Univ. Babeș-Bolyai,

Facultatea de Matematică și Informatică

Lect. dr. Darius Bufnea

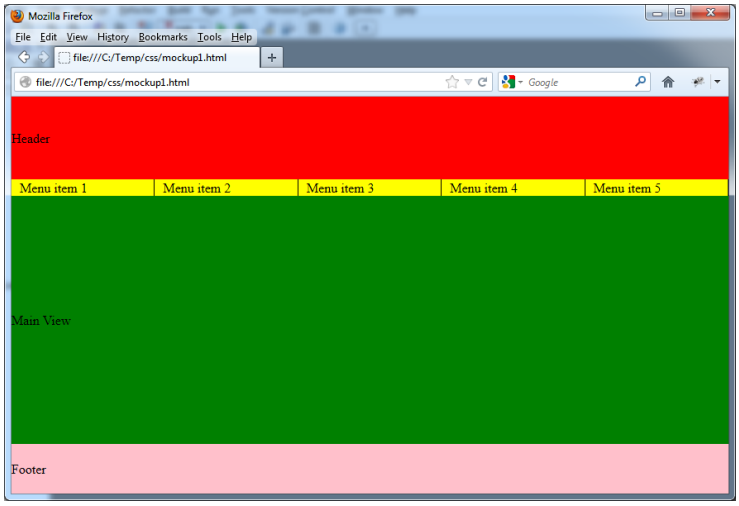
Notițe de curs Programare Web: CSS (săptămâna 3 și 4 de școală)

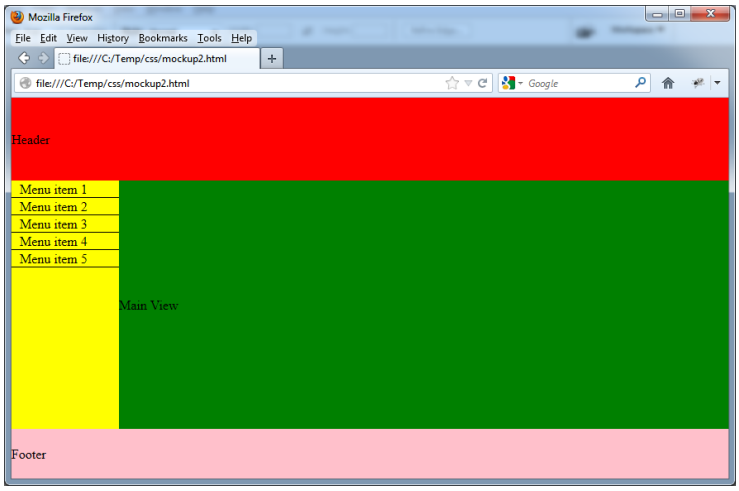
CSS – abreviere de la „Cascading Style Sheets”.

“Filozofia” ce stă în spatele CSS-ului este de a separa partea de „prezentare”, adică de „cum arată” pagina, de structura acesteia: până la apariția CSS-ului, o pagina web conținea doar tag-uri și conținut ce era „format” de aceste tag-uri. Practic, in primele versiuni ale limbajului HTML rolul tag-urile era de a „formata” textul si nu neapărat de a „structura” documentul. In ultimele versiuni ale limbajului HTML, 4.01 dar mai ales in HTML 5, s-a pus accentul pe rolul tag-urilor de a structura documentul (tag-urile specifica, că documentul conține un titlu, un heading 1, un paragraf, un meniu, o lista, un footer, etc…) și nu neapărat de a „formata” documentul. Din cursul / laboratorul anterior de HTML ar fi trebuit să rămâneți cu informația ca in HTML 5 au fost introduse unele tag-uri noi in acest sens (câteva exemple doar: article, aside, header, footer, nav), in timp ce tag-uri sau atribute de „prezentare” prezente in versiunile mai vechi de HTML sunt deprecated. Câteva exemple in acest sens: font, center, u, bgcolor, align.

Folosind CSS, o pagina web poate fi „prezentată” (colorată, aranjată) in moduri diferite, păstrându-se structura continutului acesteia. Cateva analogii simple: schimband definitiile de stiluri CSS folosite intr-o pagina web, pagina web isi poate schimba „look & feel”-ul asemanator cu o prezentare Power Point la care se schimba „tema”. Un alt exemplu pentru cei ce ati folosit „skin”-uri diferite pentru Winamp – schimband skin-ul, interfata player-ului arata diferit, dar functionalitatea acestuia era aceeasi.

Exemplul 1: Codul HTML pentru ambele exemple de mai jos este identic. Diferența dintre ele consta doar in fișierul CSS extern inclus in fiecare dintre ele.





Pe lângă prezentul material, va recomand / încurajez / rog „ferm” să parcurgeți și următoarele materiale legate de CSS:

- https://www.w3schools.com/css/default.asp
- https://www.w3schools.com/css/css_intro.asp
- https://www.w3schools.com/css/css_rwd_intro.asp
- http://www.tizag.com/cssT/index.php
- http://www.barelyfitz.com/screencast/html-training/css/positioning/

Unde pot apărea elemente legate de stil in cadrul unei pagini HTML:

1. Definiţii de stiluri externe

In antetul pagini (in interiorul tag-ului head), prin includerea unui fișier extern cu extensia .css cu ajutorul tag-ului link.

Exemplu:

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

Fișierul stiluri.css va conține definiții de stiluri externe.

2. Definiţii de stiluri interne

In antetul pagini (in interiorul tag-ului head), prin plasarea definițiilor de stiluri in interiorul tagului <style>. Exemplu:

```
<style type="text/css">
    table {
        font-family: Tahoma;
    }
</style>
```

Observații:

Tagul <style> poate fi plasat oriunde in cadrul paginii html (chiar si după sfârșitul paginii (întâlnirea tag-ului </html>). Totuși, pentru randarea corecta a paginii (browserul e posibil sa afișeze porțiuni ale paginii înainte de descărcarea completa a acesteia) se recomanda plasarea stilurilor cat mai in fata in cadrul paginii (de obicei in secțiunea head).

Definițiile externe sunt de preferat celor interne deoarece stilurile definite in cadrul unui fișier extern pot fi reutilizate si in alte documente HTML (pagini Web) din cadrul unui site, pe când cele interne pot fi utilizate doar in fișierul HTML in care sunt definite.

In aproape toate exemple de cod CSS din prezentul document am omis folosirea tag-ului <style></style> pentru demarcarea codului CSS de codul HTML pentru scurtarea secvențelor de cod. Dacă testați exemplele, rog a se folosi tag-ul <style> în mod corespunzător.

3. Definiţii inline

Definițiile inline se specifică cu ajutorul atributului style pe un tag.

Exemplu:

```
<img src="poza.jpg" width="100" height="100" style="border: 3px solid red"/>
```

Se recomanda evitarea stilurilor inline datorita imposibilității reutilizării lor (exceptând mecanismul „copy/paste") si datorita dificultății înlocuirii stilului daca acesta este repetat inline pe mai multe elemente de același tip.

Cele mai recomandate definiții de stil sunt cele externe. Acestea pot fi reutilizate / incluse / partajate in toate paginile unui site sau a unei aplicații web, sau chiar partajate intre site-uri multiple. Stilurile interne pot fi folosite doar in cadrul documentului HTML in care sunt incluse, putând fi reutilizate de către mai multe elemente prezente in cadrul același document HTML. Stilurile inline nu pot fi reutilizate și se aplica doar pe un singur element cu ajutorul atributului HTML style. Din punct de vedere al reutilizării codului CSS, gradul de reutilizare este: stiluri externe (permit gradul cel mai ridicat de reutilizare) => stiluri interne => stiluri inline (nu pot fi reutilizate).

Forma generala a definiției unui stil:

```
selector {
  atribut1: valoare1;
  atribut2: valoare2;
  ...
  atributn: valoaren
}
```

Observații:

1. după ultima componentă a unui stil nu este necesar caracterul ";"
2. pentru definițiile de stiluri inline nu se folosește selector ci doar perechi de atribute și valori asociate acestora.

3. Unele documentații CSS folosesc termenul de proprietate CSS in loc de atribut CSS, pentru a se face diferența mai ușor intre atributele HTML și atributele (proprietățile CSS) – a se vedea observația 4 de mai jos.
4. A nu se confunda atributele HTML (și valorile asociate atributelor HTML) cu atributele CSS și valorile asociate atributelor CSS. In exemplul de mai jos, care folosește un stil inline:

```
<img src="poza.jpg" alt="O poza" style="border: 3px solid red; margin: 10px"/>
```

src, alt și style sunt atribute HTML, iar "poza.jpg", "O poza" și respectiv "border: 3px solid red; margin: 10px" sunt valorile asociate acestor atribute. border și margin reprezinta atribute (proprietăți) CSS iar 3px solid red și respectiv 10px reprezintă valorile asociate acestor atribute CSS.

Tipuri de selectori

Lista de mai jos conține cei mai utilizați selectori. Pentru o lista extinsa a acestora va rog sa consultați: https://www.w3schools.com/cssref/css_selectors.asp

1. Aplicabile pe toate tagurile de acelasi tip din cadrul unui document HTML

Exemplu - se aplică pe toate imaginile (tag-urile img) din pagină (acestea vor fi înconjurate cu un chenar alabastru de 3 pixeli grosime)

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

```
/*
Acesta este un comentariu CSS. Definiția de stil de mai jos se aplică pe tag-ul body. O astfel de definiție poate apărea intr-un fișier separat (inclus cu ajutorul tag-ului link) sau in interiorul tag-ului <style></style>.
*/
body {
  font-family: Tahoma, Arial Black;
  background-color: blue;
  margin: 0;
  padding: 0;
}
```

2. Definiții de clasa - aplicabile pe orice tag

Exemplu, se aplică pe toate tag-urile din clasa „.text”, asocierea cu această clasă făcându-se cu ajutorul atributului HTML class.

```
.text {
  font-family: Tahoma;
  font-size: 12px;
  color: #666666;
}
```

```
<div class="text">
    Ana are mere și cocosul cântă.
</div>
```

Observație: un element poate fi declarat ca aparținând mai multor clase. În acest caz se vor aplica proprietățile CSS din ambele clase (cu anumite reguli ce vor fi prezentate în secțiunea: *Concatenarea / prioritatea / suprascrierea definițiilor de stil* a acestui curs). Exemplu:

```
<div class="colorat patrat">
    Ana are mere și cocosul cântă.
</div>
```

3. Aplicabile doar pe tag-ul cu un anumit id

Exemplu:

```
#demo {
  background-color: red;
  border: 1px solid black;
  color: white;
  width: 300px;
  text-align: center;
}
<div id="demo">
    Alba ca zapada si cei sapte pitici
</div>
```

Rember!: Atributul id se folosește în principal pentru a stiliza elementul din CSS și pentru al putea referi din JavaScript (sau alte librării / frameworkuri client-side bazate pe JavaScript).

4. Selectori compuși, aplicabili doar pe anumite de tag-uri și doar dacă se specifica explicit cu ajutorul atributului class

```
div.fcsb {
    background-color: blue;
    color: red;
}

div.dinamo {
    background-color: red;
    color: white;
}

<div class="fcsb">
    FCSB
</div>

<div class="dinamo">
    Dinamo
</div>

<!-- Pe elementul de mai jos nu se aplica întrucât stilul definit se aplica
pe div-urile din clasa "dinamo" nu pe span-urile din această clasă -->
```

```
<span class="dinamo">
    Dinamo vs. FCSB
</span>
```

5. Pseudo clase (sau pseudo selectori)

Exemple:

```
/* se aplică pe toate link-urile din pagina */
a:link {
  color: red;
  text-decoration: none;
}

/* se aplică pe toate link-urile vizitate */
a:visited {
  color: red;
  text-decoration: none;
}

/* se aplică pe toate link-urile când se merge cu cursorul de mouse deasupra
link-ului */
a:hover {
  color: blue;
  text-decoration: none;
}
```

6. Alte exemple de selectori

Exemplul 1 - se aplica pe tag-urile ancora vizitate plasate in clasa legătura:

```
a.legatura:visited {
  color: red;
  text-decoration: none;
}

<a href="altapagina.html" class="legatura">Link nesubliniat</a>
```

Exemplul 2 - definește cum arata celulele (td-urile) din cadrul unui tabel de tipul (din clasa) chenar:

```
table.chenar td {
   font: 80% Verdana, Arial, Helvetica, sans-serif;
   color: #666666;
   text-align: justify;
   padding: 1px 5px 1px 5px;
   border-right: 2px;
   border-bottom: 2px;
   background-color: #eeeeee;
}
```

Exemplul 3 – definește cum arata textul încapsulat in tag-ul <a> ce apare in cadrul unui element al unei liste neordonate plasate în cadrul unui div din clasa meniu:

```
div.meniu ul li a {
```

```
    ...
}
```

Exemplul 4, pseudo selector, se aplică pe toate liniile pare dintr-un tabel:

```
tr:nth-child(even) {
    background-color: pink;
}
```

Observație: O definiție de stil poate fi precedată de doi sau mai mulți selectori dacă se dorește aplicarea/folosirea aceluiași stil în mai multe situații. Stilul din exemplul de mai jos se aplică pe toate div-urile, dar și pe toate elementele (tabele, span-uri, etc.) plasate în clasa "text":

```
.text, div {
    font-family: Tahoma;
    font-size: 12px;
    color: #666666;
}
<span class="text">Ana are mere</span><br/>
<div>
    Cocosul canta
</div>
```

Ce se întâmplă dacă pe un element trebuie aplicate mai multe definiții de stil în același timp?

Concatenarea / prioritatea / suprascrierea definițiilor de stil

Pe un element pot fi aplicate mai multe definiții de stil în același timp.

Exemplul 1:

```
.patrat {
    width: 100px;
    height: 100px
}
.colorat {
    background-color: red
}
<div class="colorat patrat">
    Ana are mere și cocosul cântă.
</div>
```

Acest div va fi atât pătrat cât și colorat, adică va avea înălțimea și lățimea de 100 de pixeli (dictate de clasa patrat), iar culoarea de fundal a acestuia va fi roșie, dictată de clasa colorat.

Exemplul 2:

```
div {
    width: 100px;
    height: 100px
}
.colorat {
    background-color: red
}
<div class="colorat" style="color: blue">
    Ana are mere și cocosul cântă.
</div>
```

Div-ul anterior va fi și colorat în roșu (datorită clasei colorat) și pătrat (orice div este pătrat), iar culoarea folosită pentru text va fi albastră (datorită stilului inline).

Ordinea de aplicare a atributelor CSS / stilurilor în cazul în care se aplică mai multe pe un element

Anumite atribute CSS se pot regăsi în mai multe definiții de stil care se aplică (sau se "potrivesc") pe același element. Care dintre aceste definiții de stil are prioritate și care va fi valoarea finală a unui anumit atribut CSS care se regăsește în mai multe definiții de stil?

Ordinea de aplicare a stilurilor este dictată de doi factori:

a) "Puterea selectorului" sau cât de bine se "potrivește" selectorul cu elementul pe care se aplică. Din acest punct de vedere se aplică următoarele reguli:

- clasa are prioritate față de atributele setate de un selector specificat prin tag;
- selectorul specificat prin intermediul id-ului tag-ului are prioritate în fața clasei.

Exemplu

```
#mydiv {
    background-color: yellow
}
.colorat {
    background-color: red
}
div {
    background-color: blue
}
<div class="colorat" id="mydiv">
    Ana are mere și cocosul cântă.
</div>
```

Div-ul anterior va fi colorat în galben (pentru ca selectorul bazat pe id "bate" clasa, care bate selectorul bazat pe numele tag-ului). Daca in schimb, tag-ului div i se adaugă și un atribut style:

```
<div class="colorat" id="mydiv" style="background-color: pink">
    Ana are mere și cocosul cântă.
</div>
```

divul anterior va fi colorat in roz. Stilurile inline "bat" inclusiv selector bazat pe id (a se vedea și punctul b) de mai jos).

b) Ordinea de aplicare a stilurilor este dictată și de ordinea în care apar în cadrul documentului HTML / documentelor CSS definițiile de stil.

Reguli:

- O definiție de stil poate fi suprascrisă în totalitate sau parțial (se pot suprascrie doar anumite atribute din cadrul definiției);
- Dacă un stil apare definit în cadrul mai multor fișiere externe (fișiere CSS importate) și/sau intern, atributele care se regăsesc în mai multe definiți vor fi suprascrise, se va păstra doar valoarea atributului din ultima definiție. Atributele care se regăsesc doar în anumite definiții se vor adăuga la caracteristicile stilului.
- nu există nicio regulă de prioritate între stilurile interne și cele externe, prioritatea este dată de apariția acestora cât mai târziu în cadrul fișierului HTML.
- Stilurile inline suprascriu întotdeauna definițiile externe sau interne, acestea având prioritate absolută - cu o mică observație - folosirea modificatorului **!important**.
- Modificatorul **!important** la sfârșitul unei valori de atribut CSS "bate" inclusiv o definiție inline sau o definiție de stil bazată pe id de element. Folosirea acestui modificator nu este este recomandată. Exemplu de folosire a modificatorului **!important** găsiți mai jos.
- Valorile atributelor stilului unui tag interior suprascriu valorile acelorași atribute din stilul unui tag părinte (spre exemplu background-color aplicat pe un div suprascrie background-color-ul aplicat pe body).

Exemplu cu folosirea modificatorului **!important**. In exemplul de mai jos, div-ul va fi colorat în roșu, atributul !important având prioritate oriunde ar apărea (indiferent de selectorul folosit) și chiar în fața unei definiții de stil inline.

```
<style>
#mydiv {
    background-color: yellow
}

div {
    background-color: red !important;
}

</style>
<div id="mydiv" style="background-color: pink">
    Ana are mere și cocosul cântă.
</div>
```

Exemplul mai complex legat de ordinea de aplicare a stilurilor. Pentru exemplul de cod de mai jos, înainte sa-l testați, încercați să răspundeți la întrebarea: Ce culoare va avea fundalul pe care este scris textul Ana are mere.

```
<style type="text/css">
    div #id1 .class2 {
        background-color: blue;
    }
    div .class1 #id2 {
        background-color: red;
    }
</style>
<div>
    <div class="class1" id="id1">
        <div class="class2" id="id2">
            Ana are mere
        </div>
    </div>
</div>
```

Răspuns: Texul va fi scris pe fundal roșu. Ambele definiții de stil sa potrivesc la de bine („au aceeași putere”), contând ultima valoare setată pentru atributul background-color.

Cele mai importante / des utilizate atribute CSS

Tabelul de mai jos este consultativ și nu este complet. Pentru lista completă a atributelor CSS, rolul și efectul acestora precum și pentru valorile asociate acestor atribute vă rog să consultați documentația de la adresa: https://www.w3schools.com/cssref/default.asp

font-family	Tahoma, Arial, Verdana, Helvetica, sans-serif
font-size	12px
line-height	16px
font-weight	bold
letter-spacing	0.7px;
color	#666666
word-spacing	2px
text-align	justify
background-image	none
background-image	url(../images/selected_left.png)
background-repeat	no-repeat
width	100%
height	34px
vertical-align	middle
text-align	center
cursor	pointer
border	1px solid #CCCCCC
background-color	white
padding	1px 5px 1px 5px
margin	1px 5px 1px 5px
visibility	Hidden
z-index	2

Valori implicite pentru atribute / moștenirea valorilor atributelor CSS

Unele atribute CSS primesc o valoare default dacă nu sunt specificate explicit într-o definiție de stil. Spre exemplu atributul `border-style` prin care se poate specifica stilul bordurii unui element are valoarea default setată la "none" (adică elementul nu va avea bordură dacă nu este specificată o valoare pentru acest atribut).

Alte atribute CSS moștenesc valoare de la containerul părinte. Exemplu `text-align`, atribut CSS care specifică modul de aliniere pe orizontală a textului în cadrul unui container.

Cele două exemple de mai sus sunt orientative, fiecare atribut CSS comportându-se diferit. Pentru valoarea implicită a fiecărui atribut CSS sau faptul că valoarea acestuia se moștenește sau nu de la containerul părinte, puteți consulta https://www.w3schools.com/cssref/.

Exemplu:

```
<style type="text/css">
#id1 {
   background-color: yellow;
   text-align: center;
   width: 200px
}
</style>
<div id="id1">
   <div id="id2">
   Ana are mere
   </div>
</div>
```

Pe exemplul de mai sus, întrebările sunt: pe ce fundal se va scrie textul, cum va fi aliniat acesta și care va fi lățimea div-ului id2 (și de ce? la fiecare dintre aceste trei întrebări). Răspuns: pe fundal galben, centrat, iar lățimea div-ului id2 este de 200px. Practic, div-ul id2 are același comportament vizual cu div-ul id1 dar din trei motive diferite:

- valoarea atributului text-align se moștenește de la containerul părinte id1 la containerul fiu id2;
- background-color nu se moștenește, dar pe id2 aceasta va avea valoarea default transparent, și drept urmare se va vedea în spate divul id1 colorat in galben;
- Containerul id2 este un div, acest element având display-ul implicit bloc. Acest lucru înseamnă că lățimea sa este aceeași cu lățimea containerului părinte, div-ul id2 lățindu-se pe toată lățimea lui disponibilă (lățimea lui id1).

Shorthand property

Anumite atribute CSS se pot "colapsa" și prescurta printr-o singură proprietate. Spre exemplu:

```
border: 1px solid blue;
```

este prescurtarea de la:

```
border-width: 1px;
border-style: solid;
border-color: blue;
```

Din lista de valori primite de un atribut "shorthand", unele pot sa lipsească, caz în care atributul inițial ce a fost prescurtat primește valoarea default. In exeplul de mai jos, grosimea bordurii va fi setată la valoarea `medium` - valoarea default pentru atributul `border-width`, nefiind specificat explicit nici acest atribut, și nici o valoare pentru acesta în cadrul atributului shorthand `border`.

```
border: blue solid;
```

De asemenea, există unele situații când ordinea în care sunt specificate valorile pentru un atribut shorthand este importantă. Un exemplu în acest sens în secțiunea următoare.

CSS box model

Un element vizibil din cadrul unui document HTML poate fi privit ca un fel de chenar (box) în care distingem:

- Marginea exterioară (din exteriorul) elementului;
- Bordura elementului;
- Spațiul interior în cadrul elementului dintre bordură și conținut;
- Conținutul elementului.

Detalii aici: https://www.w3schools.com/css/css_boxmodel.asp

Marginea exterioară a unui element poate fi modificată cu atributele CSS margin-top, margin-right, margin-bottom, margin-left sau cu atributul shorthand margin.

Bordura elementului se poate specifica cu atributele CSS: border-top, border-right, border-bottom, border-left (atenție, toate aceste atribute sunt shorthand pentru alte proprietăți), ele putându-se colapsa mai departe cu atributul shorthand border.

Spațiul interior în cadrul elementului dintre bordură și conținut se poate modifică cu atributele CSS padding-top, padding-right, padding-bottom, padding-left sau cu varianta shorthand padding.

Exemplul, specifică în ordinea acelor de ceasornic, plecând de la ora 12, valorile pentru atributele padding-top, padding-right, padding-bottom, padding-left pentru div-urile din pagină:

```
div {
    padding: 1px 2px 3px 4px;
}
```

In varianta:

```
div {
    padding: 5px;
}
```
toate atributele padding-top, padding-right, padding-bottom, padding-left au valoarea de 5px. Iar in varianta:

```
div {
    padding: 5px 10px;
}
```

padding-top și padding-bottom au valoarea 5px iar padding-right și padding-left au valoarea de 10px. Există și varianta cu 3 valori pe care vă recomand sa o evitați! ☺

```
div {
    padding: 5px 10px 15px;
}
```

Funcționalitatea atributului shorthand margin e similară. Atributul padding e util și folosit mult pentru span-uri, div-uri, td-uri de tabele pentru a specifica spațiul liber dintre conținutul unor astfel de containere și bordura lor, în timp ce margin e util pentru a delimita spațiul liber din jurul unui element în general (spre exemplu din jurul unui figuri inserate cu ajutorul tag-ului img).

Culori CSS

Culorile în CSS / HTML se specifică:

- cu literali cuvinte din limba engleză: pink, red, blue, yellow, etc;
- in hexazecimal, printr-un cod de forma #RRGGBB, unde RR reprezintă cantitatea de roșu, GG cantitatea de verde, iar BB cantitatea de albastru. Aceste valori se specifică în baza 16 de la 00 (0) la FF (255). #000000 este negru, #FFFFFF este alb, iar un cod de culoare cu toate cele 6 cifre hexazecimale egale reprezintă o nuanță de gri.
- in format RGB. Exemplu: rgb(255,192,203) - 255 e cantitatea de roșu, 192 de verde și 203 de albastru.

Observație: pentru codul de culoare #AABBCC se poate folosi varianta shorthand #ABC.

Display inline vs. display block

Anumite tag-uri din limbajul HTML se folosesc pe post de containere, pentru a „îngloba” alte tag-uri. Cele mai populare containere sunt span si div. Diferența dintre acestea două este modul de randare a acestora in flow-ul normal de randare a documentului:

- Doua elemente cu display inline sunt randate unul după celălalt, pe aceeași linie, iar lățimea unui astfel de container este dată de conținutul din interiorul containerului. Printre elementele HTML cu display inline se numără: span, a, img, b, i, input
- Elementele cu display block sunt randate unul sub celalalt (se inserează un fel de „line break", sau „enter" intre ele). Printre elementele HTML cu display inline se numără: div, p, ul, ol, h1, h2, h3…

Unui span (care are display-ul inline) nu i se poate seta atributul width (lățimea acestuia fiind data de textul din interior). Daca se dorește setarea unei lățimi pe un container inline, se poate seta atributul display la valoarea inline-block (un element cu un astfel de display se comporta ca unul inline dar i se poate seta și lățimea).

Centrarea orizontală a conținutului unui container

Centrarea conținutului unui container se face cu ajutorul atributului CSS text-align. De multe ori însă acest atribut este „supra estimat" deoarece afectează doar conținutul inline sau inline-block din cadrul containerului, și nu va centra un element (subcontiner) cu display block. Exemplu:

```
<div style="border: 1px solid red; text-align: center">
Ana are mere
<ul style="width: 300px; border: 1px solid black">
<li>Element din lista</li>
</ul>
</div>
```

Pe exemplul de mai sus in cadrul div-ului se va centra textul, dar nu se va centra lista ul, deoarece display-ul acesteia nu este inline. Pentru a centra lista, acesteia trebuie sa i se adauge atributul CSS „margin: 0 auto", prin acest lucru indicându-se ca margin-left și margin-right sa fie setate automat (și să fie egale). De asemenea, pe exemplul de mai sus lista ul trebuie să aibă setată dimensiune, în caz contrar ea se lățește pe toată lățimea disponibilă a containerului părinte (și nu mai avem ce centra, cele două elemente având aceeași lățime).

Pentru lista completă a valorilor pe care le poate lua atributul CSS display, vă rog consultați documentația de pe W3Schools.

Display none vs. visibility hidden

Setarea atributului Display la valoarea none duce la ascunderea acestuia din cadrul paginii, nealocandu-se loc pentru afisarea acestuia in flow-ul normal de randare. Un exemplu de tag care foloseste display setat la none este tag-ul <script> (pentru acest tag nu se randeaza nimic in pagina, dar el este present in pagina si are effect asupra paginii).

Setarea atributului CSS visibility la valoarea hidden pentru un element, nu duce la afisarea elementului in cadrul paginii, dar in flow-ul de randare se lasa loc pentru acest element – exista spațiu alocat in pagina pentru element in locul in care acesta ar trebui să apară, doar ca elementul nu este vizibil.

Utilizarea atributului float

Utilizarea atributului float este legată mai ales de situațiile in care un text trebuie sa facă „wrap" (să înconjoare) în mod „natural" o imagine. Dacă textul de față ar fi fost scris în interiorul unui paragraf (tag <p>, imaginea alăturată din dreapta textului ar fi trebuit să aibă setat atributul CSS „float: right" pentru a se deplasa la dreapta. In caz contrar, ea ar fi fost plasata in cadrul textului in locul unde apare tag-ul img, iar in dreptul imaginii, pe orizontală, s-ar găsi o singură linie de text (indiferent de înălțimea imaginii) – acest lucru datorită display-ului inline a unei imagini. Mai multe detalii despre comportamentul atributului float și cele mai importante scenarii de utilizare ale acestuia aici.

Dacă în paragraful anterior pe imaginea din cadrul paragrafului nu s-ar fi setat atributul „float: right", imaginea ar fi fost plasată conform exemplului din prezentul paragraf. Not very nice…

CSS și JavaScript

In JvaScript, atributele CSS se pot accesa prin intermediul unei expresii de forma:

```
obiect.style.numeAtributCSS
```

cu observatia ca un atribut CSS ce contine liniuta in denumirea sa, precum text-align se transforma in textAlign. Exemplu:

```
obiect.style.textAlign
```

(acest lucru deoarece textAlign este un identificator iar in majoritatea limbajelor de programare un identificator cu numele text-align nu ar fi valid, „liniuța" neputând face parte din numele unor identificatori, variabile, constante, etc).

Unele versiunii mai vechi de Internet Explorer, permit asignarea de expresii JavaScript pentru atributele CSS. Acest lucru este util in special pentru a suplinii nesuportarea de către Internet Explorer a unor selectori cum ar fi unele pseudoclase. Exemplu:

```
<style type="text/css">
```

```
   table.last_column_justify td:last-child {
      text-align: center
   }

   table.last_column_justify td {
      text-align: expression(this.nextSibling==null?'center':'left');
   }

</style>
```

Cele două definiții de stil de mai sus sunt echivalente, cu observația că a doua nu este suportata pe niciunul din ultimele versiuni ale browser-elor moderne, fiind utilizabila doar pe versiuni mai vechi de Internet Explorer care nu suporta pseudoclase (cum ar fi :last-child).

Atribute experimentale prefixate

Unele atribute CSS experimentale care nu sunt inca pe deplin suportate de catre browsere, pot fi prefixate de catre acestea cu diferite prefixe precum: -webkit- -ms- -o- -moz-

Exemplu:

```
transform:          rotate( 29deg ) skew( -35deg );
-moz-transform:     rotate( 29deg ) skew( -35deg );
-ms-transform:      rotate( 29deg ) skew( -35deg );
-o-transform:       rotate( 29deg ) skew( -35deg );
-webkit-transform: rotate( 29deg ) skew( -35deg );
```

Atributele CSS de mai sus se aplica pe diferite browsere: transform pe toate browser-ele (sau versiunile de browser-e) ce suporta acest atribut in timp ce celelalte atribute sunt asociate diferitor versiuni de browser-e oferite de diferiți vendori (-moz- Mozilla Firefox, -ms- Microsoft Internet Explorer, -o- Opera, -webkit- Safari și Chrome). Observație: aproape toți vendorii de browsere au trecut de la un engine CSS de randare la altul in timp…

Unităţi de măsură în CSS

Valorile anumitor atribute CSS necesită precizarea unei unități de măsura. Printre cele mai populare atribute cu un astfel de comportament sunt width, height, padding, margin, left, top, right, bottom, font-size, line-height.

Unitățile de măsură în CSS se împart în:

- absolute, printre cele mai populare: px (pixeli), pt (points, 72 pt = 1 inch = 2.54 cm), mm, cm, in (inch)
- relative: em (dimensiunea fontului folosit in elementul căruia i se aplica atributul CSS a cărui valoare se specifică in em), rem (ca și em dar relativ la elementul rădăcină), %, vw (1 % din lățimea viewport-ului), vh (1 % din înălțimea viewport-ului).

Unitățile de măsură relative sunt recomandate atunci când se dorește crearea unui document HTML responsive - document care „arata bine” indiferent de device-ul pe care este afișat (desktop cu monitor de 23", tableta, telefon, ecran wide, ecran portret, etc) sau dacă fereastra browser-ului este redimensionată după ce documentul este încărcat.

Observație: Daca dimensiunea unui atribut este 0, nu mai trebuie specificată unitatea de măsura (0px, 0cm).

Pentru mai multe detalii privind unitățile de măsura in CSS:

- https://www.w3.org/Style/Examples/007/units.en.html
- https://www.w3schools.com/cssref/css_units.asp

Poziționare (atributul CSS „position”)

In funcție de tipul unui element HTML și de display-ul acestuia, el are o anumită poziție în flow-ul normal de randare a documentului. Poziția acestuia în flow-ul normal de randare poate fi modificată folosind diferite valori pentru atributul CSS position. Cele mai importante valuri pentru acest atribut:

position: static

Valoare implicită, elementul va fi randat unde ar trebui să apară in mod normal in flow-ul de randare.

position: relative

Elementul este mutat in flow-ul de randare mai la stanga, mai la dreapta, mai sus sau mai jos fata de poziția lui normală (fata de unde ar fi fost plasat in mod normal in flow-ul de randare). Valoarea atributului „relative” vine de la faptul ca elementul este plasat „relativ” la pozitia acestuia. Pentru mutarea elementului mai la stanga, dreapta, sus, jos se folosesc atributele CSS: left, right, top și respectiv bottom.

position: absolute

Elementul este mutat față de poziția primului părinte non-static (cu position diferit de static). In lipsa unui asemenea părinte, poziționarea se face față de body-ul documentului, practic poziționarea făcându-se față de document. Poziționarea față de părintele non static se face tot cu atributele CSS: left, right, top și respectiv bottom.

Revenim un pic la „position: relative”. Practic un container (cum ar fi un div) poate fi setat cu „position: relative”, din două motive:

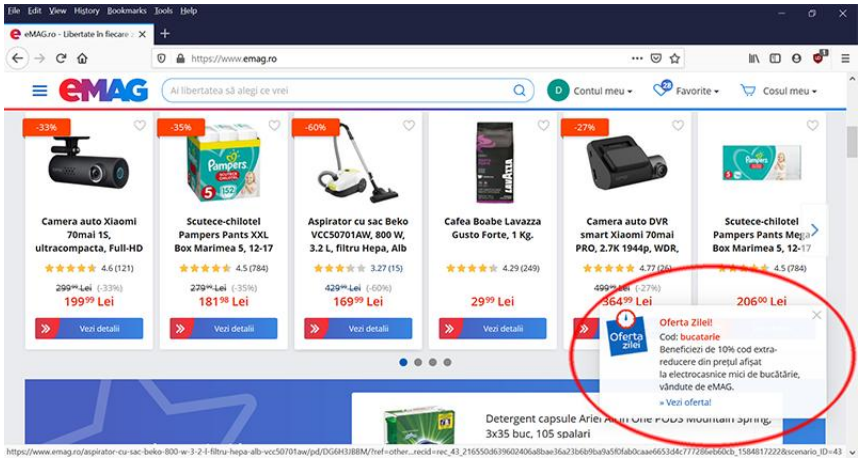
1. mutarea acestuia la o poziție relativa fata de poziția sa inițiala (caz descris deja mai sus)
2. setarea lui la position: relative, nu că să fie mutat el însuși, ci ca să devină părinte non-static pentru un alt container (de obicei div) ce are setat atributul position la absolut.

Relativ la acest al doilea caz, este des întâlnit un scenariu de utilizare precum cel de mai jos:

```
<div style="position: relative; background-color: red; width: 500px; height: 100px">
<div style="position: absolute; background-color: yellow; width: 300px; right: 0; bottom: 0">
Ana are mere
</div>
</div>
```

position: fixed

Elementul este poziționat față de fereastra browser-ului. Este folosit în special pentru a plasa containere, meniuri, etc. care sa fie poziționate tot timpul în același loc în pagină (de exemplu in header sau în footer-ul paginii). Poziționarea față de fereastra browser-ului se face tot cu atributele CSS: left, right, top și respectiv bottom. Exemplu: containerul care conține „Oferta zilei!” pe eMag.ro:



Folosirea atributului z-index

Datorită folosiri atributului `position` cu diverse valori precum cele prezentate mai sus, este posibil ca anumite elemente din pagină să se suprapună și unele dintre ele să nu mai fie vizibile fiind „acoperite” de alte elemente. În această situație, se poate folosi atributul `z-index` cu o anumită valoare număr întreg (inclusiv negative). Un element cu valoarea specificată pentru atributul `z-index` mai mare este plasat „peste” un element cu `z-index` mai mic.

Sprite-uri CSS

Un sprite CSS este o colecție de imagini (de obicei „iconițe”, dar nu este obligatoriu acest lucru) alipite toate pentru a forma o singură imagine (un singur fișier). Dacă o pagină web conține foarte multe referințe spre imagini, browser-ul este nevoit pentru randarea corectă a paginii să facă request-uri multiple pentru fiecare dintre aceste imagini. Viteza de încărcare a paginii poate avea de suferit, ținând cont că fiecare request poate presupune realizarea unei noi conexiuni TCP, transferul între serverul web și browser a unor antete adiționale pentru fiecare request (așa numitele antete HTTP), în plus fiecare fișier imagine în funcție de format mai conține un antet cu diferite date (pe lângă informația din imaginea propriu-zisă). Scopul folosiri sprite-urilor CSS este acela de a reduce viteza de încărcare a paginii, dar se pot folosi și in alte scopuri, precum crearea animațiilor.

Sprite-ul se setează de obicei ca și background pe un container, folosind atributele CSS `background-image` și `background-position`. Găsiți aici un exemplu de utilizare „drăguț” care folosește sprite-uri CSS, atribute experimentale prefixate și animații CSS.

Lectură suplimentară

Despre topicurile de mai jos mi-ar fi plăcut să mai scriu, dar din lipsa de timp mă opresc aici... ☹

- Display: grid, flex (folosite pentru elemente de tip container - elemente HTML care conțin alte elemente cum este de exemplu un div)
- Responsive Web Design
- Animații CSS
- CSS frameworks: Blueprint, Bluetrip, Bootstrap, Susy, Kube, Elasticss, Pure CSS, Foundation, Material, Semantic UI, Cascade (un framework CSS este alcătuit dintr-o colecție de clase CSS folosite in general pentru a standardiza interfețe, recicla codul CSS, realiza pagini Web responsive mai ușor, crește productivitatea).
- Preprocesoare CSS: Less, Sass

Bonus – alte exemple „nice”

Exemplul 1

Se dă codul codul HTML de mai jos. Cum credeți că arată acesta randat în browser (fără a vă uita la codul CSS)? (Credit: Baba Radu Adrian, Informatica Română, promoția 2014-2017).

```
<link href="style.css" type="text/css" rel="stylesheet" />
<ul class="hand">
    <li class="ace hearts"></li>
    <li class="ace spades"></li>
    <li class="king hearts"></li>
    <li class="ace clubs"></li>
    <li class="ace diamonds"></li>
</ul>
```

Pentru vizualizarea exemplului de mai sus "in action" click aici.

Exemplul 2

Am promis la cursul de Rețele de Calculatoare că revenim și la Programare Web la un nou episod Star Wars ☺

Star Wars Intro Scroller in Pure CSS without Javascript

May the force be with you!

Sunt deschis la orice sugestii de îmbunătățire a acestui material, observații privind greșeli… Mulțumesc.