

Am menționat mai devreme în curs că fiecare element de pe o pagină HTML poate fi reprezentat printr-un cadru (engl. *box*) imaginar trasat în jurul lui și că există elemente *Inline* și *Block*. În această lecție, ne vom ocupa în detaliu de aceste noțiuni, precum și de modul în care putem gestiona aceste cadre din jurul elementelor prin [CSS box model](#).

**IMPORTANT:** Nu confundați noțiunile de **elemente Inline** și **stilizare CSS Inline**. Elementele *Inline* reprezintă un tip de element de pe paginile HTML, în timp ce stilizarea CSS *Inline* reprezintă un mod de a posta stilizarea CSS pe orice element (mai multe detalii despre acest tip de stilizare se pot găsi în lecția anterioară).

## Bazele layout-ului (layout)

Dacă lucrăm în Adobe Illustrator, putem aranja cadrele (layouts) pentru text cum vrem, mutându-le în document. În Photoshop, putem desena pe straturi (layers) oriunde vrem. În Microsoft Word, putem muta imaginile dintr-un loc în altul prin glisare, și chiar am putea crede că acest lucru funcționează și în HTML, și CSS. Adevărul este că ordonarea elementelor este foarte diferită. Aici, ordonarea este un joc care constă în împingere și stivuire. Putem face o analogie cu clădirea unui perete de cărămidă, unde trebuie să avem un rând precedent (sau cel puțin o cărămidă) ca suport pentru următorul rând. Diferența constă în faptul că, în cazul nostru, cărămidile sunt aranjate de sus în jos, însă principiul de bază este identic.

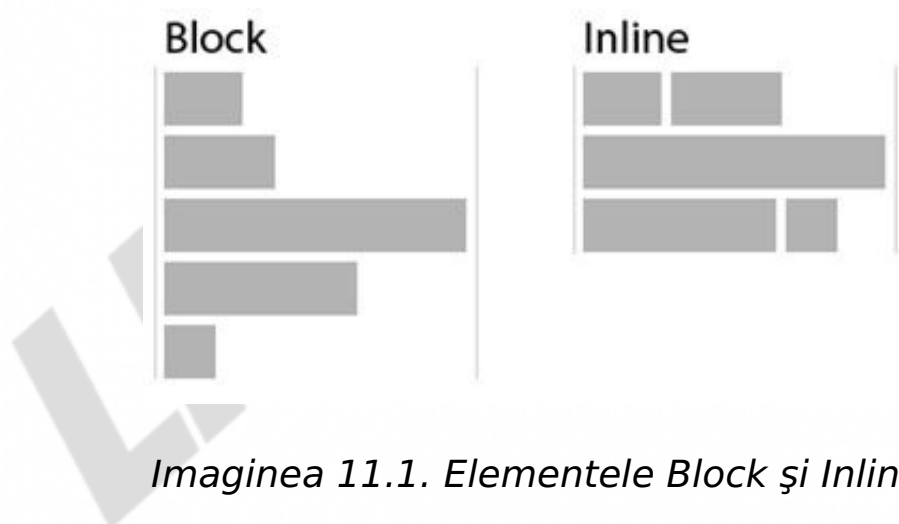
Dacă nu înțelegeți și nu acumulați cunoștințele legate de organizarea elementelor și despre modul de funcționare a combinațiilor de limbaje HTML/CSS, veți avea mari probleme mai târziu, legate de transformarea designului ideal pe care l-am desenat în Photoshop (Fireworks, Illustrator etc.) într-un document HTML funcțional.

**Elementele *Block* sunt aranjate unul sub altul, în timp ce elementele *Inline*, unul lângă altul**

Toate elementele vizibile de pe o pagină, de la imagini și paragrafe până la linkuri, sunt localizate în cadre invizibile (în limba engleză, se folosește termenul *box*).

Aceste cadre apar în două variante: *Block* și *Inline*. Diferența dintre aceste elemente constă în comportamentul lor pe pagină:

- Elementele **Block** sunt dreptunghiuri, care tind să umple întregul spațiu orizontal.
- Elementele **Inline** sunt dreptunghiuri, care se potrivesc cu alte elemente.



*Imaginea 11.1. Elementele Block și Inline*

În ilustrația de mai sus, putem vedea în ce constă diferența dintre aceste elemente. **Elementele Block se aranjează unul sub altul**, indiferent de lățimea lor. **Elementele Inline se aranjează unul lângă altul**, precum literele într-un text.

Dacă nu le definim altfel, elementele *Block* încep într-un rând nou, ocupă cât mai mult spațiu pe orizontală și mută elementul precedent și pe cel următor deasupra sau sub ele. Practic, *împing* toate elementele aflate lângă ele dedesubt și rămân singure pe rândul respectiv. Elementele *Block* reprezintă instrumentul principal de a ordona

paginile. Înălțimea depinde de conținutul inserat (în caz că nu este definit altfel). În următorul material video puteți vedea cum se comportă elementele Block și Inline pe o pagină HTML:

### *Material video 11.1 Comportamentul elementelor Block și Inline*

Elementele *Inline* se bazează pe formatarea textului și astfel se și scriu. Dimensiunile lor depind doar de conținutul din cadrul lor. Dacă definim lățimea unui element *Inline*, de exemplu la 200 px, nu se va întâmpla nimic, lățimea lui va depinde în continuare de lățimea conținutului. Contrar acestui lucru, adăugarea de text va mări lățimea, după cum puteți vedea în materialul video, câmpurile cu mai mult text erau, totodată, și mai late în afișare.

În mod implicit, fiecare element de pe pagină (în cadrul secțiunii `<body>`) este element *Block* sau *Inline*. Totuși, cu ajutorul lui CSS, putem schimba un element *Inline* într-unul *Block* sau invers, precum și alte tipuri de elemente, pe care le vom aborda în detaliu în cadrul următoarelor cursuri. De exemplu, elementele de pe o listă neordonată (*Block*) se pot transforma în elemente *Inline* și astfel obținem un rând de text cu elemente puse unul lângă altul. De asemenea, putem schimba un șir de linkuri dintr-un text (*Inline*) în elemente *Block*, pentru a obține linkuri aranjate pe verticală.

Prin urmare, **fiecare element vizibil poate fi Block sau Inline**, întrebarea este doar dacă avem nevoie de așa ceva într-un anumit context.

Conform valorilor predefinite, implicite, în elementele **Block** intră:

Elemente <i>div</i>	<code>&lt;div&gt;</code>
Paragrafe	<code>&lt;p&gt;</code>
Liste	<code>&lt;ul&gt;</code>
Elementele listelor	<code>&lt;li&gt;</code>

Titluri	<h1> - <h6>
Tabele	<table>
Elementele HTML5 de bază	<section>, <aside>, <nav>, <header> și <footer>
Tag-ul body	<body>

În mod implicit, în elementele **Inline** intră:

Elemente <i>span</i>	<span>
Linkuri	<a>
Formatarea bold	<strong> sau <b>
Formatarea italică	<em> sau <i>
Imagini	<img>
Citate	<cite>
Etichete	<label>

Elementele (tag-urile) din partea head a paginii, precum <script>, <meta>, <link> și altele, nu sunt nici *Inline*, nici *Block*, deoarece nu sunt vizibile pe pagină și nu respectă aceste reguli.

## Gruparea elementelor

### Gruparea mai multor elemente într-un singur element Block (DIV)

Elementul <div> (prescurtarea de la cuvântul englez division) ne

permite să grupăm mai multe elemente într-un singur element *Block*. De exemplu, putem să creăm un *div* pentru antetul paginii și să punem în el toate elementele antetului (logo, slogan, navigare, căutare).

Tag-ul Div, începe pe un rând nou. În afară de aceasta, el nu influențează aspectul paginii. În mod implicit, *div*-urile nu au niciun fel de stilizare (fundalul este transparent, *border*, *margin* și *padding* sunt 0, înălțimea depinde de conținut și așa mai departe). Totuși, le putem adăuga valori ID și/sau CLASS și să le țintim cu regulile CSS. Această simplitate de bază, pe de o parte, și marile posibilități de stilizare, pe de altă parte, fac din **elementele div structura de bază a fiecărui site**.

Să vedem cum arată acest lucru într-un exemplu. În imaginea de mai jos este prezentată structura primei pagini de pe site-ul Wikipedia, pe ea fiind marcat cu roșu *div*-ul principal și elementele paginii, în timp ce cu albastru sunt marcate conținuturile din cadrul elementelor *div*.



## Imaginea 11.2 Structura elementelor div – exemplu Wikipedia

După cum puteți vedea, elementul *div* (*box*) poate să conțină mai multe elemente diferite (inclusiv alte elemente *div*), creând astfel o ierarhie care reprezintă elementele încorporate ale paginii. În această ierarhie, fereastra browserului reprezintă elementul *Root* (rădăcină).

În partea anterioară a acestui capitol, am explicat cum, cu ajutorul elementelor *div*, putem forma unități pe site. Acum, pe un exemplu vom arăta cum ne poate facilita un *div* stilizarea majorității elementelor de pe pagină. Ca exemplu vom lua următorul cod HTML:

```
<div id="text">
```

```
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam dapibus  
    nisl eget neque semper, eu fermentum erat accumsan. Nunc ullamcorper mollis lacus  
    non eleifend. Vestibulum suscipit lobortis massa, sit amet tristique neque pulvinar  
    sit amet.</p>
```

```
    <p>Cras sodales lacinia gravida. Ut eget mollis risus, at pulvinar orci.  
    Vestibulum sit amet arcu in eros tincidunt aliquet vel ac tortor. Nunc congue id  
    felis gravida hendrerit. Etiam viverra velit eget eleifend suscipit. </p>
```

```
    <p>Sed sit amet blandit libero. Maecenas dapibus est nisl, posuere scelerisque  
    orci vulputate a. Curabitur quis tincidunt tortor. Morbi sit amet iaculis nunc.  
    Maecenas dignissim lorem nec malesuada fringilla. Pellentesque libero orci,  
    tincidunt non congue id, vehicula at odio. </p>
```

```
</div>
```

Putem observa că, în cadrul acestui element *div*, avem trei paragrafe de text. Acum, de exemplu, vrem să schimbăm fontul, culoarea textului și să afișăm textul în fontul italic. Folosind elementele *div* putem influența toate cele trei paragrafe simultan, iar codul CSS ar fi următorul:



```
#text {  
  
font-family: "Times New Roman";  
color: blue;  
font-style: italic;  
  
}
```

În acest fel scurtăm codul HTML și accelerăm stilizarea documentului aplicând, pe lângă aceasta, același cod CSS asupra tuturor paragrafelor în cadrul elementului `div`. Dacă nu folosim tag-ul `div`, atunci trebuie să definim separat clasa pentru aceste paragrafe și să facem, tot separat, și stilizarea.

### **Gruparea mai multor elemente într-un singur element *Inline* (SPAN)**

Elementul `<span>` este echivalentul *Inline* al elementului `<div>`. Cu ajutorul lui putem grupa mai multe elemente *Inline* într-unul singur. De obicei, se folosește pentru a separa părțile din text, deoarece putem, de exemplu, să marcăm o parte din text, să adăugăm CLASS sau ID, iar apoi, cu o descriere CSS să influențăm partea respectivă a textului (care este înconjurată de *span*). Să vedem acest lucru într-un exemplu. Vrem să obținem această afișare pe pagină:

`div` and `span` elements are **very important** to know!

#### *Imaginea 11.3 Aplicarea elementului span*

Problema este următoarea. Nu putem stiliza doar o parte din textul din paragraf, iar dacă adăugăm un nou paragraf, textul va trece într-un rând nou, deoarece, după cum am spus mai devreme, paragraful în

HTML este un element Block, ceea ce înseamnă că va trebui să treacă într-un rând nou. Aici, pe scenă, intră elementul span, acesta fiind un element Inline care nu va perturba structura textului din paragraf. Codul pentru acest exemplu ar fi următorul:

```
<p>  
  div and span elements are <span style="font-weight: bold;  
  color:red;">very important</span> to know!  
</p>
```

După cum puteți vedea, în cadrul tag-urilor paragraph și în cadrul textului, în zona în care vrem să schimbăm afișarea, deschidem tag-ul span și, în acest caz, adăugăm atributul style, precum și codul CSS pentru îngroșarea textului și schimbarea culorii, iar apoi adăugăm și textul propriu-zis, după care închidem tag-ul span.

Pentru acest exemplu, pentru o înțelegere mai bună, am folosit stilizarea Inline a elementului span, iar când vom lucra pe pagina HTML, să recomandă să atribuim fie id-ul, fie clasa și să scriem codul CSS într-un fișier CSS separat.

Ceea ce este specific pentru span este faptul că el nu poate conține elemente Block, ci doar alte elemente Inline. În specificația W3C pentru HTML se scriu următoarele:

*Generally, block-level elements may contain inline elements and other block-level elements. Generally, inline elements may contain only data and other inline elements. Inherent in this structural distinction is the idea that block elements create "larger" structures than inline elements.*<sup>1</sup>

Prin urmare, **elementele Block pot conține atât elemente Inline, cât și Block, în timp ce elementele Inline pot conține doar alte**



## elemente Inline.

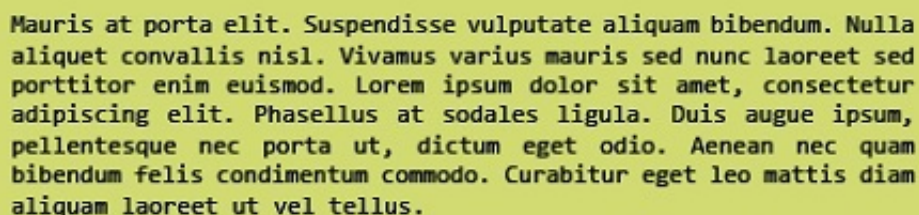
## Modelul CSS Box

După cum am menționat deja, în jurul fiecărui element vizibil de pe pagină există un cadru (*box*) imaginar pe care îl putem stiliza. Dacă am împărți afișarea unui element în etape, am obține următoarea ordine: HTML setează conținutul și creează un cadru imaginar. CSS stilizează cadrul respectiv, îl modifică după nevoie, pentru ca, în final, să afișeze utilizatorului elementul respectiv.

Modelul *Box* este alcătuit din 5 proprietăți de bază ale elementelor. Acestea sunt:

- **Width** (lățimea),
- **Height** (înălțimea),
- **Margin** (marginea sau spațiul extern),
- **Padding** (padding sau spațiul intern),
- **Border** (bordură, cadru),
- Deși nu intră în modelul CSS *Box*, deseori se adaugă și proprietatea **Background** (fundalul).

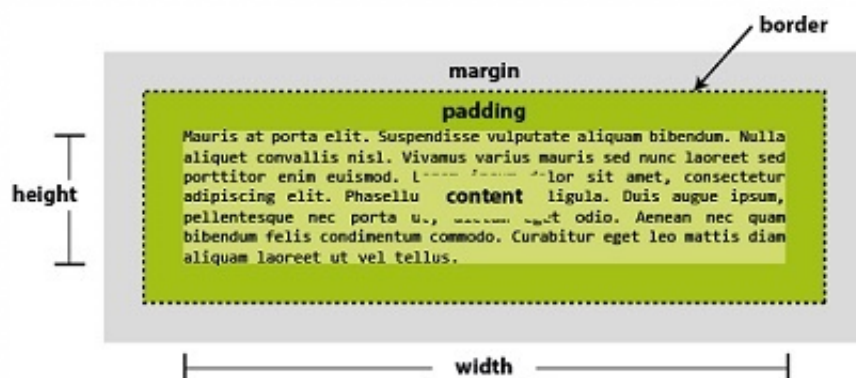
Iată un exemplu în acest sens. Mai jos se află un paragraf de text stilizat:



Mauris at porta elit. Suspendisse vulputate aliquam bibendum. Nulla aliquet convallis nisl. Vivamus varius mauris sed nunc laoreet sed porttitor enim euismod. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus at sodales ligula. Duis augue ipsum, pellentesque nec porta ut, dictum eget odio. Aenean nec quam bibendum felis condimentum commodo. Curabitur eget leo mattis diam aliquam laoreet ut vel tellus.

### Imaginea 11.4. Paragraf de text stilizat pe pagină

Acesta conține toate elementele modelului CSS *box* marcate în imaginea de mai jos:



### Imaginea 11.5. Același paragraf de text cu elementele modelului CSS *box* afișate

Fiecare regiune sau cadru are un anumit scop. **Marginile** sunt întotdeauna transparente și separă elementul de alte elemente de pe pagină. Cadrele (**Borders**) pot avea diferite stiluri și stabilesc granițele elementului, se aplică deseori pe paginile web. **Padding** separă conținutul (Content) unui element de cadrele sale (Border). Dacă nu folosim Padding asupra elementelor, aceste elemente pot părea neclare pe pagină. Setările elementelor de fundal se referă la aria din interiorul cadrului, împreună cu secțiunea **Padding** și **Content** (în imagine, secțiunea Padding este afișată cu verde mai închis, deși, în realitate este invizibilă).

Proprietățile Margins, Borders și Padding pot avea valori diferite pentru fiecare parte a unui element (Top, Right, Bottom, Left). În mod implicit, aceste proprietăți sunt, de obicei, 0. Marginile pot avea și o valoare negativă, deși acest lucru nu este recomandat.

Este important de reținut că valorile Width și Height determină doar lățimea, respectiv înălțimea conținutului elementului. **Înălțimea și lățimea reală** pe care elementul le ocupă pe pagină sunt **Content + Padding + Border**.

De exemplu, dacă avem un element `div` în care conținutul este de 100x100 pixeli, `Padding` este 10 px (pe toate părțile), iar `Border` este 2 pixeli (de asemenea, pe toate părțile), acesta va ocupa 124x124 pixeli. Asta se datorează faptului că lățimea totală este: **100px** pentru conținut + **10px pentru Padding x2** pentru `Padding`-ul din stânga și dreapta + **2px pentru Border x2**, deoarece avem și cadru în stânga și în dreapta ( $100 + 10 \times 2 + 2 \times 2 = 124\text{px}$ ). Același lucru este valabil și pentru înălțime.

Codul CSS pentru elementele din acest exemplu ar putea fi:

```
div{  
width:100px;  
height:100px;  
  
padding:10px;  
margin:0;  
  
border-width:2px;  
border-style:solid;  
border-color:#f90;  
}
```

În continuarea lecției, vom vedea detaliile acestor proprietăți.

## Proprietățile CSS în modelul box

### Width și Height

Cele două proprietăți reprezintă lățimea, respectiv înălțimea conținutului elementului. În mod implicit, lățimea și înălțimea sunt atât de mari cât este nevoie să înconjoare conținutul. Excepție face lățimea elementelor `Block`, care se extinde la maxim, respectiv fără proprietatea `width` definită, elementele vor ocupa întreaga lățime, de

la marginea stângă la marginea dreaptă a paginii.

Valoarea poate fi în **pixeli**, **procente** sau în **valori em**. Pixelii sunt cel mai popular mod și se folosesc în majoritatea cazurilor. Dacă folosim procente, dimensiunea va fi stabilită de dimensiunea ferestrei browser-ului, respectiv de dimensiunea elementului părinte (dacă există). La valoarea em, dimensiunea depinde de dimensiunea textului (fontului) din cadrul lui:

```
.myDiv {  
width:400px;  
height:80%;  
}
```

Atenție la următorul lucru. **Prin definirea proprietăților height și width, noi doar definim dimensiunile conținutului în cadrul elementului (vezi imaginea 11.5)**, însă dacă avem proprietăți suplimentare, cum sunt border și margin, și acestea trebuie luate în considerare atunci când setăm aspectul elementului propriu-zis pe pagină. Prin urmare dacă, de exemplu, spunem că div-ul are lățimea de 400px, aceasta este doar dimensiunea părții interne, concret, a unui conținut, cum ar fi textul sau imaginile în div, însă dacă avem definit elementul margin de 100px, elementul nostru va ocupa, vizual, mai mult spațiu pe pagină decât cei 400 pixeli definiți. Mai multe detalii despre categoriile width și height, precum și despre proprietăți, găsiți în materialele video ale acestei lecții, așadar, după ce studiați lecțiile scrise, vizionați obligatoriu și materialele video în care, prin exemple, prezentăm modul în care aceste elemente influențează conținuturile, dar și pagina propriu-zisă.

## Padding

Această proprietate definește spațiul intern (*Padding*). Dacă nu punem

nicio valoare, cea implicită este 0. Dacă proprietatea nu conține niciun sufix, atunci aceasta se aplică pe toate cele patru părți.

```
...  
padding:10px;  
...
```

Dacă vrem valori Padding diferite sus, la dreapta, la stânga și jos, putem scrie, de exemplu:

```
...  
padding-top:10px;  
padding-right:15px;  
padding-bottom:20px;  
padding-left:25px;  
...
```

Există și un mod prescurtat de scriere (shorthand), cu ajutorul căruia într-un rând, putem folosi doar proprietatea *Padding* (fără sufix), pentru a determina toate cele 4 valori. La proprietatea *Shorthand padding*, putem să scriem:

```
...  
padding:10px 15px 20px 25px;  
...
```

O declarație scrisă astfel va avea același rezultat ca și cele patru de mai sus. Este important de menționat doar că valorile se scriu **întotdeauna** în sensul acelor de ceasornic, începând de sus. În exemplul nostru, 10 px este valoarea *Padding*-ului de sus, 15 px a celui

din dreapta, 20 px a celui de jos, iar 25 px a celui din stânga.

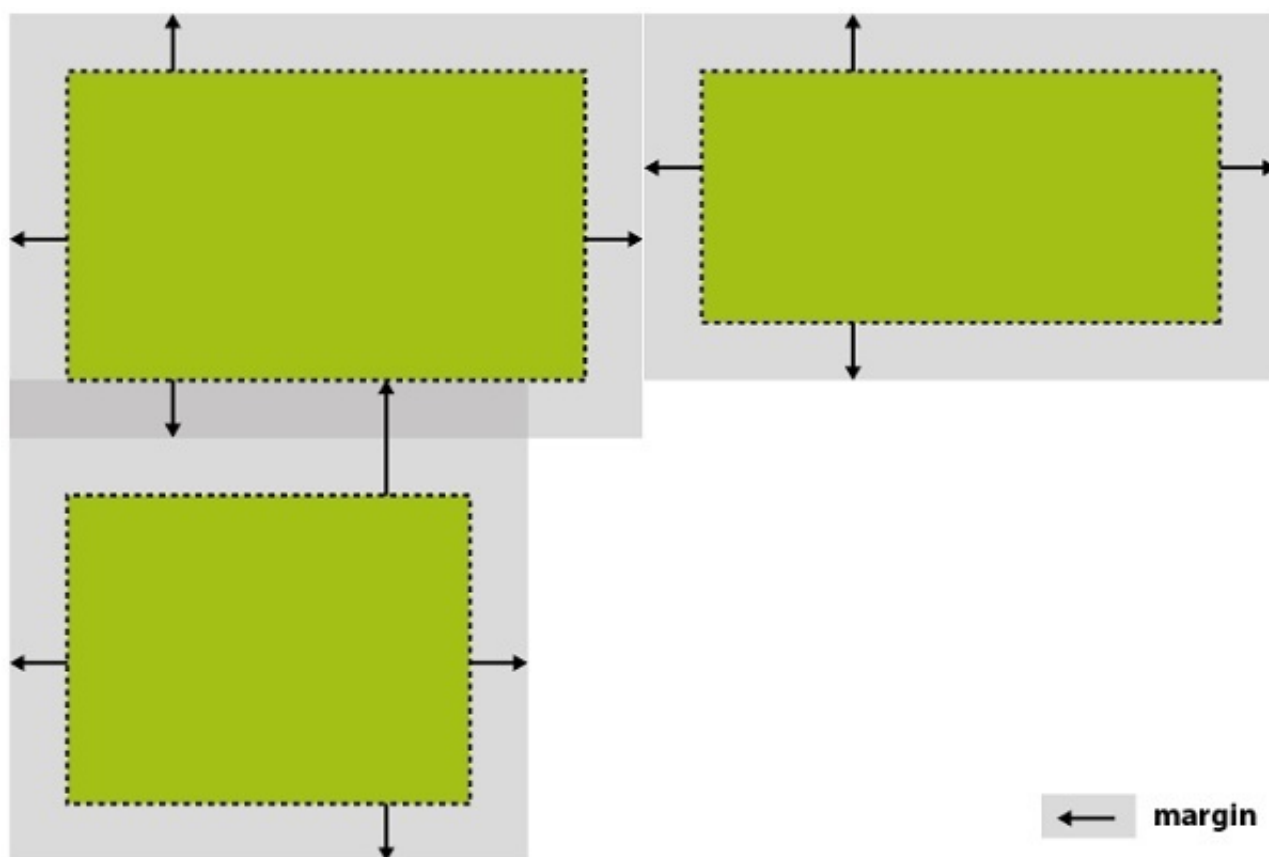
## Margin

Această proprietate definește spațiul extern (Margin). Toate regulile valabile pentru Padding sunt valabile și pentru Margin. Valoarea implicită este 0 la elementele div. Putem să scriem proprietatea generală (Margin) sau să folosim aceleași sufixe pentru părți ca și la Padding sau putem să scriem și proprietatea shorthand (prescurtată).

```
...  
margin:100px;  
/* toate identice */  
.....  
margin-top:100px;  
margin-right:30px;  
margin-bottom:200px;  
margin-left:25px;  
.....  
margin:100px 30px 200px 25px;  
...
```

În ceea ce privește marginile, este important de știut că ele sunt **pliabile (collapse)**, respectiv dacă două elemente se află unul sub altul și dacă există margini între ele (marginea de jos a elementului de sus și marginea de sus a elementului de jos), spațiul total **nu va fi** egal cu suma marginilor; va fi aplicată cea mai mare valoare dintre cele două. În opoziție cu acest lucru, dacă elementele se află unul lângă altul, marginile nu se pliază, ci sunt adunate.





*Imaginea 11.6. Ilustrarea marginii*

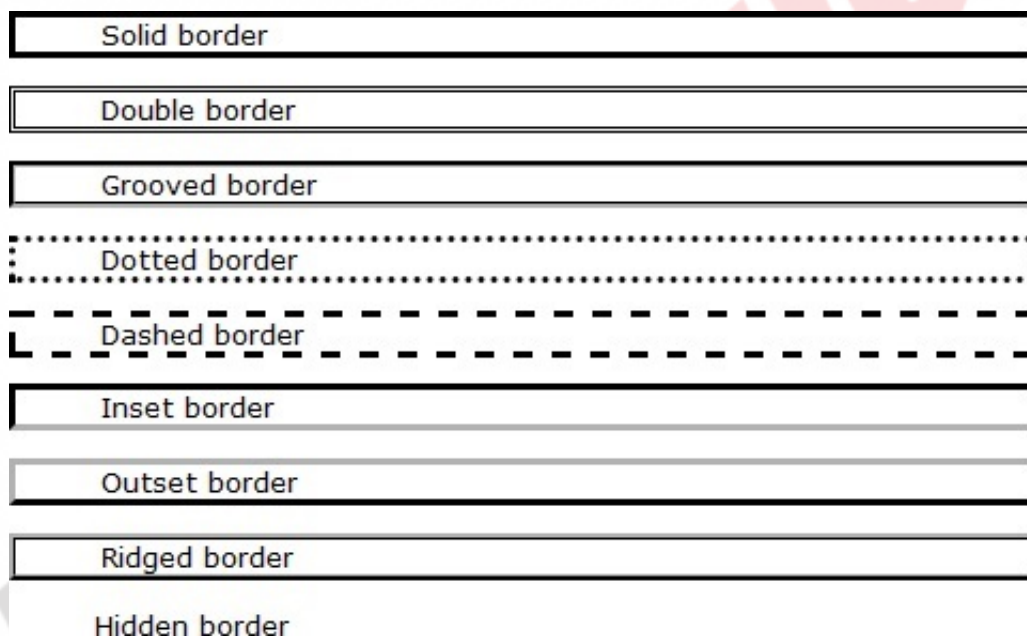
## Border-width

Folosim această proprietate pentru a defini grosimea marginii din jurul elementului. Putem folosi pixeli sau una dintre următoarele trei valori: *thin*, *medium*, *thick*. În acest caz, procente și alte unități nu sunt permise. La fel ca la margin și padding, valoarea border se aplică tuturor celor patru părți. Valorile diferite le putem scrie prescurtat (shorthand) sau separat, de exemplu:

```
...  
border-top-width: 6px;  
border-right-width: medium;  
border-bottom-width: 3px;  
border-left-width: thin;  
...
```

## Border-style

Această proprietate se folosește pentru a determina tipul de bordură, cadrul (border) din jurul elementului. În imaginea de mai jos, putem vedea opțiunile posibile. În general, se folosește *solid* (o linie simplă). Valoarea este propriul nume al tipului: solid, double, grooved etc.



*Imaginea 11.7. Tipuri de border*

Dacă toate marginile elementului sunt identice, scriem:

```
...  
border-style:solid;  
...
```

De asemenea, putem determina și tipul marginii, folosind:

```
...  
border-top-style: solid;
```

```
border-right-style: groove;  
border-bottom-style: dotted;  
border-left-style: outset;  
...
```

## Border-color

După cum îi spune numele, această proprietate definește culoarea marginii. Valoarea culorii poate fi scrisă în diferite moduri, cu ajutorul sistemului hex, RGB sau cu ajutorul numelor culorilor (despre culorile pe web vom discuta într-o lecție separată).

```
border-color: #FF9900;
```

## Alte proprietăți importante

### Display

Mai devreme, am menționat că elementele *Block* se pot transforma în elemente *Inline* și invers. Tipul de element poate fi determinat folosind proprietatea **Display** prin CSS. Dacă, de exemplu, la valoarea **Display** a unui element *Inline* îi atribuim valoarea *Block*, atunci acest element va fi tratat ca oricare alt element *Block*.

```
span {  
  display: block;  
}
```

Cu acest exemplu, am transformat elementele *span* în elemente *Block*.

În afară de valorile Block și Inline, pe care le folosim pentru a schimba tipul, proprietatea Display poate avea și valoarea none. În acest caz, dacă aplicăm *display:none*, browserul se va comporta ca și când elementul nu ar exista, deși acesta se va afla în cod. De asemenea, orice element încorporat în cadrul elementului respectiv nu va fi afișat, deși îi este acordată o altă valoare Display.

De exemplu, tag-ul span este un element Inline. Dacă atribuim valoarea display:block, practic aceasta se va comporta ca un div. Pe de altă parte, dacă postăm display:none, span-ul menționat nu va fi vizibil. Totuși, trebuie să reținem faptul că, datorită acestei opțiuni nu eliminăm în totalitate elementul, ci doar ascundem afișarea lui, însă utilizatorul va putea citi întotdeauna codul-sursă HTML, pe care îl conține partea respectivă. Datele sensibile nu ar trebui ascunse în acest mod.

La proprietatea Display trebuie să rețineți un lucru important. **Prin aplicarea proprietății display vom schimba doar modul în care va fi afișat elementul, nu și tipul lui în HTML.** Prin urmare, de exemplu, dacă atribuim elementului span, care este unul Inline, display:block; acesta se va comporta, vizual, ca un element block, însă în el putem introduce și alte elemente block, deoarece el este pentru HTML, în continuare, un element Inline.

### Notă:

În timpul lucrului puteți întâlni și elemente Inline-Block. Elementele Inline-Block, după cum le și spune numele, conțin proprietăți de la ambele tipuri de elemente. Acestora li se poate seta înălțimea și lățimea, spre deosebire de elementele Inline și, de asemenea, ele nu întrerup linia, spre deosebire de elementele block. În esență, acestea sunt elemente Inline care pot conține elemente Block, cărora li se poate seta și dimensiunea. Cel mai des se aplică în urma creării meniurilor orizontale, pentru nevoile linkurilor de navigare, deoarece permit introducerea elementelor block pe orizontală, unul lângă altul. În cadrul acestui curs introductiv, nu vom vorbi în detaliu despre aceste elemente, având în vedere că necesită setări suplimentare în CSS și că pot provoca și probleme în ceea ce privește afișarea în

câteva browsere mai vechi.

## Visibility

Proprietatea *visibility* asigură ascunderea unui anumit element, în timp ce spațiul rezervat pentru elementul respectiv rămâne. Poate avea una dintre următoarele două valori: *hidden* (ascunde elementul) sau *visible* (afișează elementul - aceasta este valoarea implicită și nu trebuie să o scriem).

```
span {  
  visibility:hidden;  
}
```

Diferența dintre *visibility:hidden;* și *display:none;* constă în faptul că, în primul caz, elementul dispare, însă rămâne un spațiu rezervat pentru el, iar celelalte elemente nu se mută, în timp ce, în al doilea caz, elementul dispare de pe pagină și odată cu el și spațiul rezervat, așadar și celelalte elemente se mută pe pagină. Vă reamintim că, atunci când se utilizează aceste două tehnici, elementele sunt ascunse utilizatorului doar din punct de vedere vizual, rămânând în continuare disponibile în cod, așadar nu trebuie să ascundem date sensibile în acest mod.

## Overflow

Această proprietate se folosește pentru a defini ce se întâmplă în cazul în care conținutul depășește cadrele date. De exemplu, putem să definim lățimea și înălțimea unui div la 200x300 px, iar apoi să introducem întreaga pagină de text. Deoarece dimensiunile definite

sunt mai mici decât cele necesare, textul întreg nu va fi vizibil (partea care depășește div-ul). Totuși, folosind proprietatea `overflow`, putem să definim cazul în care conținutul din cadrul elementului se derulează utilizând `overflow:scroll`; . O altă valoare posibilă este `overflow:hidden`;, care va tăia partea ce iese în afara cadrului. Valoarea implicită este `overflow:visible`;

```
div.myTekst {  
  overflow:scroll;  
}
```

În trecut, această proprietate se folosea mai mult pentru introducerea textelor mai lungi în niște cadre limitate. Acum, odată cu dezvoltarea web-ului, proprietatea a devenit inutilă, chiar inefficientă. Totuși, o caracteristică secundară a acestei proprietăți a devenit utilă pe site-urile moderne, însă despre ea vom vorbi mai târziu, în cadrul lecției care abordează aranjarea elementelor.

## Min-width, max-width

După cum ne și spun denumirile acestor proprietăți, este vorba de o lățime minimă, respectiv maximă a elementelor. În loc să folosim dimensiuni fixe, putem să definim dimensiunea minimă și/sau maximă.

```
#wrapper {  
  min-width:960px;  
  max-width:1280px;  
}
```

Utilizarea ambelor proprietăți în același element nu este obligatorie. **Nu trebuie combinate cu lățimea fixă.**



## Min-height, max-height

Proprietățile *min-height* și *max-height* se comportă la fel ca proprietățile menționate mai sus, singura diferență constând în faptul că se referă la înălțimea elementului.

```
p {  
  min-height:10px;  
  max-height:26px;  
}
```

### Notă:

Proprietățile min și max pentru lățime și înălțime sunt importante în cazul în care fereastra motorului de căutare este mai mică decât lățimea elementului. Fără aceste proprietăți definite, pe o fereastră mică a motorului de căutare va apărea un scrollbar orizontal, deoarece div-ul continuă în afara limitelor vizibile. Acest lucru este important în caz că vrem să afișăm site-ul nostru corect pe telefoanele mobile. Având în vedere că acest lucru face parte din designul adaptat al paginilor (responsive design), în următoarele cursuri vom vorbi în detaliu despre aceste proprietăți și despre aplicarea lor.

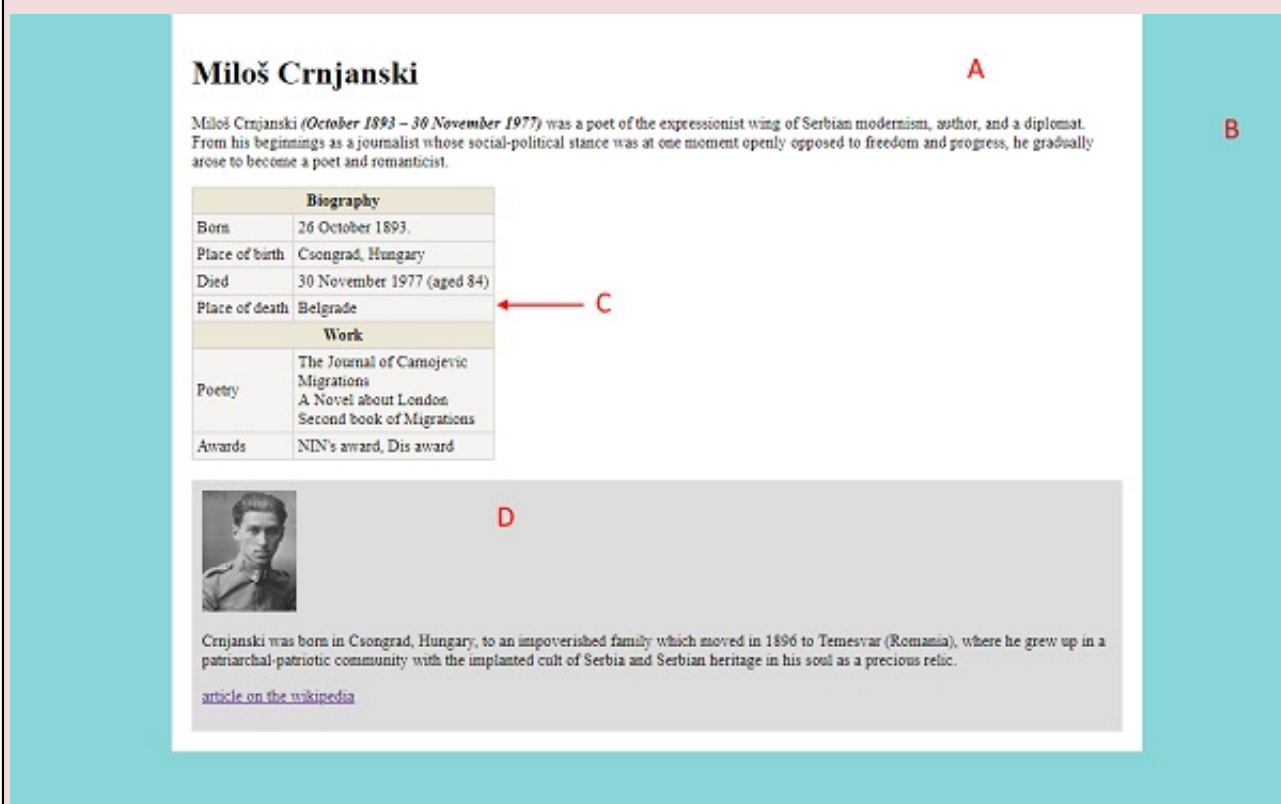
Pentru cei care vor să afle mai multe informații despre acest subiect, în materialul video de mai jos este prezentat comportamentul elementului div, care are definită lățimea fixă de 1000px și un div care are definită lățimea maximă de 1000px.

*Materialul video 11.2 Comportamentul elementelor div în raport cu proprietățile lățimii elementului*

Codul culorii de fundal este [1](http://www.w3.org/TR/html401/struct/global.html#h-7.5.3). Sursa:  
<http://www.w3.org/TR/html401/struct/global.html#h-7.5.3>

## Exercițiu:

În acest exercițiu vom crea o pagină HTML conform imaginii de mai jos, iar apoi vom stiliza această pagină folosind CSS.



*Imaginea 11.8 Exercițiu CSS*

**A - div-ul cu o culoare albă de fundal, din CSS sunt necesare următoarele:**

Lățimea bazei (partea albă) este de 960px

Codul culorii de fundal este: #FFFFFF

Padding-ul este de 20px de toate părțile

**B - culoarea de fundal a paginii, codul culorii este #89D6D9**

**C - tabelul prezentat în imagine, din CSS sunt necesare următoarele:**

Codul culorii de fundal a tag-ului th este #ECE7D7

Culoarea celorlalte celule în tabel este: #F5F4F2

Culoarea pentru border este #CCCCCC

**D - Footer-ul (subsolul) paginii (partea gri), din CSS, sunt necesare următoarele:**

Padding-ul este de 10px de toate părțile

Codul culorii de fundal a div-ului este: #DDDDDD

**Textul și imaginea se pot prelua de la următorul [link](#).**

**Notă:**

Prezentarea textului pe pagină depinde de browserul folosit și de dimensiunea ferestrei, așadar, soluția voastră nu trebuie să fie identică.

**Dacă întâmpinați dificultăți în rezolvarea acestui exercițiu sau vreți să verificați codul HTML, soluția exercițiului se poate prelua de la următorul [link](#).**

1. Sursa: <http://www.w3.org/TR/html401/struct/global.html#h-7.5.3>

Codul culorii de fundal a tag-ului th este [1](#). Sursa:  
<http://www.w3.org/TR/html401/struct/global.html#h-7.5.3>