

# Webprogramozás

## CSS kijelölők, kombinátorok, specificity és EMMET

Rostagni Csaba

2024. szeptember 2.

# Tartalom

## 1 CSS Kijelölők (CSS selectors)

- Univerzális kijelölő
- Típus kijelölő
- Azonosító kijelölő
- Osztály kijelölő
- Attribútum kijelölők

## 2 Kombinátorok

## 3 Pszeudo osztályok

- Linkekhez
- Felhasználói műveletekhez
- Sorrend
- Összetett felhasználás
- Pozíció szerint

## 4 Emmet

- Gyorskódok

## 5 CSS Specificity és rangsorolás

• Példák

# Univerzális kijelölő - universal selector

- CSS-ben a \* (csillag karakter) jelöli.
- Tetszőleges HTML elemre vonatkozhat

```
* {color: blue}
```

CSS

```
<div>Ez kék!</div>  
<p>Ez egy kék bekezdés</p>
```

HTML

Ez kék!  
Ez egy kék bekezdés

Eredmény

# Típus kijelölő - Type selector

- CSS-ben a HTML elem típusát kell megadni.
- A w3schools Element Selector néven emlegeti, így **Elemkijelölőként** is találkozhatunk vele.
- Nem konkrét elemet, hanem az összes adott típusú elemet jelöli ki.

```
div {color: blue}
```

CSS

```
<p>Sima bekezdés</p>  
<div>Ez kék!</div>
```

HTML

Sima bekezdés  
Ez kék!

Eredmény

# Azonosító kijelölő - ID selector

- CSS-ben # (kettőskereszt) jelöli.
- HTML-ben id attribútumként kell megadni.
- Csak egy szerepelhet belőle egy adott oldalon.

```
#pelda {color: blue}
```

CSS

```
<p>Sima bekezdés</p>
```

```
<p id="pelda">Azonosítóval ellátott bekezdés</p>
```

HTML

Sima bekezdés

Azonosítóval ellátott bekezdés

Eredmény

# Osztály kijelölő - Class selector

- CSS-ben . (pont) jelöli.
- HTMLben **class** attribútumként kell megadni.
- Egy osztályt több elem is megkaphat az oldalon.

```
.pelda {color: blue}
```

CSS

```
<p>Sima bekezdés</p>
```

```
<p class="pelda">Ez egy kék bekezdés!</p>
```

HTML

Sima bekezdés

Ez egy kék bekezdés!

Eredmény

# Osztály kijelölő - Class selector

- Egy osztályt több elem is megkaphat az oldalon.

```
.x {color: blue}
```

CSS

```
<p>Sima bekezdés</p>  
<div class="x">Ez kék!</div>  
<p class="x">Ez egy kék bekezdés!</p>
```

HTML

Sima bekezdés  
Ez kék!  
Ez egy kék bekezdés!

Eredmény

# Osztály kijelölő - Class selector

- Egy elem egyszerre több osztályba is sorolható
- Ilyenkor a class attribútumon belül szóközzel választjuk el az osztályok nevét.

CSS

```
.x {color: blue}  
.y {font-weight: bold;}
```

HTML

```
<p class="x">Ez egy kék bekezdés!</p>  
<p class="y">Ez egy félkövér bekezdés!</p>  
<p class="x y">Ez kék és félkövér is!</p>
```

Eredmény

Ez egy kék bekezdés!  
Ez egy félkövér bekezdés!  
Ez kék és félkövér is!



# Attribútum kijelölők - Attribute selectors

- CSS-ben `[]` jelöli.
- HTML-ben a megfelelő attribútumot kell kitölteni.

```
[lang=hu] {color: blue}
```

CSS

```
<p lang="en">Angol szöveg</p>  
<p lang="hu">Magyar szöveg</p>
```

HTML

Angol szöveg  
Magyar szöveg

Eredmény

# Tartalom

## 1 CSS Kijelölők (CSS selectors)

- Univerzális kijelölő
- Típus kijelölő
- Azonosító kijelölő
- Osztály kijelölő
- Attribútum kijelölők

## 2 Kombinátorok

## 3 Pszeudo osztályok

- Linkekhez
- Felhasználói műveletekhez
- Sorrend
- Összetett felhasználás
- Pozíció szerint

## 4 Emmet

- Gyorskódok

## 5 CSS Specificity és rangsorolás

- Példák

# Leszármazott - Descendant (□)

- CSS-ben whitespace (pl.: szóköz) karakterrel jelöljük.
- Olyan elem, ami egy másik elemen belül található.
- Tetszőleges mélységben lehet!

```
div b {color: blue}
```

CSS

```
<div>  
  <b>Ez kék!</b>  
  <p>Ez egy <b>bekezdésben</b> van</p>  
</div>  
<b>Ez csak félkövér</b>
```

HTML

Ez kék!  
Ez egy bekezdésben van  
Ez csak félkövér

Eredmény

# Gyermek - Child (>)

- CSS-ben > karakterrel jelöljük.
- Olyan elem, ami egy másik elemen belül található **közvetlenül**.

```
div > b {color: blue}
```

CSS

```
<div>  
  <b>Ez kék!</b>  
  <p>Ez egy <b>bekezdésben</b> van</p>  
</div>  
<b>Ez csak félkövér</b>
```

HTML

Ez kék!  
Ez egy bekezdésben van  
Ez csak félkövér

Eredmény

# Közvetlen testvér - Next-sibling (+)

- CSS-ben + karakterrel jelöljük.
- Olyan elem, ami egy másik elemet követ **közvetlenül** és egy szintet van vele (testvérek).

```
.a + p {color: blue}
```

CSS

```
<p class="a">Első bekezdés</p>  
<p class="b">Második bekezdés</p>  
<p class="c">Harmadik bekezdés</p>
```

HTML

Első bekezdés  
Második bekezdés!  
Harmadik bekezdés

Eredmény

# Következő testvér(ek) - Subsequent-sibling (~)

- CSS-ben ~ karakterrel jelöljük.
- Olyan elem, ami egy másik elemet követ és egy szinten van vele (testvérek). Több elemre is lehet igaz!

```
.a ~ p {color: blue}
```

CSS

```
<p class="a">Első bekezdés</p>  
<p class="b">Második bekezdés</p>  
<p class="c">Harmadik bekezdés</p>
```

HTML

Első bekezdés  
Második bekezdés!  
Harmadik bekezdés

Eredmény

# Tartalom

## 1 CSS Kijelölők (CSS selectors)

- Univerzális kijelölő
- Típus kijelölő
- Azonosító kijelölő
- Osztály kijelölő
- Attribútum kijelölők

## 2 Kombinátorok

## 3 Pseudo osztályok

- Linkekhez
- Felhasználói műveletekhez
- Sorrend
- Összetett felhasználás
- Pozíció szerint

## 4 Emmet

- Gyorskódok

## 5 CSS Specificity és rangsorolás

- Példák

# Pseudo osztályok - Pseudo-classes (:)

- A **pseudo** szó jelentése: nem igazi, hamis állítólagos
- Jelentése CSS-ben:
  - Olyan "osztály", amit nem mi határozzunk meg.
  - Nem mi, hanem a böngésző szabja meg melyik elemre vonatkozik.  
"Magától jön létre."
- Jelölése kettőspont.
- Gyakori használata linkek formázása.
- Egy elemre egyszerre több pseudo osztály is megfelelhet.
- A böngészőben kikényszeríthetjük az állapotokat teszteléskor.



# :link

- A még meg nem látogatott linkeket formázza.
- Olyan `<a>` tag-et, aminek a `href` attribútuma ki van töltve.

```
a:link {color: green}
```

CSS

```
<a href="example.com">link</a>  
<a name="x">könyvjelző</p>
```

HTML

link  
könyvjelző

Eredmény

## :visited

- A már látogatott linkeket formázza.
- Olyan `<a>` tag-et, aminek a `href` attribútuma ki van töltve.
- A `:link` és a `:visited` kölcsönösen kizárja egymást.

CSS

```
a:link {color: green}
a:visited {color: red}
```

HTML

```
<a href="example.com/uj.html">Ezt még nem láttam</a>
<a href="example.com/regi.html">Ezt már láttam</a>
```

Eredmény

Ezt még nem láttam  
Ezt már láttam

# :hover

- Akkor aktiválódik, amikor a mutató eszközt (pl.: egér) az adott elem fölé visszük.
- Telefonon is elérhető ez az állapot (hosszan nyomva).

```
a:hover {color: red}  
p:hover {color: green}
```

CSS

```
<a href="example.com">Ez egy link</a>  
<p>Ez egy bekezdés</p>
```

HTML

Az egér a link felett áll

Ez egy link  
Ez egy bekezdés

Az egér a bekezdés felett áll

Ez egy link  
Ez egy bekezdés

# :focus

- Akkor aktiválódik, amikor az adott elem kerül a fókuszba.
- Leggyakrabban űrlap elemeknél találkozhatunk vele, amikor a tab lenyomásával a következő beviteli mezőre visszük a fókuszt.

CSS

```
input:focus {outline: 1px solid blue;}
```

HTML

```
<input type="text" name="nev" placeholder="Név">  
<input type="password" name="jelszo" placeholder="Jelszó">
```

Eredmény

Név

I

## :active

- Akkor érvényes, amikor az adott elemet aktiváljuk, pl az egér bal gombját lenyomjuk az elemen állva.
- Nem csak linkekre alkalmazható!


```
a:active {color: red}
p:active {color: green}
```

CSS


```
<a href="example.com">Ez egy link</a>
<p>Ez egy bekezdés</p>
```

HTML

Az egér éppen kattint

Ez egy link   
Ez egy bekezdés

Az egér éppen kattint

Ez egy link  
Ez egy bekezdés 

# A sorrend lényeges!

- Egy elemre egyszerre több pszedo osztály is passzolhat.
- Ilyenkor a később írt kód felülírja a korábbi.

CSS logikai hiba!

```
a:active {color: yellow;}  
a:focus  {color: limegreen;}  
a:hover  {color: red;}  
a:visited {color: green;}  
a:link    {color: blue;}
```

HTML

```
<a href="http://example.com">Ez a link nem mutatja az  
↪ állapotváltozást</a>
```

Eredmény

Ez a link nem mutatja az állapotváltozást👍

- A `:hover`, `:focus`, `:active` sose érvényesül.

# A helyes sorrend!

- Egy elemre egyszerre több pszedo osztály is passzolhat.
- Ilyenkor a később írt kód felülírja a korábbi.

```
a:link      {color: blue;}  
a:visited   {color: green;}  
a:hover     {color: red;}  
a:focus     {color: limegreen;}  
a:active    {color: yellow;}
```

CSS

```
<a href="http://example.com">Ez egy link</a>
```

HTML

- Ha az egeret fölé visszük, akkor piros lesz.
- Ugyanakkor ha lenyomjuk a bal egér gombot sárgára vált.
- Ha fókuszot kap, akkor limegreen színt kap.
- A fenti CSS mindig az aktuális színnel jelenik meg.

# Pseudo osztályok kombinálása

- Egy adott elemre egyszerre több pseudo osztály is alkalmazható.

```
a:link:active {color: green}
```

CSS

- Csak a még meg nem látogatott linkek kapnak zöld színt, ha aktiváljuk az elemet.



# Leszármazott kijelölése

- Összetett kijelölő esetén a szülő elem is megkaphatja a pszeudo osztályt, mégis a leszármazottjára hivatkozunk

```
p: hover b { color: blue }
```

CSS

```
<p>  
  Ez itt  
  <b>kék</b>  
  lesz!  
</p>
```

HTML

Ez itt **kék** lesz

Az egér a bekezdés felett áll

# :first-child

- Az adott elem az első gyermek eleme egy másik elemnek.

CSS

```
p:first-child {color: blue}
```

HTML

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

Eredmény

Első bekezdés

Második bekezdés

Harmadik bekezdés

# :first-child

```
p:first-child {color: blue}
```

Itt nincs hatása

HTML

```
<div>
  <h1>Címsor 1</h1>
  <p>Első bekezdés</p>
  <p>Második bekezdés</p>
</div>
```

Eredmény

## Címsor 1

Első bekezdés

Második bekezdés

- A `<div>` elem első gyermek eleme a `<h1>` elem.
- Nincs olyan `<p>` elem, ami első gyermekelem lenne.

# :first-of-type

- Az adott típusú elem első előfordulása.

CSS

```
p:first-of-type {color: blue}
```

HTML

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

Eredmény

Első bekezdés

Második bekezdés

Harmadik bekezdés

# :first-of-type

CSS

```
p:first-of-type {color: blue}
```

HTML

```
<div>
  <h1>Címsor 1</h1>
  <p>Első bekezdés</p>
  <p>Második bekezdés</p>
</div>
```

Eredmény

# Címsor 1

Első bekezdés

Második bekezdés

- A `<div>` elem első gyermek eleme a `<h1>` elem.
- Mivel tartalmaz `<p>` elemeket így azok közül választja ki az elsőt.

# :nth-child(n)

- Az adott elem az n-edik gyermek eleme egy másik elemnek.
- Megadhatjuk számmal
  - Első: `li:nth-child(1)`
  - Ötödik: `li:nth-child(5)`
- Megadhatjuk szövegesen
  - páros: `li:nth-child(even)`
  - páratlan: `li:nth-child(odd)`
- Képlettel, ahol n helyére nem negatív egész számok kerülhetnek (0, 1, 2, ...)
  - páros: `li:nth-child(2n)`
  - páratlan: `li:nth-child(2n+1)`
  - minden harmadik: `li:nth-child(3n)`

## :nth-child() páros példa

CSS

```
b:nth-child(even) {color: blue;}  
b:nth-child(2n)   {text-decoration:underline;}
```

HTML

```
<p>  
  <b>1</b><b>2</b><b>3</b>  
  <b>4</b><b>5</b><b>6</b>  
</p>
```

Eredmény

1 2 3 4 5 6

$n = 0$  esetén:  $2 \cdot 0 = 0$

$n = 1$  esetén:  $2 \cdot 1 = 2$

$n = 2$  esetén:  $2 \cdot 2 = 4$

$n = 3$  esetén:  $2 \cdot 3 = 6$

# :nth-child() páratlan példa

CSS

```
b:nth-child(odd) {color: blue;}  
b:nth-child(2n+1) {text-decoration:underline;}
```

HTML

```
<p>  
  <b>1</b><b>2</b><b>3</b>  
  <b>4</b><b>5</b><b>6</b>  
</p>
```

Eredmény

1 2 3 4 5 6

$n = 0$  esetén:  $2 \cdot 0 + 1 = 1$

$n = 1$  esetén:  $2 \cdot 1 + 1 = 3$

$n = 2$  esetén:  $2 \cdot 2 + 1 = 5$

$n = 3$  esetén:  $2 \cdot 3 + 1 = 7$



# :nth-child() minden harmadik példa

CSS

```
b:nth-child(3n) {color: blue;}
```

HTML

```
<p>  
  <b>1</b><b>2</b><b>3</b>  
  <b>4</b><b>5</b><b>6</b>  
</p>
```

Eredmény

1 2 3 4 5 6

$n = 0$  esetén:  $3 \cdot 0 = 0$

$n = 1$  esetén:  $3 \cdot 1 = 3$

$n = 2$  esetén:  $3 \cdot 2 = 6$

$n = 3$  esetén:  $3 \cdot 3 = 9$

## :nth-child() speciális példa

CSS

```
p :nth-child(odd) {color: blue;}  
p *:nth-child(odd) {text-decoration:underline;}
```

HTML

```
<p>  
  <b>1</b><b>2</b><b>3</b>  
  <b>4</b><b>5</b><b>6</b>  
</p>
```

Eredmény

1 2 3 4 5 6

- A `<p>` elem gyermek elemeire hivatkozunk.
- A `p` és a `:nth-child(odd)` között szóköz szerepel!

# :nth-of-type(n)

- Az elem az n-edik adott típusú gyermek eleme egy másik elemnek.
- Megadhatjuk számmal
  - Első: `li:nth-of-type(1)`
  - Ötödik: `li:nth-of-type(5)`
- Megadhatjuk szövegesen
  - páros: `li:nth-of-type(even)`
  - páratlan: `li:nth-of-type(odd)`
- Képlettel, ahol n helyére nem negatív egész számok kerülhetnek (0, 1, 2, ...)
  - páros: `li:nth-of-type(2n)`
  - páratlan: `li:nth-of-type(2n+1)`
  - minden harmadik: `li:nth-of-type(3n)`

## :nth-of-type() páros példa

CSS

```
b:nth-of-type(even) {color: blue;}  
b:nth-of-type(2n)   {text-decoration:underline;}
```

HTML

```
<p>  
  <b>A</b><i>B</i><b>C</b>  
  <i>D</i><b>E</b><i>F</i>  
</p>
```

Eredmény

A B C D E F

- A **C** a bekezdés harmadik eleme.
- Ugyanakkor a második **<b>** elem.

# Gyermekek

Első / utolsó

<code>:first-child</code>	<code>:first-of-type</code>
<code>:last-child</code>	<code>:last-of-type</code>

Első n. / utolsó n.

<code>:nth-child(n)</code>	<code>:nth-of-type(n)</code>
<code>:nth-last-child(n)</code>	<code>:nth-last-of-type(n)</code>

Egyetlen

<code>:only-child</code>	<code>:only-of-type</code>
--------------------------	----------------------------

# Tartalom

## 1 CSS Kijelölők (CSS selectors)

- Univerzális kijelölő
- Típus kijelölő
- Azonosító kijelölő
- Osztály kijelölő
- Attribútum kijelölők

## 2 Kombinátorok

## 3 Pszeudo osztályok

- Linkekhez
- Felhasználói műveletekhez
- Sorrend
- Összetett felhasználás
- Pozíció szerint

## 4 Emmet

- Gyorskódok

## 5 CSS Specificity és rangsorolás

- Példák

# Emmet

- A fejlesztés sebességét növelő plugin.
- VS Code-ba integrált.
- Egyéb szerkesztőbe külön telepíthető.
- A szerkesztett fájl típusát be kell állítani!
- A **.html** és **.css** kiterjesztéssel elmentett fájlt felismeri.
- A tab aktiválja a kódkiegészítést, csak a végén kell állni!
- <https://docs.emmet.io/cheat-sheet/>

# HTML kód generálása

html:5

Emmet

HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
      ↪ initial-scale=1.0">
    <title>Document</title>
  </head>
  <body>
  </body>
</html>
```



# HTML kód generálása (rövid)

```
html
```

Emmet

```
<html></html>
```

HTML

# HTML kód generálása (legrövidebb)

!

Emmet

HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
      ↪ initial-scale=1.0">
    <title>Document</title>
  </head>
  <body>
  </body>
</html>
```

# Egyszerű HTML elem

p

Emmet

<p></p>

HTML

h1

Emmet

<h1></h1>

HTML

# Egyszerű HTML elem attribútumokkal

- Bizonyos elemeknél a leggyakrabban használt attribútumokat is megjeleníti.

img

Emmet

```
<img src="" alt="">
```

HTML

a

Emmet

```
<a href=""></a>
```

HTML

# Azonosító kijelölő

#x

Emmet

<div id="x"></div>

HTML

- Alapértelmezetten <div> elemet fog használni.

p#x

Emmet

<p id="x"></p>

HTML

- Ha megadjuk, akkor a megadott elemet kapjuk.
- NINCS szóköz a # előtt, sem utána!

# Osztálykijelölő

Emmet

`.a`

HTML

```
<div class="a"></div>
```

- Alapértelmezetten `<div>` elemet fog használni.

Emmet

`p.a`

HTML

```
<p class="a"></p>
```

- Ha megadjuk, akkor a megadott elemet kapjuk.
- NINCS szóköz a pont előtt, sem utána!

# Azonosító és osztálykijelölő együtt

#x.a

Emmet

<div id="x" class="a"></div>

HTML

- Alapértelmezetten <div> elemet fog használni.

p#x.a

Emmet

<p id="x" class="a"></p>

HTML

- Ha megadjuk, akkor a megadott elemet kapjuk.
- Nincsenek szóközök!

# Tartalom

## 1 CSS Kijelölők (CSS selectors)

- Univerzális kijelölő
- Típus kijelölő
- Azonosító kijelölő
- Osztály kijelölő
- Attribútum kijelölők

## 2 Kombinátorok

## 3 Pszeudo osztályok

- Linkekhez
- Felhasználói műveletekhez
- Sorrend
- Összetett felhasználás
- Pozíció szerint

## 4 Emmet

- Gyorskódok

## 5 CSS Specificity és rangsorolás

- Példák



# Különböző helyeken elhelyezett stílus definíciók

style.css

```
div {border: 3px solid blue;}
```

HTML

```
<head>
  <link rel="stylesheet" href="style.css">
  <style>
    div {border-style: dotted;}
  </style>
</head>
<body>
  <div style="border-color:red"></div>
</body>
```

- border-width: 3px; border-style: dotted; border-color: red;
- "Minél közelebb áll a formázandó elemhez..."

# Példa 1.

HTML

```
<head>
  <style>
    p {color: red;}
    p {color: green;}
  </style>
</head>
<body>
  <p id="x" class="a b c">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? zöld
- A két kijelölő specifikussága azonos.
- A zöld sorrendben később van, "közelebb áll" az elemhez.

## Példa 2.

HTML

```
<head>
  <style>
    p {color: white;}
    p#x {color: blue;}
  </style>
</head>
<body>
  <p id="x" class="a b c">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? kék
- Az azonosító kijelölő specifikusabb
- A kék ugyan sorrendben később van, "közelebb áll" a szöveghez, de ez most nem számít

## Példa 3.

HTML

```
<head>
  <style>
    p#x {color: blue;}
    p {color: white;}
  </style>
</head>
<body>
  <p id="x" class="a b c">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? kék
- Az azonosító kijelölő specifikusabb
- Itt is a sorrendet a specifikusság felülbírálja

## Példa 4.

HTML

```
<head>
  <style>
    p#x {color: orange;}
    p.a {color: black;}
  </style>
</head>
<body>
  <p id="x" class="a b c">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? narancssárga
- Az azonosító kijelölő specifikusabb
- Itt is a sorrendet a specifikusság felülbírálja

## Példa 5.

HTML

```
<head>
  <style>
    p.a {color: green;}
    p.b {color: white;}
  </style>
</head>
<body>
  <p id="x" class="a b c">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? fehér
- A két kijelölő specifikussága azonos.
- A fehér sorrendben később van, "közelebb áll az elemhez".

## Példa 6.

HTML

```
<head>
  <style>
    p.b {color: white;}
    p.a {color: green;}
  </style>
</head>
<body>
  <p id="x" class="a b c">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? zöld
- A két kijelölő specifikussága azonos.
- A zöld "közelebb áll" az elemhez.
- A class-on belül a sorrend nem számít.

## Példa 7.

HTML

```
<head>
  <style>
    p.a.b.c {color: darkblue;}
    p.a {color: lightgreen;}
  </style>
</head>
<body>
  <p id="x" class="a b c">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? sötétkék
- A több class specifikusabb.
- A világoszöld közelebb áll az elemhez, de a specifikusabb nyer.
- A css-ben nincs szóköz a class-ok között!



## Példa 8.

HTML

```
<head>
  <style>
    p#x {color: gold;}
    p.a.b.c.d.e.f.g.h.i.j.k {color: purple;}
  </style>
</head>
<body>
  <p id="x" class="a b c d e f g h i j k">Szöveg</p>
</body>
```

- Milyen színű lesz a szöveg? gold
- Egy azonosító kijelölő specifikusabb, mint akárhány elem vagy osztály kijelölő. (kivéve ha bugos a böngésző)
- A lila ugyan sorrendben később jön, de a specifikusabb nyer.

# CSS kódok összeillesztése

```
body {background-color: red}
```

a.css

```
body {background-color: green}
```

b.css

```
@import "b.css"  
body {background-color: blue}
```

c.css

```
<head>  
  <link rel="stylesheet" href="a.css">  
  <link rel="stylesheet" href="c.css">  
  <style>  
    body {background-color:yellow}  
  </style>  
</head>  
<body style="background-color: black">  
</body>
```

HTML

# CSS kódok összeillesztése

Eredmény

```
body {background-color: red}  
body {background-color: green}  
body {background-color: blue}  
body {background-color: yellow}  
body {background-color: black}
```

- Minden szabály eredete ugyanaz a szerző css kódjai.
- Minden szabálynak azonos a fontossága, ugyanis egyik sem !important.
- Minden elem ugyanannyira specifikus.
- **Az utolsó szó jogán** a háttér színe fekete lesz.