CSS

```
selector {
   prop1: val1;
   prop2: val2;
}
```

Definițiile de stiluri pot apărea în 3 locuri:

- fișiere externe (cel mai indicat) cu extensia .css → tot ce e pus în fișier poate fi reutilizat:

  <link rel="*stylesheet*" type="*text/css*" href="*style.css*">

- intern, în interiorul unui tag *style* din secțiunea *head* a documentului HTML (practic se poate oriunde în document)
- inline → de evitat:

  <img src="*poza.jpg*" width="*100*" height="*100*" style="*border: 1px solid red*">

!!! E bine să specificăm proprietățile *width* și *height* ale unei imagini, deoarece ajutăm browserul să randeze mai repede. Dacă nu specificăm, browserul vede că trebuie să încarce o poză și face un request separat pentru a aduce poza. Până o aduce, browserul nu știe cât loc trebuie să lase în pagină pentru poză, și atunci procesul de randare va dura mai mult.

Care e ordinea de aplicare a stilurilor?

Stilurile inline au prioritate aproape absolută. Regula e: perechea atribut-valoare care apare ultima (cât mai aproape de tag-ul care se aplică) are prioritate.

```
<style>
   div {
      color: blue;
   }
</style>
<link rel="stylesheet" type="text/css" href="style.css">
<div> Ana are mere </div>
```

În exemplul de mai sus, culoarea textului va fi dată de ce este scris în fișierul extern style.css.

Dacă se pune **!important** la o pereche atribut-valoare, atunci se ignoră orice regulă și se aplică stilul determinat de acea pereche.

Dacă într-un fișier .css avem:

```
div {
   color: blue;
   background-color: yellow;
}
```

și avem definiția inline:

 Ana are mere

→ culorile textului se suprascriu și ajunge să fie de culoare verde, iar culoarea de fundal va fi galbenă.

Display:

- inline → lățimea elementelor e dată de conținutul din interiorul lor (span, a, img etc.)
- block → elementele se lățesc pe toată lățimea disponibilă a container-ului (div, p, ul, ol, h1-h6 etc.)

Inline-block – elementele sunt afișate pe aceeași linie, dar se poate seta și width-ul lor.

```
margin: 10px;          → toate marginile vor fi de 10px
margin: 10px 20px;     → margin top & bottom: 10px și margin left & right: 20px
margin: 10px 20px 30px; → margin top: 10px, margin left & right: 20px și margin bottom: 30px
```

* { …. } → selector wildcard

 Ana are mere

```
#mydiv {
  color: yellow;
}
```

```
div {
   color: red;
}
```

```
.patrat {
   color: blue;
}
```

→ textul va fi de culoare galbenă

<div class=”*patrat colorat*”> … → face parte din două clase

<div class=”*colorat1 colorat2*”> … → are prioritate al doilea definit în fișierul .css

```
<div id=”id1”>
   <div id=”id2”>
      Ana are mere
   </div>
</div>
```

```
#id1 {
   background-color: red;
   text-align: center;
}
```

Pentru div-ul id2:

1. Lățimea: lățimea div-ului părinte (are display block)
2. Culoarea de fundal: transparent (s-a colorat div-ul părinte)
3. Valoarea proprietății css text-align: center (s-a moștenit valoarea atributului)

Curs 5

z-index → există situații în care mai multe elemente din cadrul paginii ajung să fie suprapuse vizual; cu cât z-index e mai mare, cu atât elementul respectiv o să fie mai deasupra

Ce se întâmplă când avem un element cu display: block băgat într-un element cu display: inline?

Bufny nu răspunde plm. Am găsit răspuns (cred): un container inline nu poate conține un container block. Forțat, container-ul inline va deveni block pentru a acomoda conținutul container-ului fiu.

!!! text-align centrează textul doar din cadrul unui container cu display: inline; la un *ul* (de exemplu), care are display: block, nu va fi centrat conținutul

Folosind dimensiuni relative, e mai ușor să facem conținutul respectiv să fie responsive.

!!! Randarea unei pagini se face de la stânga la dreapta, de sus în jos.