

Până acum, în cadrul acestui modul ați putut vedea că limbajele HTML și CSS au anumite reguli pe baza cărora se efectuează aranjarea elementelor HTML pe pagină. Până acum ne-am familiarizat cu o proprietate cu care se poate influența un astfel de proces. Desigur, este vorba de proprietatea `display`. Pe lângă această proprietate, mai există unele ale căror valori influențează aranjarea elementelor.

Inițial, toate elementele din cadrul paginii HTML se aranjează conform regulilor fluxului natural (engl. *normal flow*). Practic, aceasta înseamnă că elementele block se aranjează unul sub altul, ocupând întreaga lățime disponibilă, iar elementele inline și inline-block se aranjează unul lângă altul, într-o linie orizontală. Totuși, pe lângă proprietatea `display`, mai există unele proprietăți care pot influența fluxul documentului HTML. De aceea, în această lecție vom aborda următoarele:

- poziționarea elementelor folosind proprietatea CSS de tip `position`;
- proprietățile CSS de tip `float` și `clear`;
- proprietatea CSS `z-index`;
- proprietatea CSS `overflow`;
- proprietatea CSS `visibility`;
- proprietatea CSS `box-sizing`.

## Poziționarea

Prima noțiune care va fi explicată în această lecție despre aranjarea elementelor, este noțiunea de poziționare. Fiecare element HTML poate fi poziționat într-unul din următoarele patru moduri:

- `static`;

- relative;
- absolute;
- fixed.

Modurile tocmai prezentate de poziționare sunt, de fapt, diferite valori pe care le poate avea proprietatea CSS de tip `position`.

Poziționările de tip `static` și `relative` mențin elementele într-un flux natural, în timp ce poziționările de tip `absolute` și `fixed` elimină elementul din fluxul natural al paginii.

În continuare vom explica în detaliu diferite poziționări.

## Poziționarea statică

Poziționarea statică este poziționarea implicită la elementele HTML. Prin această poziționare, elemente se aranjează conform unui flux normal al documentului. Când pentru un element spunem că **nu este poziționat**, ne referim, de fapt că elementul respectiv are poziționare statică. Pentru ilustrarea poziționării statice, implicite, va fi folosită următoarea structură HTML:

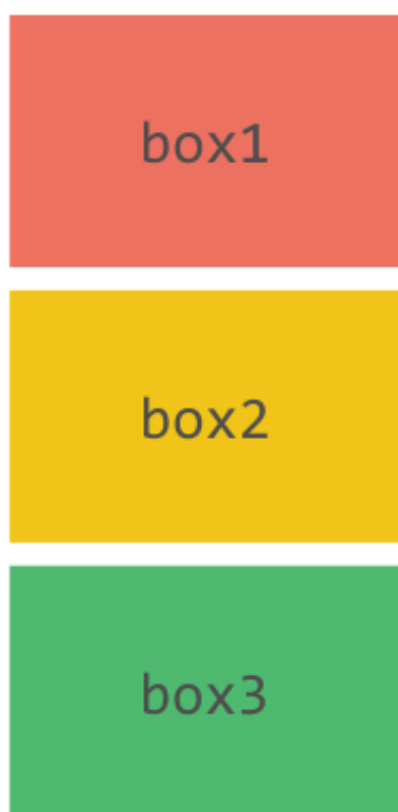
```
<div id="box1">  
</div>
```

```
<div id="box2">  
</div>
```

```
<div id="box3">  
</div>
```

Structura prezentată este formată din trei elemente `div`, aceasta este structura care va fi folosită pentru demonstrarea tuturor celorlalte poziționări. Pentru poziționarea statică a elementelor, nu trebuie setată nicio valoare a proprietății `position`, având în vedere că `static` este valoarea implicită.

Dacă elementelor `div` prezentate le introducem înălțime și lățime, margine de jos și culoare de fundal, ele vor forma pe pagină o ilustrare ca în imaginea 15.1.



*Imaginea 15.1. Trei elemente `div`, aranjate conform regulilor fluxului normal*

Având în vedere că este vorba de trei elemente `block`, ele se vor afișa în rânduri separate, unul sub altul.

## Poziționarea relativă

Elementul cu poziționare relativă este poziționat relativ în raport cu poziția sa normală, pe care ar fi avut-o dacă ar fi fost poziționat ca element static. În acest fel, poziționarea relativă nu elimină elementul din fluxul normal al documentului, ci doar asigură o influență mai precisă asupra poziției sale. Această influență asupra poziției elementului se obține prin folosirea celor patru proprietăți CSS: `top`, `right`, `bottom` și `left`. Aceste proprietăți se mai numesc și proprietăți *Offset*.

Următorul exemplu ilustrează diferența dintre poziționarea statică și cea relativă. Codul HTML din exemplu va fi identic celui anterior.

```
<div id="box1">  
</div>
```

```
<div id="box2">  
</div>
```

```
<div id="box3">  
</div>
```

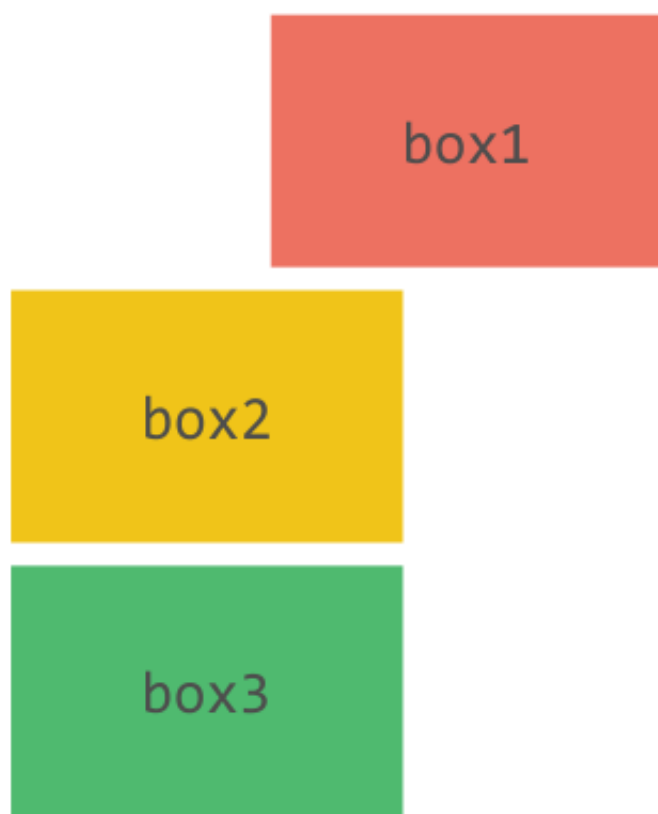
Stilizarea CSS care se referă la toate cele trei elemente div specificate este următoarea:

```
div{  
  width: 200px;  
  height: 100px;  
  margin-bottom: 20px;  
}
```

În acest fel se introduce dimensiunea elementelor `div` și marginea de jos a acestora. În continuare toate elementele sunt poziționate static. Următoarea regulă CSS setează poziționarea relativă pentru primul element `div`:

```
#box1{  
  position: relative;  
  left: 120px;  
}
```

În acest fel, elementul `div` cu id-ul `box1` obține poziționare relativă prin definirea valorii `relative` pentru proprietatea `position`. Valoarea de tip `relative` a acestei proprietăți nu are niciun efect până la specificarea uneia dintre proprietățile `offset`. În exemplu este specificată proprietatea `offset` de tip `left` și în acest fel elementul este mutat cu 120 de pixeli în dreapta, în raport cu poziția sa naturală. De fapt, valoarea de 120 de pixeli indică distanța marginii din stânga a elementului de poziția pe care s-ar fi aflat marginea sa din stânga, dacă elementul ar fi fost poziționat static. De aceea, prin specificarea valorii pozitive pentru proprietatea `left`, elementul trebuie mutat în dreapta. Imaginea 15.2. ilustrează tocmai această situație.



*Imaginea 15.2. Elementul div de tip box1 este poziționat relativ*

## Poziționarea absolută

Se spune că un element este poziționat absolut atunci când acesta este poziționat relativ cu cel mai apropiat ascendent poziționat în orice mod, în afară de utilizarea poziționării `static`. Dacă elementul nu deține niciun ascendent care nu este poziționat static, poziția elementului se stabilește pe baza elementului `body`.

Poziționarea absolută este singura poziționare care elimină elementul din fluxul normal al documentului.

Următorul exemplu ilustrează elementul `div` poziționat absolut. Codul HTML din exemplu va fi identic celui anterior, când am demonstrat poziționarea relativă.

```
<div id="box1">  
</div>
```

```
<div id="box2">  
</div>
```

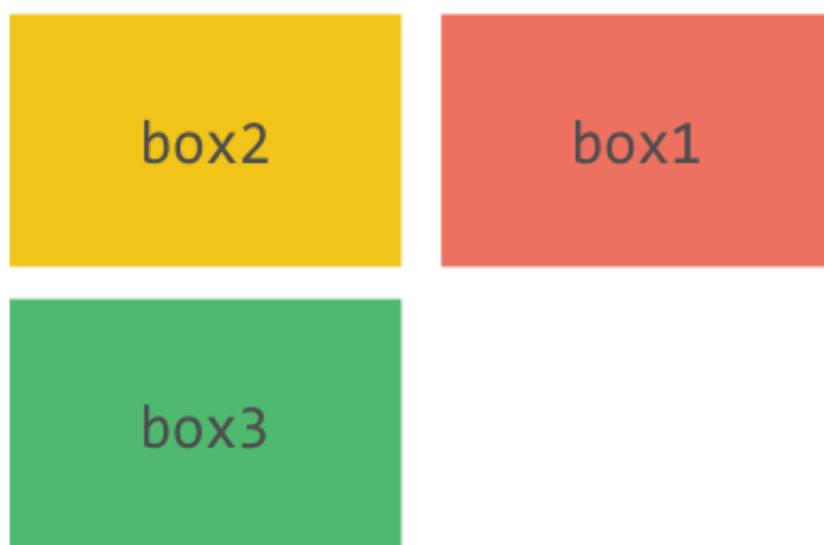
```
<div id="box3">  
</div>
```

Din cele trei elemente `div`, doar primul va fi poziționat absolut. Având în vedere că elementul menționat nu are descendenți înainte de elementul `body`, poziționarea absolută va fi efectuată în raport cu elementul `body`. Pentru ca primul `div` să se poziționeze absolut, va fi scrisă următoarea regulă CSS:

```
#box1{  
  position: absolute;  
  left: 220px;  
}
```

Ca valoare a proprietății CSS de tip `position` este setat `absolute`. În acest fel, elementul este eliminat din fluxul normal al documentului, poziția sa stabilindu-se pe baza elementului `body` și a valorii proprietăților offset (`top`, `right`, `bottom`, `left`). Pentru valoarea proprietăți `left` sunt setați 220 px, ceea ce înseamnă că spațiul de la marginea din stânga elementului `box1` până la marginea din stânga a elementului `body` va fi de 220 px.

Efectul pe care îl are poziționarea absolută este prezentat în imaginea 15.3.



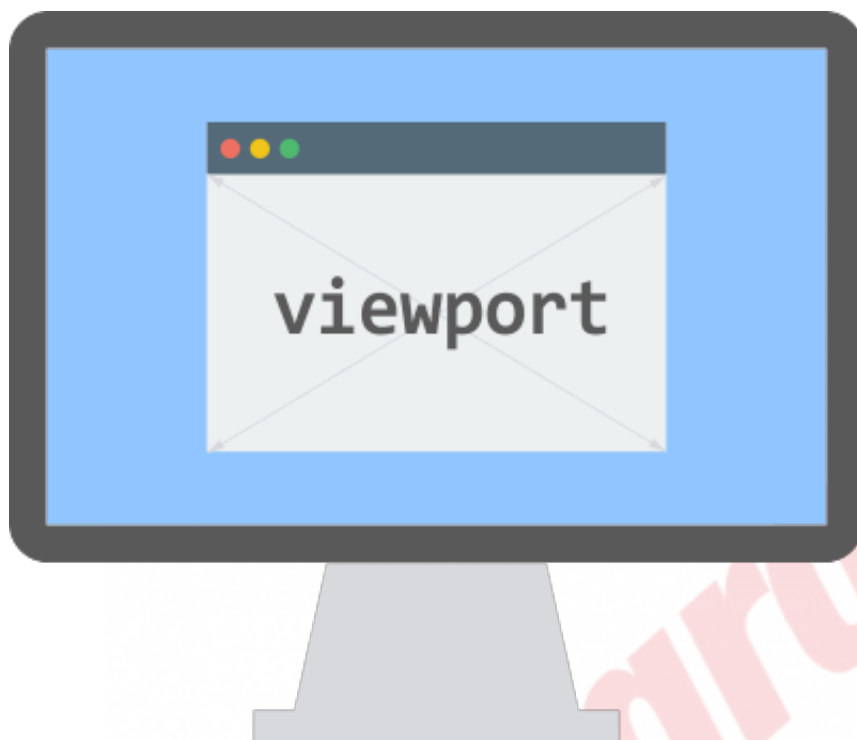
*Imaginea 15.3. Elementul div de tip box1 poziționat absolut*

Elementele poziționate absolut nu influențează celelalte elemente, ceea ce se poate vedea în imagine. Elementele `box2` și `box3` se comportă de parcă elementul `box1` nu există, așadar elementul `box2` se află pe poziția pe care s-ar fi aflat, de regulă, elementul `box1`.

## Poziționarea fixă

Poziționarea fixă asigură poziționarea elementului relativ în afișarea curentă (engl. *viewport*) a browserului. Viewport-ul se poate observa și ca un câmp de vizualizare a browserului, respectiv aceasta este zona în cadrul căreia se afișează conținutul documentelor HTML. Imaginea 15.4. ilustrează noțiunea de viewport în cadrul unui browser. Indiferent cât este de mare documentul pe care îl afișează browserul, viewport-ul are întotdeauna o dimensiune fixă, până ce se schimbă dimensiunea ferestrei browserului.





*Imaginea 15.4. Viewport-ul browserului*

Efectul poziționării fixe va fi prezentat în exemplul identic celui anterior. Structura HTML este următoarea:

```
<div id="box1">  
</div>
```

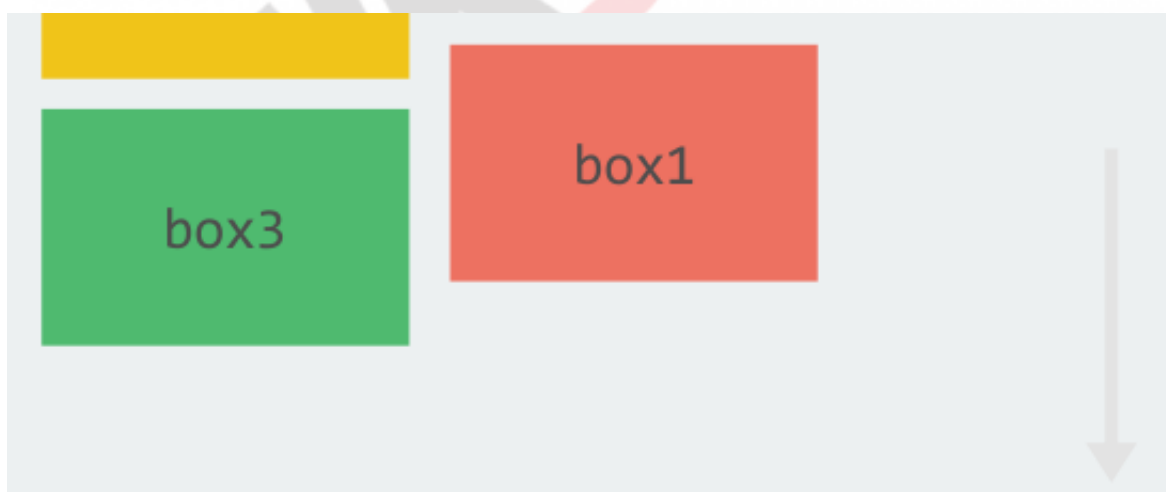
```
<div id="box2">  
</div>
```

```
<div id="box3">  
</div>
```

Elementul `div` cu `id`-ul `box1` va fi poziționat fix în felul următor:

```
#box1{  
  position: fixed;  
  left: 220px;  
}
```

Se folosește o valoare identică a proprietății `offset` de tip `left`, ca în exemplul anterior. La prima vedere, poziționarea fixă este identică celei absolute, când poziționarea absolută se referă la elementul `body`. Totuși, dacă se efectuează derularea/scroll-ul paginii, ne este clar că la poziționarea fixă elementul își menține poziția în raport cu viewport-ul. Un astfel de comportament este ilustrat în imaginea 15.5.



*Imaginea 15.5. Elementul `box1` cu poziționare fixă*

## Proprietatea `float`

După diferite moduri de poziționare a elementelor HTML pe pagină,

vom prezenta rolul pe care îl are proprietatea `float` asupra aranjării elementelor pe pagină. Folosind această proprietate, este posibilă eliminarea elementului din fluxul normal, poziționându-l de partea stângă sau de cea dreaptă, în cadrul containerului său părinte. Efectul pe care îl are proprietatea `float` va fi prezentat în exemplul următoarei structuri HTML:

```
<div id="box1">
```

```
</div>
```

```
<div id="box2">
```

```
</div>
```

```
<div id="box3">
```

```
</div>
```

```
<div id="box4">
```

```
</div>
```

```
<div id="box5">
```

```
</div>
```

```
<div id="box6">
```

```
</div>
```

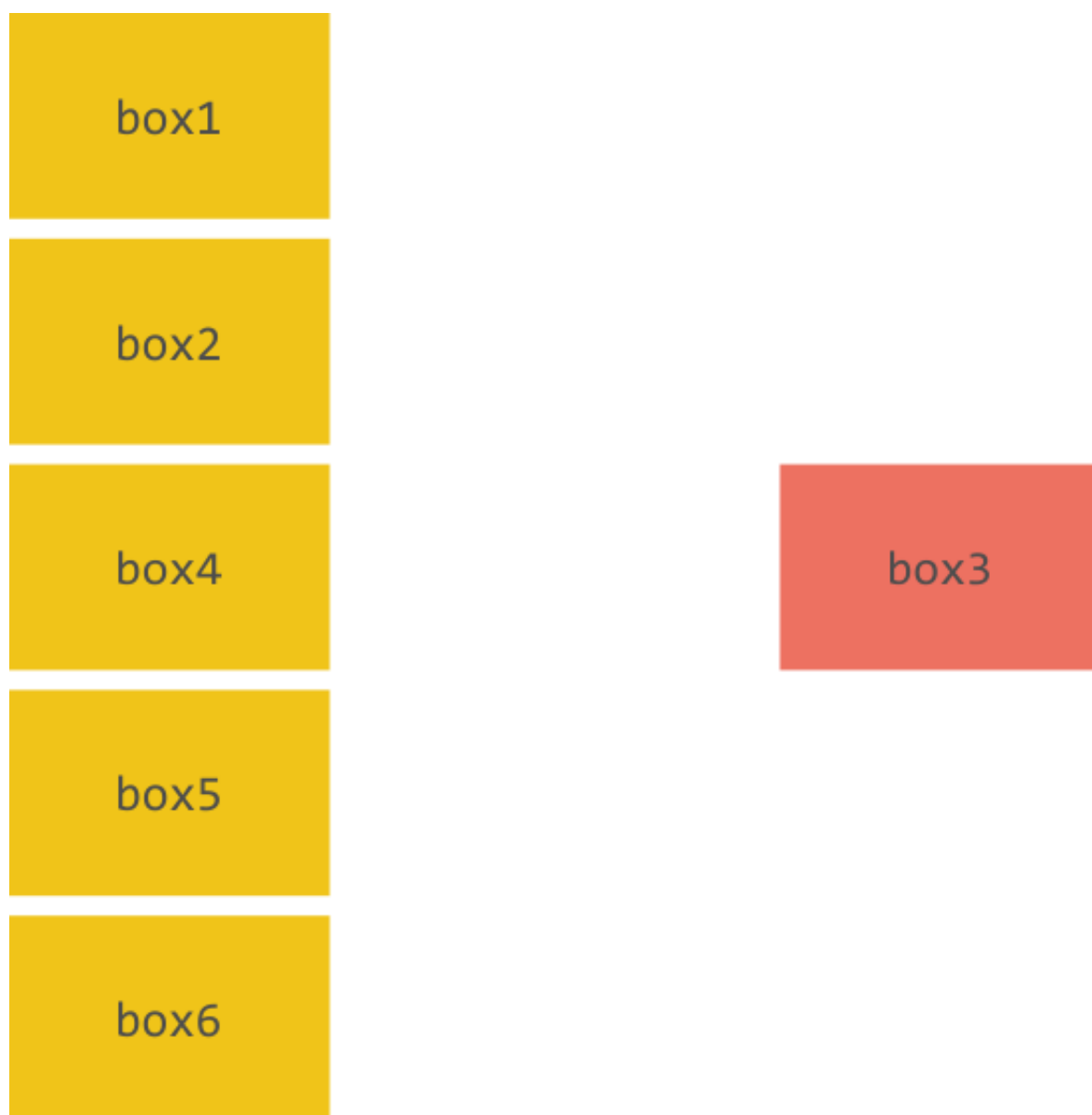
Structura prezentată este formată din 6 elemente `div`, aflate în cadrul părții `body` a paginii. Stilizarea care se referă la toate elementele `div` este următoarea:

```
div{  
width: 200px;  
height: 100px;  
margin-bottom: 20px;  
background-color:#F0C419;  
}
```

Asupra elementelor `div` este definită lățimea și înălțimea, marginea de jos și culoarea de fundal. Observați ce se va întâmpla dacă pe unul dintre elementele `div` definim valoarea proprietății `float` în felul următor:

```
#box3{  
  float: right;  
  background-color: #ED7161;  
}
```

Setarea proprietății `float` la `right` pentru elementul `box3` are un efect identic celui din imaginea 15.6.



*Imaginea 15.6. Elementul box3 cu valoarea proprietății float setată la right*

În imaginea 15.6. se poate vedea clar că prin setarea proprietății `float`, elementul a ieșit din fluxul natural al documentului. Având în vedere că proprietatea `float` este setată la `right`, elementul `box3` s-a poziționat în partea din dreapta, în cadrul containerului său părinte (în acest caz, în elementul `body`), în timp ce primul element ce urmează a ocupat locul lui.

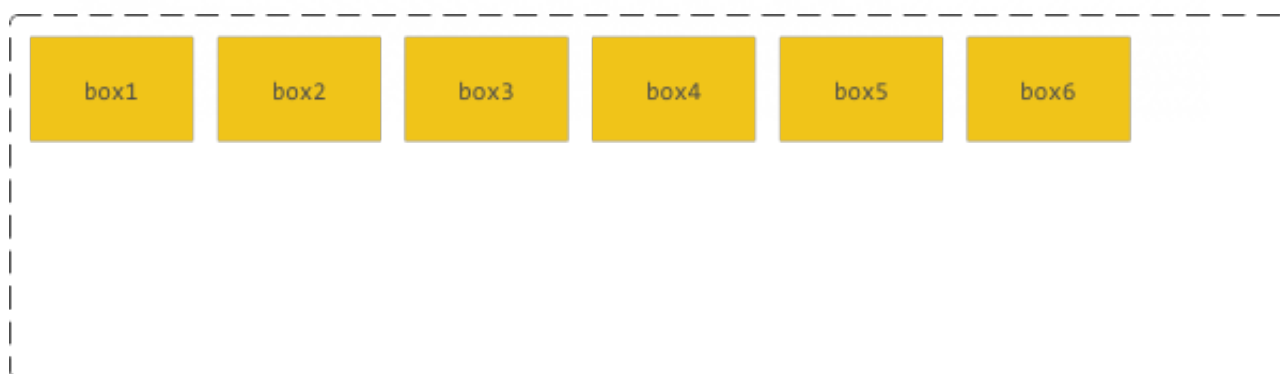
În general, proprietatea `float` poate avea trei valori diferite, și anume:

- `none`
- `left`
- `right`

Valoarea `none` este implicită și ea ne arată că elementul nu este mutat în nicio parte. `Float left` poziționează elementul în stânga în cadrul părintelui, iar `float right` îl poziționează în dreapta. Este interesant că prin utilizarea proprietății `float` se pot aranja elementele `block` unul lângă altul, într-un mod identic cu aranjarea elementelor `inline-block`. Aceasta ilustrează următorul exemplu:

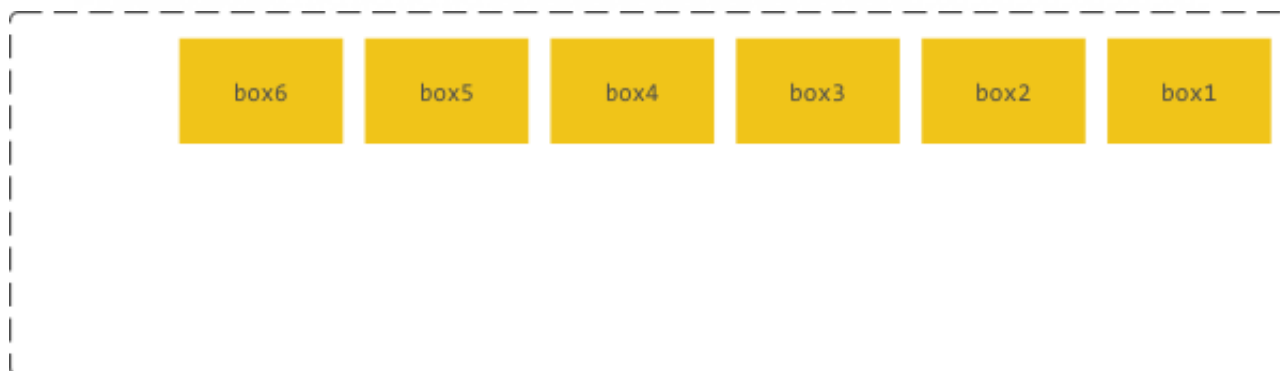
```
div{  
  margin-right: 20px;  
  float: left;  
}
```

În acest fel, toate elementele `div` vor avea valoarea proprietății `float` setată la `left`, care va avea un rezultat ca în imaginea 15.7.



*Imaginea 15.7. Elementele `div` cu proprietățile `float` setate la `left`*

Dacă toate elementele `div` ar fi avut setată valoarea proprietății `float` la `right`, s-ar fi obținut un efect identic celui din imaginea 15.8.



*Imaginea 15.8. Elementele `div` cu proprietăți `float` setate la `right`*

### **Notă**

*Elementele poziționate absolut ignoră proprietatea `float`. De asemenea, marginile verticale ale elementului floated nu se unesc niciodată cu marginile elementelor aflate deasupra sau dedesubt.*

## **Proprietatea `clear`**

Proprietatea `clear` este legată în mod direct tocmai cu proprietatea descrisă, `float`. Scopul ei este controlul comportării elementelor aflate în jurul elementelor mutate prin folosirea proprietății `float`.

Având în vedere că prin definirea proprietății `float` pe un element el va ieși din fluxul natural al documentului, celelalte elemente tind să completeze spațiul în jurul acestui element. O astfel de situație este ilustrată în exemplul cu următoarea structură HTML:

```
<div id="box1">  
</div>
```

```
<p>  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lobortis, justo ut feugiat efficitur, sapien nulla posuere dolor, vitae scelerisque lacus nunc accumsan nisl. Mauris ipsum felis, consectetur vitae tellus efficitur, lacinia dignissim nibh. Etiam mollis eget dolor in porttitor  
.  
</p>
```

Structura HTML prezentată conține un element `div` și un paragraf. Conform fluxului normal al documentului, aceste două elemente ar fi fost poziționate ca în imaginea 15.9.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lobortis, justo ut feugiat efficitur, sapien nulla posuere dolor, vitae scelerisque lacus nunc accumsan nisl. Mauris ipsum felis, consectetur vitae tellus efficitur, lacinia dignissim nibh. Etiam mollis eget dolor in porttitor.

*Imaginea 15.9. Două elemente poziționate conform fluxului normal al documentului*

Dacă elementul `div`, prin folosirea proprietății `float`, se mută în stânga, situația se schimbă, având în vedere că un astfel de element va ieși din fluxul normal, așadar, ordinea elementelor este ca în imaginea 15.10.





Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Donec lobortis, justo ut  
feugiat efficitur, sapien nulla posuere  
dolor, vitae scelerisque lacus nunc  
accumsan nisl. Mauris ipsum felis,

consectetur vitae tellus efficitur, lacinia dignissim nibh. Etiam mollis eget  
dolor in porttitor.

*Imaginea 15.10. Elementele div și paragraph, unde elementului div îi este setată proprietatea float la stânga/left*

Este evident că conținutul paragrafului tinde să înconjoare elementul div cu o valoare `float` definită. Dar ce se întâmplă dacă noi nu vrem așa ceva?

Este suficient să folosim proprietatea `clear`, ca în următorul exemplu:

```
p{  
  clear: left;  
}
```

În exemplu este folosită valoarea `left` a proprietății `clear`, având în vedere că trebuie „curățat” efectul proprietății `float` cu aceeași denumire. Proprietatea `clear` poate avea câteva valori diferite, ele fiind afișate în tabelul 15.1.

Valoarea proprietății	Descriere
left	Nu permite apariția elementelor float de partea stângă
right	Nu permite apariția elementelor float de partea dreaptă

both	Nu permite apariția elementelor float, atât de partea stângă cât și de partea dreaptă
------	---

*Tabelul 15.1. Valoarea proprietății clear*

Una dintre aplicările frecvente ale proprietății `clear` sunt situațiile în care elementul container conține exclusiv elemente mutate prin folosirea proprietății `float`. Într-o astfel de situație, elementul părinte **nu are propria înălțime**, având în vedere că aceste elemente sunt eliminate din fluxul natural al documentului. Când apare o astfel de situație, fiecare element ce urmează pe pagină, nu va lua în considerare înălțimea reală a elementelor cu proprietatea `float`, așadar, se va ajunge la suprapunerea elementelor pe pagină. Următorul exemplu ilustrează o astfel de problemă.

```
<div id="container1">
```

```
<div>
```

```
</div>
```

```
<div>
```

```
</div>
```

```
<div>
```

```
</div>
```

```
</div>
```

```
<div id="container2">
```

```
</div>
```

Structura HTML este formată din două elemente `div`. Primul `div`, cu `id`-ul `container1`, conține trei elemente `div`, în timp ce al doilea `div`,

care are id-ul `container2`, este gol. Elementele prezentate vor fi stilizate în felul următor:

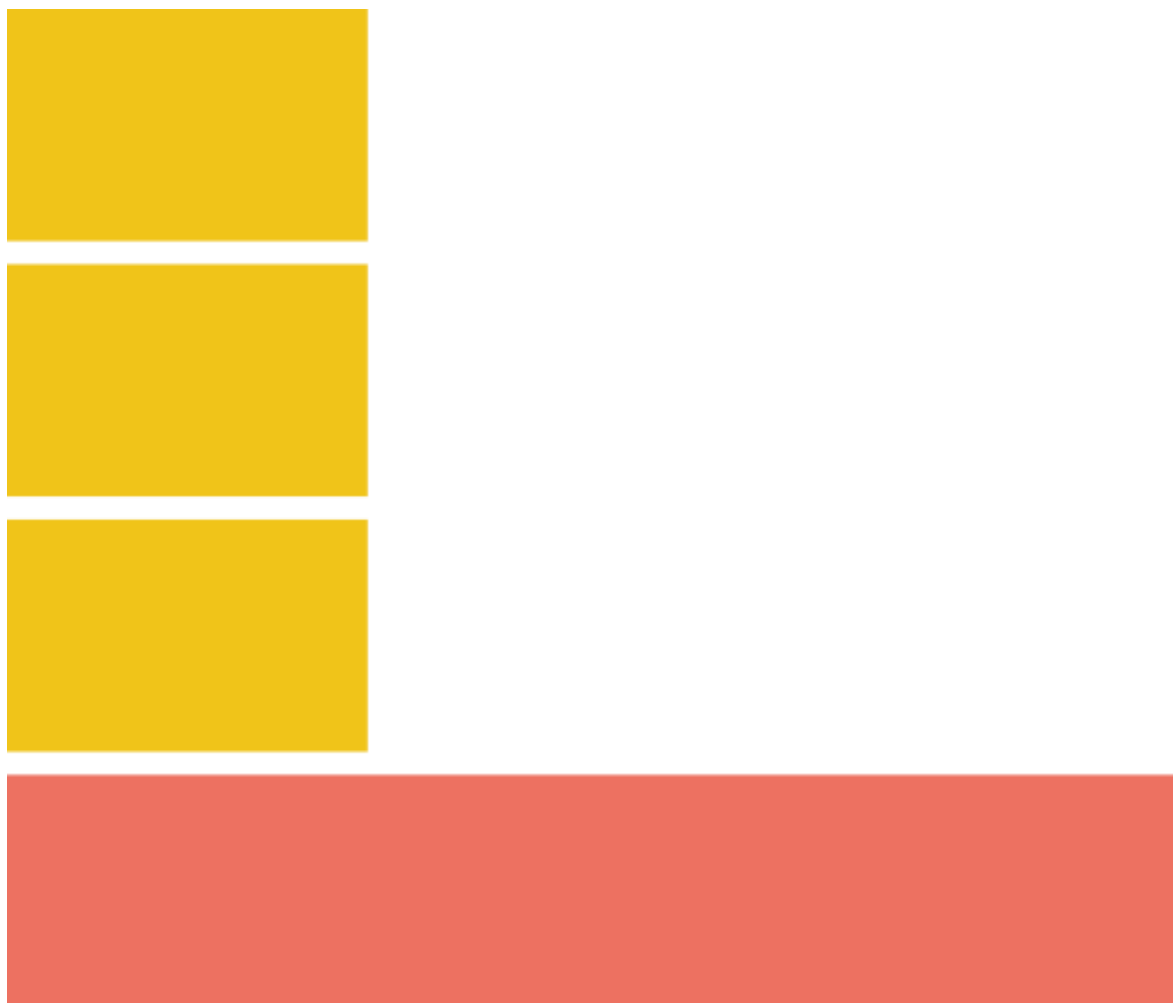
```
#container1 div{  
width: 200px;  
height: 100px;  
margin: 20px;  
background-color: #FoC419;  
}
```

```
#container2 {  
height: 100px;  
margin: 20px;  
background-color: #ED7161;  
}
```

Cu regulile CSS prezentate, este definită înălțimea și lățimea elementelor găsite în elementul `div` de tip `container1`, precum și marginile și culoarea de fundal.

Pentru elementul `container2` sunt definite înălțimea, marginile și culoarea.

În modul prezentat, elementele pe pagină vor fi aranjate ca în imaginea 15.11.



*Imaginea 15.11. Patru elemente div aranjate în flux normal*

Toate elementele participă la fluxul normal al documentului, așadar, nu există nicio dilemă cu privire la poziționarea sa pe pagină. Dar ce s-ar fi întâmplat dacă toate elementele elementului `div container1` ar fi avut valoarea `float` setată la `left`, ca în următorul exemplu?

```
#container1 div{  
width: 200px;  
height: 100px;  
margin: 20px;  
background-color: #F0C419;  
float: left;  
}
```

În acest fel, elementul `div container1` ar fi conținut exclusiv elemente cu proprietatea `float` setată la `left`. De aceea, elementul `container1` nu va avea înălțimea sa, pentru că nu conține niciun element aranjat conform fluxului normal al documentului. O astfel de situație va produce un efect identic celui prezentat în imaginea 15.12.



*Imaginea 15.12. Când toate elementele din cadrul unui container sunt scoase din fluxul normal, elementul lor părinte nu are înălțime, așadar acesta este ignorat pe pagină de toate celelalte elemente*

Desigur, scopul este să plasăm elementele unul sub altul, iar pentru a obține aceasta, este suficient să folosim proprietatea CSS `clear`.

```
#container2 {  
...  
clear: left;  
}
```

Cu o simplă adăugare a proprietății `clear` cu valoarea `left` pe elementul `container2`, se rezolvă problema de suprapunere a elementelor, provocată de faptul că elementul părinte, cu toți

descendenții mutați, nu are propria înălțime. Efectul va fi ca în imaginea 15.13.



*Imaginea 15.13. Utilizarea proprietății clear pentru rezolvarea problemei cu elementele fără înălțime*

## Proprietatea z-index

Până acum, în cadrul acestei lecții am prezentat câteva abordări cu ajutorul cărora se poate obține suprapunerea mai multor elemente. Când spunem suprapunere, ne referim la afișarea unui element deasupra unui alt element, ca în imaginea 15.14.



*Imaginea 15.14. Două elemente div care se suprapun*

Suprapunerea se poate obține prin folosirea poziționării relative,

absolute și fixe, dar și prin folosirea proprietății `float`. Când se suprapun două elemente, ordinea lor vizuală pe pagină este determinată de poziția lor în cod. Astfel elementul care este primul specificat în codul de pe pagină, apare sub elementul specificat mai târziu. Exemplul ilustrat în imaginea 15.14. se poate obține în felul următor:

```
div{  
  width: 180px;  
  height: 100px;  
  margin: 20px;  
}
```

```
#box1{  
  background-color: #FoC419;  
  position: relative;  
  top: 70px;  
  left: 50px;  
}
```

```
#box2{  
  background-color: #ED7161;  
  position: relative;  
}
```

```
<div id="box1">  
</div>
```

```
<div id="box2">  
</div>
```

Structura HTML este formată din două elemente `div`. Ambele elemente `div` au o înălțime și lățime identică și sunt poziționate relativ. Elementul `div` cu numele `box1` este mutat cu 70 px în jos și cu 50 px în dreapta. Astfel se obține situația ilustrată deja în imaginea 15.14.

Având în vedere că elementul `div` cu numele `box1` este poziționat în cod înainte de elementul `box2`, el se va afișa pe pagină vizual, sub elementul `box2`. Acest comportament implicit se poate influența prin folosirea proprietății **z-index**:

```
#box1 {  
...  
  z-index: 1;  
}  
  
#box2 {  
...  
  z-index: 0;  
}
```

În acest fel se obține afișarea ca în imaginea 15.15.



*Imaginea 15.15. Cu proprietatea z-index influențăm ordinea elementelor pe pagină*



Proprietatea `z-index` poate accepta orice număr întreg, pozitiv sau negativ. În timpul stabilirii ordinii în care vor fi poziționate elementele, luăm în considerare valoarea proprietății `z-index`. Elementul care are cea mai mare valoare a acestei proprietăți, se afișează deasupra elementelor cu o valoare mai mică.

### **Notă**

*Este foarte important să menționăm că proprietatea `z-index` are efect doar pe elementele poziționate. Mai devreme am spus că sub noțiunea de elemente poziționate se consideră elementele poziționate absolut, relativ sau fix.*

## **Proprietatea `visibility`**

Într-una din lecțiile anterioare, în care am vorbit despre proprietatea `display`, am demonstrat și efectul valorii `none` a acestei proprietăți. Ați putut vedea că elementul cu valoarea `none` a proprietății `display` dispăre de pe pagină și că alte elemente îi ocupă locul. Folosind CSS, se poate obține ceva similar. Prin urmare, elementul se poate ascunde de pe pagină, dar spațiul său rămâne rezervat, așadar alte elemente nu îl pot ocupa. Acest lucru se obține prin folosirea proprietății `visibility`:

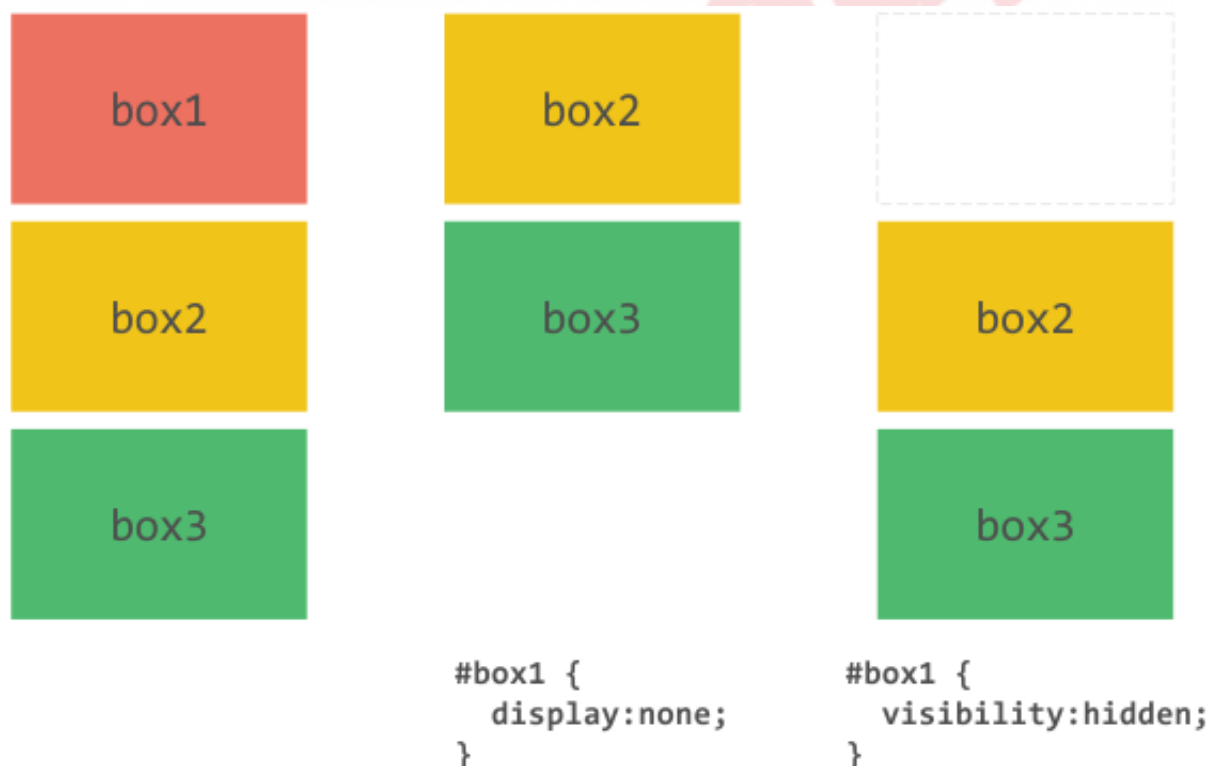
Proprietatea `visibility` poate avea două valori:

- `visible;`
- `hidden.`

Valoarea implicită este `visible`, ceea ce înseamnă că elementul va fi vizibil pe pagină, în cadrul browserului. Prin setarea valorii `hidden`, elementul se ascunde:

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

Imaginea 15.16. ilustrează diferența dintre valorile `display: none` și `visibility: hidden`.



*Imaginea 15.16. Diferența dintre valorile `display: none` și `visibility: hidden`*

## Proprietatea `overflow`

Uneori se poate întâmpla ca un conținut să depășească dimensiunea elementului său părinte. În astfel de situații, CSS ne permite să definim

ce se va întâmpla cu conținutul care va ieși din cadrul părintelui său folosind proprietatea `overflow`. Cele mai importante trei valori ale proprietății `overflow` sunt prezentate în tabelul 15.2.

Valoare	Descriere
<code>visible</code>	Conținutul care va ieși din cadrul părintelui va fi vizibil; aceasta este o valoare implicită
<code>hidden</code>	Conținutul care va ieși din cadrul părintelui nu va fi vizibil
<code>scroll</code>	Conținutul care va ieși din cadrul părintelui nu va fi vizibil, însă se vor afișa bare de derulare pentru a putea să se vadă conținutul care va ieși din cadrul părintelui

*Tabelul 15.2. Valorile proprietății `overflow`*

Următorul exemplu ilustrează utilizarea proprietății `overflow`. Codul HTML este următorul:

```
<style>
```

```
#container1{  
width: 200px;  
height: 300px;  
margin: 20px;  
background-color: #FoC419;  
display: inline-block;  
}
```

```
</style>
```

```
<div id="container1">
```

```
<p>
```

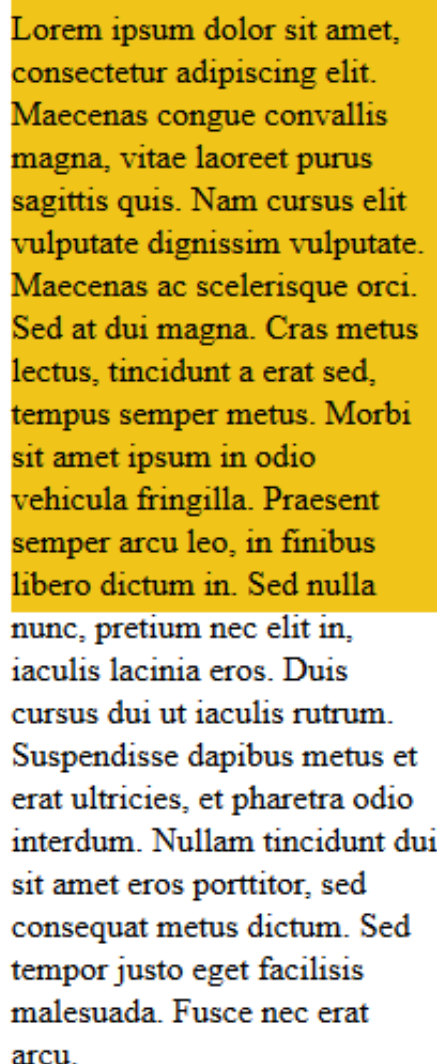
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas congue convallis magna, vitae laoreet purus sagittis quis. Nam cursus elit vulputate dignissim vulputate. Maecenas ac scelerisque orci. Sed at dui magna. Cras metus lectus, tincidunt a erat sed, tempus semper m

etus. Morbi sit amet ipsum in odio vehicula fringilla. Praesent semper arcu leo, in finibus libero dictum in. Sed nulla nunc, pretium nec elit in, iaculis lacinia eros. Duis cursus dui ut iaculis rutrum. Suspendisse dapibus metus et erat ultricies, et pharetra odio interdum. Nullam tincidunt dui sit amet eros porttitor, sed consequat metus dictum. Sed tempor justo eget facilisis malesuada. Fusce nec erat arcu.

</p>

</div>

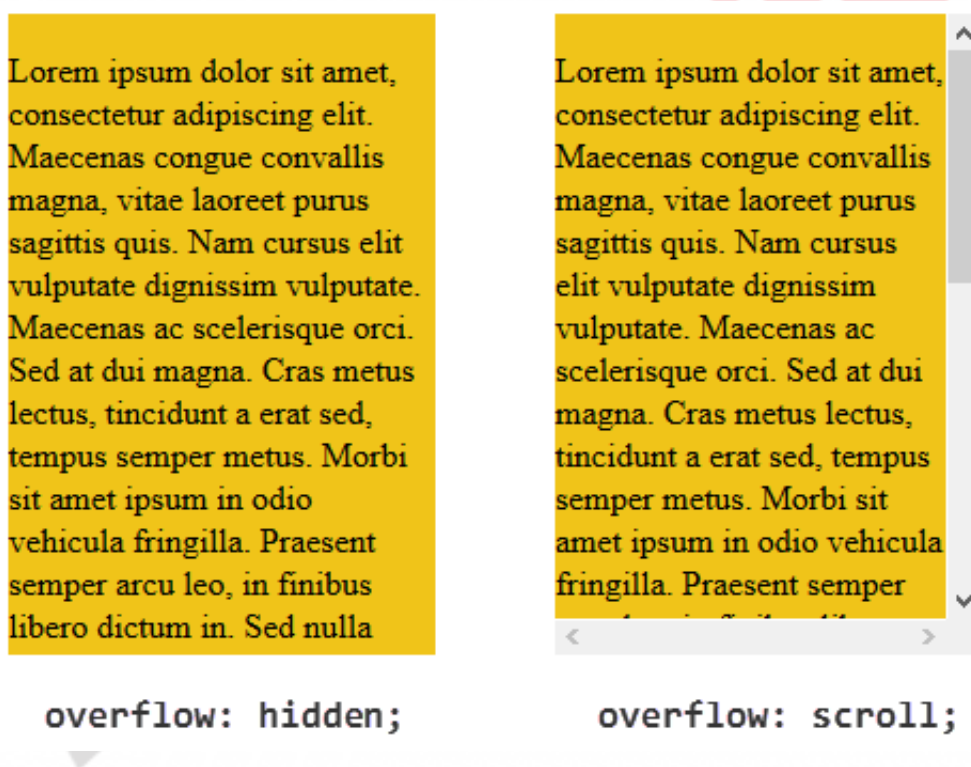
Codul specificat va produce un efect identic celui din imaginea 15.17.



Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Maecenas congue convallis  
magna, vitae laoreet purus  
sagittis quis. Nam cursus elit  
vulputate dignissim vulputate.  
Maecenas ac scelerisque orci.  
Sed at dui magna. Cras metus  
lectus, tincidunt a erat sed,  
tempus semper metus. Morbi  
sit amet ipsum in odio  
vehicula fringilla. Praesent  
semper arcu leo, in finibus  
libero dictum in. Sed nulla  
nunc, pretium nec elit in,  
iaculis lacinia eros. Duis  
cursus dui ut iaculis rutrum.  
Suspendisse dapibus metus et  
erat ultricies, et pharetra odio  
interdum. Nullam tincidunt dui  
sit amet eros porttitor, sed  
consequat metus dictum. Sed  
tempor justo eget facilisis  
malesuada. Fusce nec erat  
arcu.

*Imaginea 15.17. Conținutul elementului div iese din cadrele sale*

În imaginea 15.17. se vede clar că conținutul elementului `div` iese din cadrele sale. Având în vedere că valoarea implicită a proprietății `overflow` este `visible`, conținutul care va ieși în afara cadrului părinte, va fi vizibil. Totuși, dacă nu se specifică unele valori ale acestei proprietăți, se vor obține alte efecte, ilustrate în imaginea 15.18.



*Imaginea 15.18. Diferite efecte care se obțin prin folosirea proprietății `overflow`*

Dacă pentru valoarea proprietății `overflow` se setează `hidden`, conținutul care iese din cadrele părintelui său nu va fi vizibil. Un astfel de efect este prezentat de partea stângă în imaginea 15.18. Dacă utilizatorului trebuie să-i asigurăm vizualizarea conținutului care iese din cadrele elementului părinte, ca valoare a atributului `overflow` se poate pune `scroll`. Într-o astfel de situație, pe elementul al cărui conținut este tăiat, va apărea una sau două bare de derulare. În

imaginea 15.18., această situație este ilustrată de partea dreaptă, unde se poate vedea elementul `div` cu bare de derulare verticale și orizontale, dintre care este activă bara verticală.

## Exemplu: Crearea layoutului prin folosirea abordărilor din această lecție

În lecția anterioară am ilustrat primele exemple de creare a layoutului. După diferite noțiuni CSS din această lecție, suntem capabili să îmbunătățim layoutul din lecția anterioară. De aceea, în continuare prezentăm crearea unui layout identic celui din lecția anterioară, dar fără folosirea elementelor `inline-block`. De aceea, structura corpului documentului HTML va fi identică celei din lecția anterioară:

```
<header>
  
</header>

<nav>
  <a href="#">Home</a>
  <a href="#">About Us</a>
  <a href="#">Contact</a>
</nav>

<h1>Home</h1>

<section>
  <h2>Main content</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum vehicula quam e
ros, eget condimentum augue dictum sed. Vivamus blandit purus nibh, sit amet molestie feli
s sollicitudin id. Morbi tortor enim, varius eget gravida eu, dictum in velit. Suspendisse sit am
et metus aliquet, viverra est id, efficitur diam. Integer laoreet nisi arcu, a venenatis nisl conva
llis a. Cras malesuada lobortis ex. Praesent vel massa a eros dignissim commodo. Ut justo pur
us, tincidunt ut ultrices ac, ultrices eget tellus. Cras sodales libero sed velit aliquet condiment
um.</p>

</section><aside>
  <h2>Side content</h2>
```

<p>Pellentesque sem dolor, tempus at felis quis, dignissim lacinia enim. Curabitur non efficitur eros. Praesent pretium neque diam, non sollicitudin purus vulputate sed. Vestibulum posuere tincidunt commodo. Aliquam nec neque feugiat, gravida mi quis, pellentesque lacus. Cras mollis odio ac dignissim tincidunt. Phasellus sed blandit eros. Donec malesuada lectus vel massa ultricies, sed rhoncus nulla vestibulum. Mauris vulputate ac ipsum at elementum. Proin aliquam sit amet justo dapibus cursus. Donec id dignissim arcu. Sed sed justo in lectus efficitur maximus.</p>

</aside>

<footer>

Copyright © Link Group

</footer>

De această dată, codul de stilizare va arăta astfel:

```
body {  
  width: 960px;  
  margin: 20px auto;  
}  
  
header {  
  border-bottom: 1px solid gray;  
  padding: 8px 0 8px 0;  
}  
  
nav {  
  padding: 16px 0 16px 0;  
  border-bottom: 1px solid gray;  
}  
  
a {  
  margin-right: 6px;  
}
```



```
section {  
  float: left;  
  width: 600px;  
  padding: 0 8px 0 0;  
  border-right: 1px solid gray;  
  margin-bottom: 8px;  
  box-sizing: border-box;  
}
```

```
aside {  
  float: left;  
  width: 360px;  
  padding: 0 0 0 8px;  
  box-sizing: border-box;  
}
```

```
footer {  
  padding: 16px 0 16px 0;  
  border-top: 1px solid gray;  
  clear: left;  
}
```

Majoritatea CSS-ului prezentat este identic celui din lecția anterioară. Totuși, există câteva diferențe importante. Prima este că elementele `section` și `aside` nu mai sunt `inline-block`. Acum, acestea sunt elemente `block` clasice (deoarece elementele semantice sunt, implicit, elemente de tip `block`). Totuși, pentru obținerea afișării conținutului în două coloane, aceste două elemente sunt mutate acum din fluxul normal al documentului, folosind proprietatea `float` care este setată la `left` pentru ambele elemente. Pentru a nu apărea probleme cu afișarea elementului `footer`, care în această situație urmează după elementele mutate prin folosirea proprietății `float`, pe el este setată proprietatea `clear` cu valoarea `left`.



În cele din urmă, CSS-ul prezentat are încă o îmbunătățire. Puteți vedea că lățimile elementelor `section` și `aside` sunt setate la 600 și la 360 de pixeli. Deși ambele elemente dețin și spații interne (câte 8 pixeli de marginea dreaptă, respectiv stângă), iar elementul `section` și cadrul au câte un pixel de partea dreaptă, afișarea va fi identică celei din lecția anterioară. Atunci, cum se poate ca acum să avem posibilitatea să împărțim în mod direct spațiul disponibil, fără să luăm în considerare eventualele spații și cadre interne? Răspunsul la această întrebare se află în utilizarea proprietății **box-sizing**.

### Proprietatea box-sizing

Proprietatea CSS `box-sizing` asigură alegerea modului în care vor fi calculate dimensiunile elementului HTML. Astfel, prin folosirea acestei proprietăți putem defini dacă spațiile interne și cadrele vor intra sau nu în înălțimea și lățimea totală a elementului.

Proprietatea `box-sizing` poate avea următoarele valori:

- `content-box;`
- `border-box.`

Valoarea `content-box` este valoarea implicită și ea definește dacă înălțimea și lățimea vor cuprinde doar conținutul elementului. Pe de altă parte, valoarea `border-box` în calcularea înălțimii și lățimii include și spațiile interne și cadrele. După cum ați putut vedea din exemplul tocmai prezentat, valoarea `border-box` este mai mult decât utilă în unele situații, deoarece ne eliberează de calcularea suplimentară inutilă în timpul aranjării spațiului disponibil între mai multe elemente.

## HTCF9\_15 - HTML & CSS Fundamentals

**1. Pentru ca elementele care urmează după elementul float să fie aranjate corect și pentru a nu se ajunge la suprapunerea lor, folosim proprietatea:**

- ☐ a) float-reset
- ☐ b) reset
- ☐ c) clear
- ☐ d) float-clear

**2. Dacă setăm proprietatea float pe un element, elementele care urmează nu se vor baza pe elementul pe care este setat float-ul.**

- ☐ a) adevărat
- ☐ b) fals

**3. Poziționarea de bază, cea implicită, se mai numește și poziționare de tip:**

- ☐ a) static
- ☐ b) fixed
- ☐ c) relative
- ☐ d) absolute

**4. Poziționarea care ne permite să setăm elementul în mod relativ față de câmpul de vizualizare a browserului, este poziționarea de tip:**

- ☐ a) static
- ☐ b) relative
- ☐ c) absolute
- ☐ d) fixed

**5. Proprietatea care se poate folosi pentru influențarea ierarhiei elementelor, respectiv proprietatea care influențează care element se va afla deasupra și care dedesubt în timpul suprapunerii, este:**

- ☐ a) z-index
- ☐ b) display
- ☐ c) visibility
- ☐ d) overflow

**6. Introduceți termenul corespunzător.**

Răspunsuri oferite: relative, absolute, block, inline

Poziționările de tip static și

\_\_\_\_\_ mențin elementele în fluxul normal, în timp ce poziționările de tip

\_\_\_\_\_ și fixed elimină elementul din fluxul normal al paginii.

## **7. Introduceți termenul corespunzător.**

Răspunsuri oferite: visibility, display

Proprietatea

\_\_\_\_\_ asigură ascunderea unui anumit element HTML, iar pe lângă aceasta, spațiul pe care îl ocupă va rămâne rezervat.

**1. Pentru ca elementele care urmează după elementul float să fie aranjate corect și pentru a nu se ajunge la suprapunerea lor, folosim proprietatea:**

C

**2. Dacă setăm proprietatea float pe un element, elementele care urmează nu se vor baza pe elementul pe care este setat float-ul.**

a

**3. Poziționarea de bază, cea implicită, se mai numește și poziționare de tip:**

a

**4. Poziționarea care ne permite să setăm elementul în mod relativ față de câmpul de vizualizare a browserului, este poziționarea de tip:**

d

**5. Proprietatea care se poate folosi pentru influențarea ierarhiei elementelor, respectiv proprietatea care influențează care element se va afla deasupra și care dedesubt în timpul suprapunerii, este:**

a

**6. Introduceți termenul corespunzător.**

relative, absolute

**7. Introduceți termenul corespunzător.**

visibility