}
        Akkor lesz érvényes, ha linkről van szó és hivatkozunk a minta sablonra
.minta a:hover {}
        Akkor lesz érvényes, ha linkről van szó és hivatkozunk
                a minta sablonra és a link fölé visszük az egeret

a:link img {border: solid blue;}            - hivatkozással ellátott képeknél
```

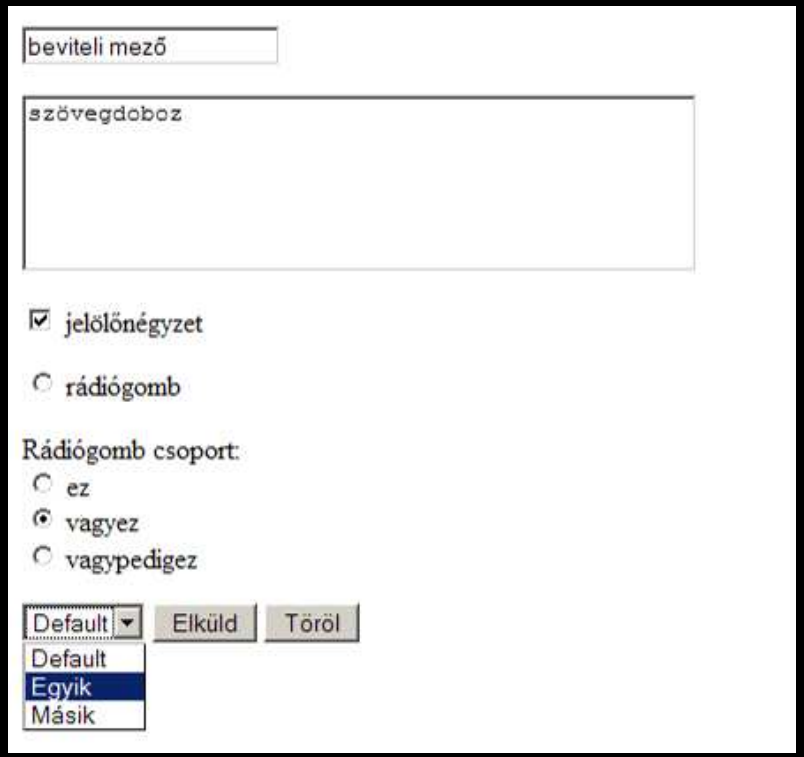
Kódpéldák – űrlapok

Űrlapokkal nem foglalkozunk HTML-nél sem, úgyhogy először nézzük azt:

```
<form name="form1" method="post" action="">        - űrlap kezdete
        IDE JÖNNEK AZ ŰRLAP ELEMEI
</form>                                            - űrlap vége
```

`method, action` egyenlőre lényegtelen → lásd PHP, MySQL (3.ea)

Űrlapelemek:



Beviteli mező:	rövid szöveges adat számára
Szövegdoboz:	nagyobb adatnak
Jelölőnégyzet:	igen/nem típusú választás
Rádiógomb:	igen/nem típusú választás
Rádiógomb-csoport:	több közül 1 kiválasztása
Legördülő lista:	több közül 1 kiválasztása
Submit:	küldés gomb, űrlap elküldése
Reset:	kitöltött mezők törlése

Jelölőnégyzet-rádiógomb különbség:
a rádiógombot ha kijelöltük akkor nem vonható vissza, csak a teljes űrlap törlésével

Rádiógomb-csoport és legördülő lista különbség:
kb semmi, legördülő ajánlott pl ha sok és rendezett lehetőség van (pl. születési évszám kiválasztása – listában ekkor évszámok 1900-2010-ig)

Kódpéldák – űrlapok

Űrlapelemek fontosabb attribűtumai:

type="text" / password / hidden / checkbox / radio
- Ezzel adjuk meg a mező típusát, de nem minden esetben
- → ellenpélda: <textarea> = szövegmező, erre külön tag van

name="valami"
- Tetszőleges elnevezés, erre majd PHP-nál lesz szükség
- Ezzel azonosítjuk, hogy melyik űrlapelemből jött az adott adat
- Emiatt egy űrlapon belül mindennek a neve legyen KÜLÖNBÖZŐ!!!
- KIVÉTEL: egy rádiógomb-csoporton belül az egyes lehetőségek

value="valamiertek"
- Tetszőleges érték, erre majd PHP-nál lesz szükség
- Ezt az értéket küldjük el az adott űrlapelemből a küldés gomb hatására
- Nem kell annak lennie, ami pl. jelölőnégyzet melletti szöveg
  - Pl: jelölőnégyzet mellett ez áll: *„elfogadom a szabályokat”*
  - Értéknek elég azt megadni, hogy „1”
  - Ha bejelöli akkor 1-et kapunk, ha nem akkor „” üres stringet
- Rádiógomb-csoportnál értelemszerűen minden választási lehetőségnek
  - különböző értéket kell adni, mert ha van két azonos értékű,
  - akkor nem tudhatjuk, hogy abból melyik is lett bejelölve

Kódpéldák – űrlapok

Beviteli mező:

```
<input type="text" name="valami">
          maxlength="5" attribútum → max. 5 karakter írható bele
          type="password" → a beírt karakterek helyett csak pöttyök lesznek (nem látszanak)
          <input>-nak nincs lezáró tagja
```

Szövegdoboz:

```
<textarea name="valami"></textarea>
          rows="10" cols="50" attribútumok → magasság, szélesség megadása
```

Jelölőnégyzet:

```
<label><input type="checkbox" name="valami" value="1" checked>Szöveg</label>
          ha alapértelmezetten bejelöltnek szeretnénk, írjuk be az input-ba, hogy checked
          mivel az inputot nem kell lezárni, így a <label> (címke) segítségével tudjuk
                    jelezni, hogy a Szöveg az a jelölőnégyzethez tartpozik
```

Rádiógomb:

```
<input type="radio" name="valami" value="radiobutton">
```

Rádiógomb csoport (több lehetőségből csak 1 választható – fontos: name mindenhol ugyanaz!)

```
<label><input type="radio" name="ugyanazok" value="elso">1. válasz</label><br>
<label><input type="radio" name="ugyanazok" value="masodik">2. válasz </label>
```

Legördülő lista

```
<select name="select">                                  - lista kezdete
<option selected>kék</option>                           - lista első eleme
<option>zöld</option>                                   - lista második eleme
</select>                                               - lista vége
```

Megjegyzés: ahol selected van, az az alapértelmezett, vagyis ha nem választunk az marad

Küldés gomb:

```
<input type="submit" name="Submit1" value="Mehet">
```

Törlés gomb (kitöltött űrlapmezők tartalmának törlése, alaphelyzetbe állít):

```
<input type="reset" name="Reset1" value="Töröl">
```

Kódpéldák – űrlapok

Mostmár jöhet a CSS:

```
input {                                 - minden input tagra vonatkozó stílus
        font-size: 12px;
        border: 1px solid #000000;
        background: #ffffff;
}
input:hover {                           - ha fölé visszük az egeret
        border: 1px solid #0000FF;
}
input:active {                          - ha aktív elem
        font-style: bold;
}
input:focus {                           - ha kiválasztjuk (belekattintunk)
        background: #dddddd;
}
```

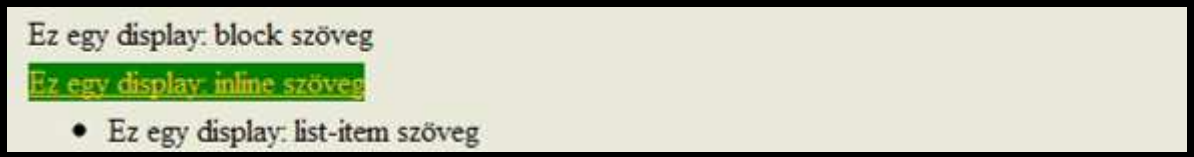
Ezek segítségével pl. a Küldés gomb is tetszőlegesen formázható

Kódpéldák – egyéb

```
display: block / inline / list-item
        megjelenítés szabályozása olyan szövegeknél, amik linkek
```

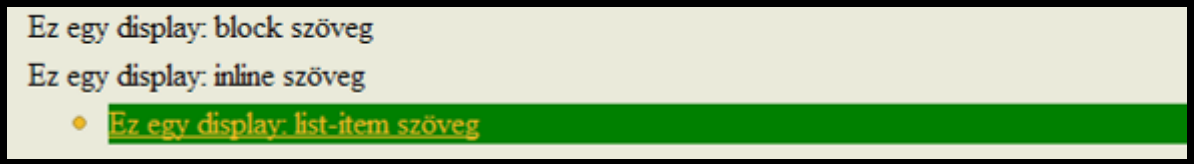
block:
Ha fölé visszük az egeret, akkor a teljes dobozt kijelöli (lásd dobozszerű dormázásmodell). Itt pl egy táblázatcellában van a szöveg, így az egész cellát. Ráadásul az egész cellaterület link lesz, nem csak a szöveg.

inline:



Csak a szöveg fölött érvényes a link és a doboznak csak a szövegig tartó része lesz kijelölve.

list-item: teljes doboz, és a listajel is kijelölve (nem minden böngésző támogatja)



Következő dián a három eset használata: itt a display tulajdonságot csak a változáshoz kell megadni
De lehet persze az is, hogy alapból is a doboz háttere valami más szín és akkor oda is be kell írni.

Kódpéldák – egyéb (használata)

```
<head>
<title>Paksy Patrik honlapja</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<link href="../../style.css" rel="stylesheet" type="text/css">
<style type="text/css">
.egyeb1 a {
          color: #000000;
          text-decoration: none;}
.egyeb1 a:hover {
          display: block;
          background-color: green;
          text-decoration: underline;
          color: f2bc26;}
.egyeb2 a {
          color: #000000;
          text-decoration: none;}
.egyeb2 a:hover {
          display: inline;
          background-color: green;
          text-decoration: underline;
          color: f2bc26;}
.egyeb3 a {
          color: #000000;
          text-decoration: none;}
.egyeb3 a:hover {
          display: list-item;
          background-color: green;
          text-decoration: underline;
          color: f2bc26;}
</style>
</head>
```

Vége