CSS (Cascading Style Sheets)

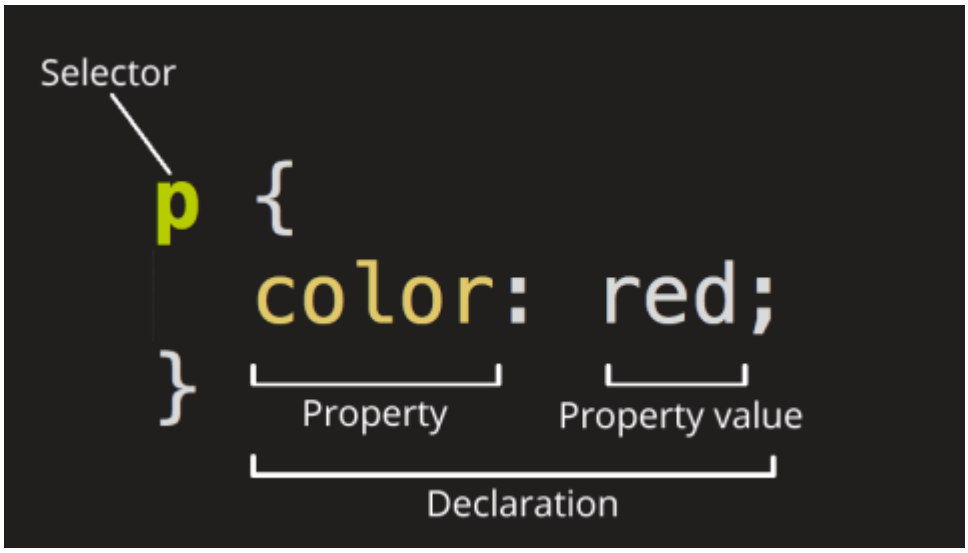
Mi a CSS?

A CSS (Cascading Style Sheets) egy stílusok megadására alkalmas nyelv, amelyet weboldalak formázására használnak. Segítségével lehetőség van a HTML elemek megjelenését, elrendezését és formázását meghatározni.

Míg az HTML leírja a weboldal szerkezetét és tartalmát, a CSS lehetővé teszi a fejlesztők számára, hogy stílusokat alkalmazzanak, mint például színek, betűtípusok, térközök és elhelyezés, hogy vizuálisan vonzó és felhasználóbarát felületeket hozzanak létre.

Selectorok és deklarációk

A CSS **szelectorokra** és **deklarációkra** épül.



A szelektorok meghatározzák, hogy mely HTML elemeket célozzák meg a stílusok. Lehetnek egyszerűek, mint például egy elem kiválasztása a neve alapján (pl. a p szelektor minden bekezdést kiválaszt), de választhatunk HTML elemeket osztályok, azonosítók, és attribútumok alapján is.

A deklarációk meghatározzák azokat a stílus tulajdonságokat, amelyeket alkalmazni kell a kiválasztott elemekre. Egy deklaráció tulajdonságból (pl. `color`) és értékből (pl. `blue`) áll, kettősponttal elválasztva, és kapcsos zárójelek közé zárva. Például, a `p { color: blue; }` a szöveg színét kékre változtatja minden bekezdés esetén.

Nézzünk meg egy egyszerű példát:

```
<!DOCTYPE html>
<html>
<head>
  <title>Kék bekezdés</title>
  <style>
    p {
      color: blue;
```

```
    }
  </style>
</head>
<body>
  <p>Ez egy kék bekezdés.</p>
  <p>Ez is egy kék bekezdés.</p>
</body>
</html>
```

CSS stílusok megadása

Inline

Az inline CSS közvetlenül az egyes HTML-elemekre alkalmaz stílusokat a `style` attribútumon keresztül. Az elem nyitó tagjában van meghatározva, például egy <p> vagy <h1> tagben.

```
<p style="color: red; font-size: 16px;">Ez egy egyedi stílusú bekezdés,
piros színnel és 16 pixel betűmérettel.</p>
<p>Ez a bekezdés az alapértelmezett formázást használja, nincs rá alkalmazva
CSS.</p>
```

Az inline CSS hasznos lehet az oldalon belül csak egy adott elem módosítására anélkül, hogy a lap többi részére hatással lenne. Az inline CSS gyakran nem ideális megoldás, mivel ezek a stílusok nem használhatók könnyen újra az oldal más részein.

Beágyazott

Beágyazott CSS a <style> tagben található, a HTML-dokumentum <head> szekciójában. Ez a módszer lehetővé teszi, hogy több elemre alkalmazzunk azonos stílusokat az oldalon belül.

```
<head>
<style>
  h1 {
    color: blue;
  }

  .bold {
    font-weight: bold;
  }
</style>
</head>
<body>
  <h1>Kék címsor</h1>
  <p>Ez egy <span class="bold">példa</span> bekezdés.</p>
</body>
```

A beágyazott CSS hasznos lehet egyetlen oldal stílusának meghatározására, mivel az összes CSS-szabály egy helyen marad a dokumentumon belül. Kevésbé hatékony azonban, ha több oldallal dolgozunk, mivel a stílusokat HTML-dokumentumok között nem tudjuk újrafelhasználni.

Külső

A külső CSS egy külön `.css` fájlban van meghatározva, és a HTML-dokumentumhoz a `<link>` elemmel kapcsolható. Az így kezelt CSS fájlra akár több HTML-fájl is hivatkozhat.

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

Egy külső stíluslap a következőhöz hasonló szabálygyűjteményt tartalmazhat:

```
/* styles.css */
p { color: blue; line-height: 1.5; }
h1 { font: 32px Arial; }
```

A külső CSS ideális több lapból álló weboldalak számára, mert lehetővé teszi az egységes stílusok alkalmazását több dokumentumban. Továbbá javítja a betöltési időket azáltal, hogy a böngészők gyorsítótárazhatják a különálló CSS-stíluslapot.

Kommentek

A CSS-ben a kommentek arra szolgálnak, hogy magyarázatokat adjunk a kódhoz, vagy ideiglenesen letiltsunk kódrészleteket anélkül, hogy ez befolyásolná a weboldal stílusát. A böngésző nem jeleníti meg ezeket, és nem hatnak a lap elrendezésére vagy megjelenésére.

```
/* Formázási szabályok a bekezdésekhez */
p {
  color: black; /* A szöveg színének beállítása feketére */
  font-size: 16pt; /* Betűméret beállítása 16 pontra */
}
```

Elem, ID és class selectorok

A CSS-szelektorok szolgálnak a HTML dokumentumon belül elemek kiválasztására. Három gyakori típusa az **elem**, a **class**, és az **ID** szelektor, amelyek mindegyike eltérő célt szolgál.

Az **elem szelektor** egy adott HTML-elemet céloz meg az oldalon, egyszerűen az elem nevével van meghatározva, például `p`, `h1` vagy `div`. Az alábbi példában az összes <p> elem 16px betűmérettel és fekete szövegszínnel jelenik meg:

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

```
    <style>
        p {
            font-size: 16px;
            color: black;
        }
    </style>
</head>
<body>
    <p>Ez a bekezdés elem szelektorral kerül formázásra.</p>
    <p>Erre a bekezdésre ugyanazok a stílusok érvényesek.</p>
</body>
</html>
```

A **class szelektor** egy vagy több elemet céloz meg, melyek ugyanazzal a `class` attribútummal rendelkeznek. Az osztályszelektorokat egy pont és az osztály neve határozza meg. Az alábbi példában minden `p` és `div` elem, amely a `highlight` osztályt tartalmazza, sárga háttérrel és félkövér betűstílussal jelenik meg. Mivel az osztályok több elemhez is hozzárendelhetők, a class szelektorok nagyobb rugalmasságot biztosítanak a stílusok alkalmazásában.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .highlight {
            background-color: yellow;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <p class="highlight">Ez a bekezdés class szelektorral kerül
formázásra.</p>
    <p>Ez nem egy kiemelt bekezdés.</p>
    <div class="highlight">Ezen az elem a legelső bekezdéshez hasonlóan van
kiemelve.</div>
</body>
</html>
```

Az **ID szelektor** egyetlen HTML-elemet céloz meg, annak `id` attribútuma alapján. Az ID szelektorokat a # szimbólum és az azonosító neve jelöli. Az alábbi példában kizárólag a <h1> elem (amelynek `id="main-header"` van beállítva) kapja meg a megadott stílust, míg a többi elem változatlan marad.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        #main-header {
            font-size: 24px;
```

```
            text-align: center;
        }
    </style>
</head>
<body>
    <h1 id="main-header">Ez a címsor ID szelektorral kerül formázásra.</h1>
    <p>Ezt a bekezdést nem érinti az ID szelektor.</p>
</body>
</html>
```

Összegzésként az elem szelektorok széles körű formázásra használhatók, az ID szelektorok egyedi elemek célzott formázására szolgálnak, míg a class szelektorok lehetővé teszik több elem egyidejű formázását, magas szintű rugalmasságot biztosítva. Ezen szelektorok megértése és megfelelő kombinálása kulcsfontosságú a hatékony és könnyen karbantartható CSS írásához.

Az elemszelektorokat az elem neve jelöli, az osztályszelektorokat egy `.` és az osztály neve, míg az ID szelektorokat egy # és az azonosító neve:

Szelektor típusa	HTML példa	CSS szelektor
Elem szelektor	Test	p
Class szelektor	Test	.highlight
ID szelektor	Test	#main-text

Kaszkádolás

Alapértelmezés szerint, ha több szabály célozza meg ugyanazt a szelektort, a későbbi szabály felülírja az előzőt:

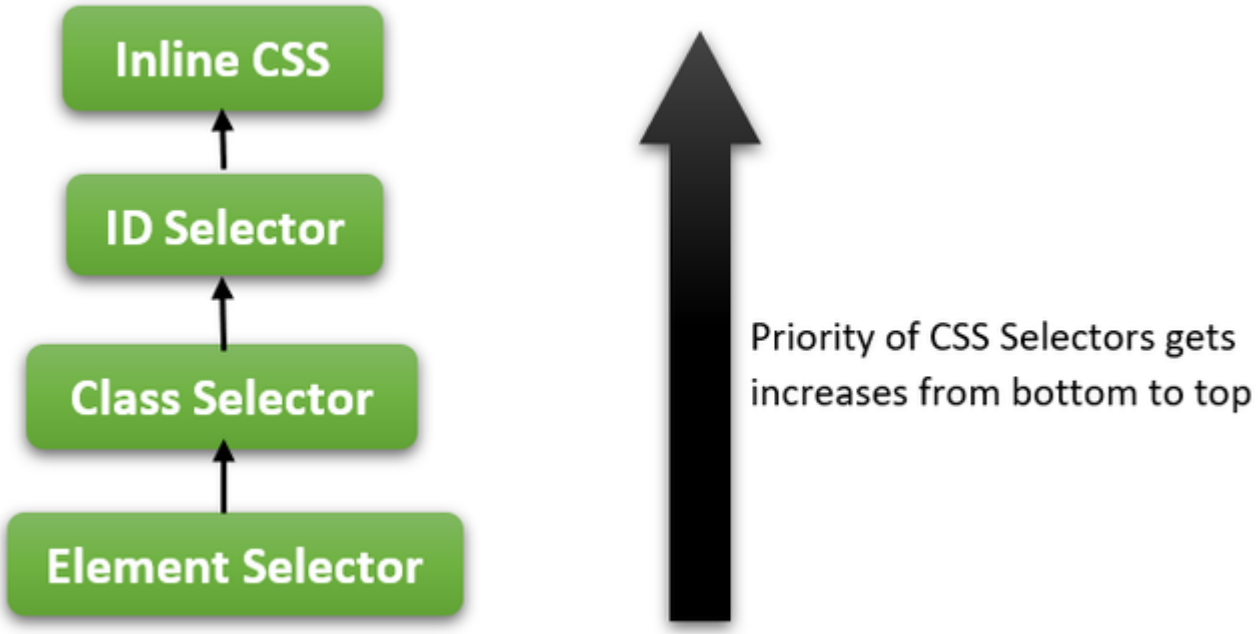
```
<head>
    <style>
        p {
            color: red;
        }

        p {
            color: green;
        }
    </style>
</head>
<body>
    <p>Ennek a bekezdésnek zöld színe lesz.</p>
</body>
```

Ezen a működésen kivételes esetekben az !important tulajdonság alkalmazásával tudunk változtatni.

Hierarchia

Egyes szabályok képesek egymást felülírni. Mindig a hierarchiában magasabb szinten álló szabály fog érvényesülni az elemre.



A következő példában a HTML-bekezdésre mindkét szabály érvényesül. Az egymásnak ellentmondó stílusok (zöld vagy piros szín) közül a specifikusabb szelektor segítségével meghatározott stílus kerül alkalmazásra:

```
<head>
    <style>
        /* id szintű selector */
        #green-text {
            color: green;
        }

        /* elem szintű selector */
        p {
            font-weight: bold;
            color: red;
        }
    </style>
</head>
<body>
    <p id="green-text">Ez egy félkövér, zöld színű bekezdés.</p>
</body>
```

Az ID szintű kiválasztás specifikusabb, mint az elem szintű, így a bekezdés színe zöld lesz.

span és div elemek szerepe

A **span** és **div** elemek alapvető HTML-komponensek, amelyeket tartalmak csoportosítására és szervezésére használnak. Ezek olyan konténerek, amelyeket CSS segítségével lehet formázni.

A span elem egy **inline konténer**, melyet általában kisebb szövegrészek formázására használnak. Ebben a példában a span csak a „kék” szót színezi kékre anélkül, hogy a többi szövegre hatással lenne. A span elem ideális lehet specifikus stílusok alkalmazására egy hosszabb szövegen belül, például a szövegszín vagy betűvastagság megváltoztatására:

```
<p>Ez egy <span style="color: blue;">kék</span> szó egy bekezdésben.</p>
```

A div elem egy **blokk szintű konténer**, ami azt jelenti, hogy alapértelmezés szerint a szülőelem teljes szélességét elfoglalja, és előtte és utána sortörést hoz létre. Általában nagyobb tartalmi egységek, például bekezdések, képek vagy más blokk szintű elemek csoportosítására használják. Például:

```
<div style="background-color: lightgrey;">
  <h1>Szakasz címe</h1>
  <p>Ez a bekezdés egy div konténerben található.</p>
</div>
```

Háttér formázása

A háttérstílusok használata CSS-ben lehetővé teszi a fejlesztők számára, hogy testre szabják egy elem hátterét. Számos olyan tulajdonság létezik, amelyekkel a háttér színét, képeit, átmeneteit és pozícióját lehet szabályozni. Például:

```
body {
    background-color: #f0f0f0;
}

#header {
    background-image: url('header.jpg');
}
```

Margin, padding és border

A **doboz-modell** (**box model**) egy alapvető CSS-koncepció, amely leírja, hogyan vannak az elemek felépítve és elhelyezve egy weboldalon. Minden HTML-elem egy téglalap alakú dobozként jelenik meg, amely négy különálló részből áll: margó (**margin**), szegély (**border**), kitöltés (**padding**) és maga a tartalom (**content**). A dobozmodell megértése kulcsfontosságú a webalkalmazások hatékony elrendezése és dizájnja szempontjából.

A `margin` az elem körüli külső tér, míg a `padding` a tartalom és a szegély közötti belső tér. A `szegély` határozza meg az elem körvonalát.

Az alábbi példa megmutatja, hogyan lehet egy gombot formázni. Próbáld ki, mi történik, ha módosítod a border, margin, és padding tulajdonságok értékeit!

```
<!DOCTYPE html>
<html>
<head>
  <title>Formázott gomb példa</title>
  <style>
    .styled-button {
      color: green;
      background-color: lightgreen;
      margin: 10px;
      padding: 20px;
      border: 1px solid red;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
  <button class="styled-button">Kattints rám!</button>
  <p>Maecenas sapien arcu, finibus eget dignissim quis, placerat vel
ante.</p>
</body>
</html>
```

Szélesség és magasság beállítása

A `width` és `height` tulajdonságokkal lehet a méreteket meghatározni.

```
img {
    width: 100px;
    height: 100px;
}
```

Szövegformázás

A szövegformázásra számos tulajdonságot lehet használni.

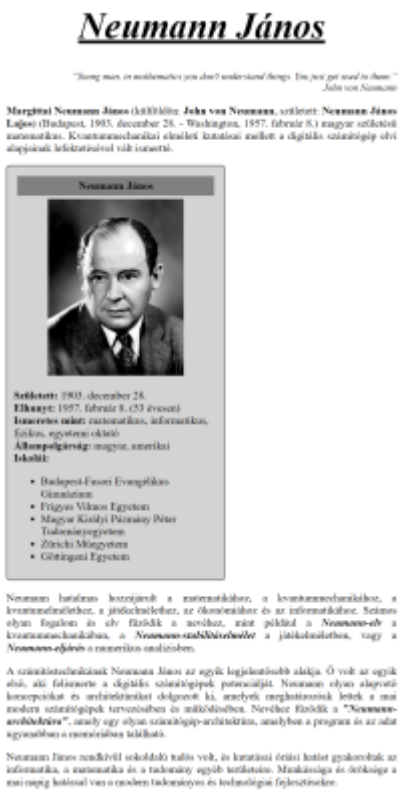
```
p {
    font-family: Arial;
    font-size: 16px;
    font-weight: bold;
    font-style: italic;

    text-decoration: underline;
    text-align: justify;
}
```

Ha érdekel, próbálj ki többet itt.

Feladat

Készítsd el a következő weboldalt, a szöveg formázásához használj inline CSS-t, valamint class és id alapú selectorokat is! A kész megoldást töltsd fel a GitHub repositorydba!



Neumann János

"Young man, in mathematics you don't understand things. You just get used to them."
John von Neumann

Margittai Neumann János (külföldön: **John von Neumann**, született: **Neumann János Lajos**) (Budapest, 1903. december 28. – Washington, 1957. február 8.) magyar születésű matematikus. Kvantummechanikai elméleti kutatásai mellett a digitális számítógép elvi alapjainak lefektetésével vált ismertté.

Neumann János

Született: 1903. december 28.
Elhunyt: 1957. február 8. (53 évesen)
Ismeretes mint: matematikus, informatikus, fizikus, egyetemi oktató
Állampolgársága: magyar, amerikai
Iskolái:

- Budapest-Fasori Evangélikus Gimnázium
- Trigyn Vilmos Egyetem
- Magyar Királyi Pázmány Péter Tudományegyetem
- Zürichi Műegyetem
- Göttingeni Egyetem

Neumann hatalmas hozzájárult a matematikához, a kvantummechanikához, a kvantumstatisztikához, a játékelmélethez, az ökonómiához és az informatikához. Számos olyan fogalom és elv fűződik a nevéhez, mint például a ***Neumann-elv*** a kvantummechanikában, a ***Neumann-stabilitáselmélet*** a játékelméletben, vagy a ***Neumann-féleírás*** a matematikai analízisben.

A számítástechnikának Neumann János az egyik legjelentősebb alakja. Ő volt az egyik első, aki felismerte a digitális számítógépek potenciáljait. Neumann olyan alapvető koncepciókat és architektúrákat dolgozott ki, amelyek megkönnyítették a mai modern számítógépek tervezésében és működésében. Nevéhez fűződik a **"Neumann-architektúra"**, amely egy olyan számítógép-architektúra, amelyben a program és az adat egyaránt a memóriában található.

Neumann János rendkívül sokoldalú tudós volt, és kutatásai óriási hatást gyakoroltak az informatika, a matematika és a tudomány egyéb területeire. Munkássága és öröksége a mai napig hatással van a modern tudományos és technológiai fejlesztésekre.

Nyers szöveg:

```
Neumann János

“Young man, in mathematics you don't understand things.
You just get used to them.”
```

John von Neumann

Margittai Neumann János (külföldön: John von Neumann,
született: Neumann János Lajos)
(Budapest, 1903. december 28. - Washington, 1957. február 8.)
magyar születésű matematikus. Kvantummechanikai elméleti kutatásai mellett a digitális számítógép elvi alapjainak lefektetésével vált ismertté.

Neumann János

https://upload.wikimedia.org/wikipedia/commons/thumb/d/d6/JohnvonNeumann-LosAlamos.jpg/462px-JohnvonNeumann-LosAlamos.jpg

Született: 1903. december 28.
Elhunyt: 1957. február 8. (53 évesen)
Ismeretes mint: matematikus, informatikus, fizikus, egyetemi oktató
Állampolgárság: magyar, amerikai
Iskolái:

Budapest-Fasori Evangélikus Gimnázium
Frigyes Vilmos Egyetem
Magyar Királyi Pázmány Péter Tudományegyetem
Zürichi Műegyetem
Göttingeni Egyetem

Neumann hatalmas hozzájárult a matematikához, a kvantummechanikához, a kvantumelmélethez, a játékelmélethez, az ökonómiához és az informatikához. Számos olyan fogalom és elv fűződik a nevéhez, mint például a Neumann-elv a kvantummechanikában, a Neumann-stabilitáselmélet a játékelméletben, vagy a Neumann-eljárás a numerikus analízisben.

A számítástechnikának Neumann János jelentős alakja, elsők között ismerte fel a digitális számítógépek potenciálját.
Neumann olyan alapvető koncepciókat és architektúrákat dolgozott ki, amelyek meghatározóak lettek a mai modern számítógépek tervezésében és működésében. Nevéhez fűződik a "Neumann-architektúra", amely egy olyan számítógép-architektúra, amelyben a program és az adat ugyanabban a memóriában található.

Neumann János rendkívül sokoldalú tudós volt, és kutatásai óriási hatást gyakoroltak az informatika, a matematika és a tudomány egyéb területeire. Munkássága és öröksége a mai napig hatással van a modern tudományos és technológiai fejlesztésekre.