Până acum, ne-am familiarizat cu bazele limbajului CSS şi cu proprietățile elementelor. În cadrul acestei lecţii, ne vom extinde cunoştinţele şi vom discuta despre fluxul documentului, respectiv modul de aranjare a elementelor unul peste altul şi de creare a structurii. Vom explica diferenţele dintre un flux normal (engl. *natural stack*) şi alte tipuri. Vom vedea ce este valoarea *float* şi când se aplică.

Poziţionarea

Când poziţionăm elementele, distingem următoarele tipuri:

- static (default),
- relative,
- absolute,
- fixed.

Poziţionarea statică şi relativă creează un flux normal al documentului.

Poziţionarea statică

În mod implicit, *Static* este poziţionarea de bază a fiecărui element de pe pagină. Deseori, această poziţionare nici nu este menţionată în literatura de specialitate sau este indicată ca implicită, normală etc. Important este faptul că există şi că poate fi aplicată pe elemente.

Poziţionarea statică este aplicată pe toate elementele care nu au o poziţie specificată în prealabil prin *Absolute* sau *Fixed* şi care nu sunt *Floated* (despre *float* vom discuta mai târziu).

În lecția despre elemente, am aflat care sunt regulile de bază ale fluxului normal, iar acum le vom repeta. Elementele *Block* sunt poziţionate unul sub altul pe verticală, în timp ce elementele *Inline* sunt poziţionate pe orizontală, de la stânga la dreapta. Marginile verticale se suprapun în fluxul normal al documentului. Aceasta înseamnă că, în loc să adunăm valorile marginilor elementului de sus şi de jos, calculăm doar marginea mai mare. Marginile orizontale nu se

© Copyright Link group

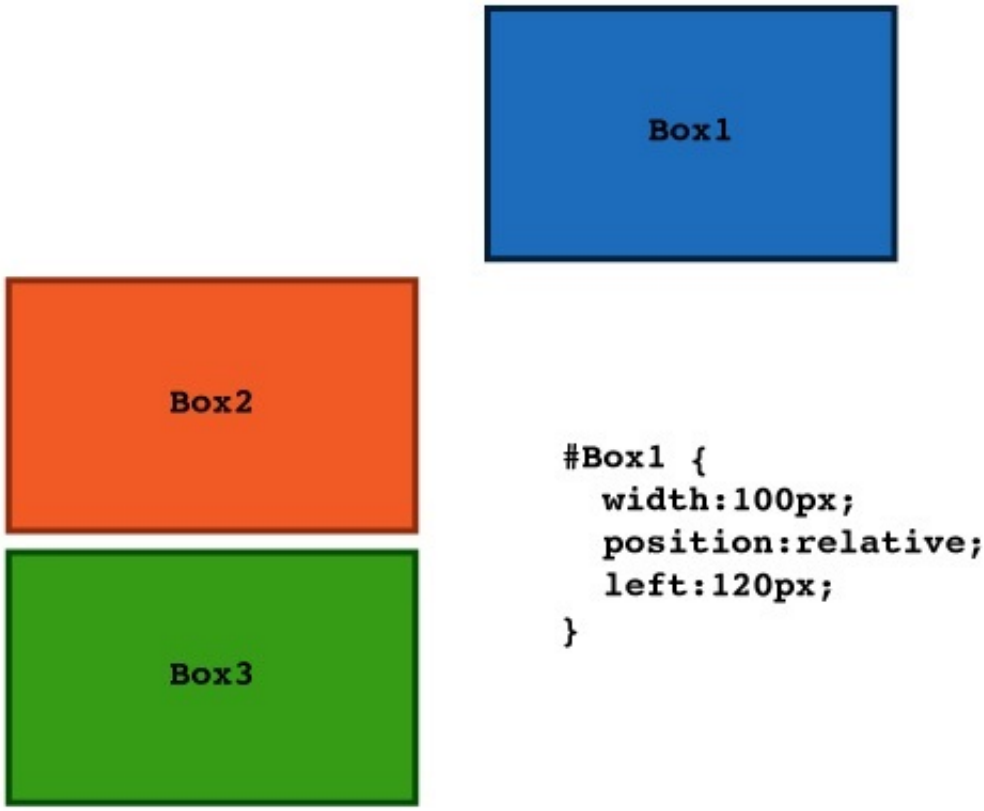
suprapun niciodată.

Poziţionarea relativă

Când unui element i se acordă proprietatea *position:relative*, atunci elementul este poziţionat inițial după regulile fluxului normal, respectiv complet identic, dar cu o singură diferenţă în raport cu poziționarea statică; de aceea, poziţionarea relativă este considerată un mod separat de poziţionare.

Elementul poate fi mutat din poziţia sa iniţială, iar spaţiul pe care îl ocupă rămâne gol. Îl mutăm conform proprietăţilor **Offset***,* care pot fi *Left* (stânga)*, Right* (dreapta)*, Top* (sus) *şi Bottom* (jos). Valoarea fiecăreia dintre aceste proprietăţi reprezintă distanţa la care ar trebui mutată marginea corespunzătoare a elementului în raport cu poziţia elementului în varianta statică.

© Copyright Link group



Imaginea 12.1. Poziţionarea relativă

Box 1 foloseşte poziţionarea relativă pentru a se muta spre dreapta cu 120px, în raport cu poziţia lui de start. Box-urile 2 şi 3 rămân în acelaşi loc, pe poziţii neschimbate (se comportă ca și când Box 1 nu şi-ar fi schimbat poziţia, respectiv ca și când ar fi rămas în acelaşi loc), deoarece poziţionarea relativă nu mută elementele din fluxul normal.

Atenţie: Deplasarea spre dreapta se marchează cu proprietatea *left*, deoarece elementul cu valoarea sa este îndepărtat de marginea stângă a poziţiei de bază. În mod analog, proprietatea *top* mută elementul în jos.

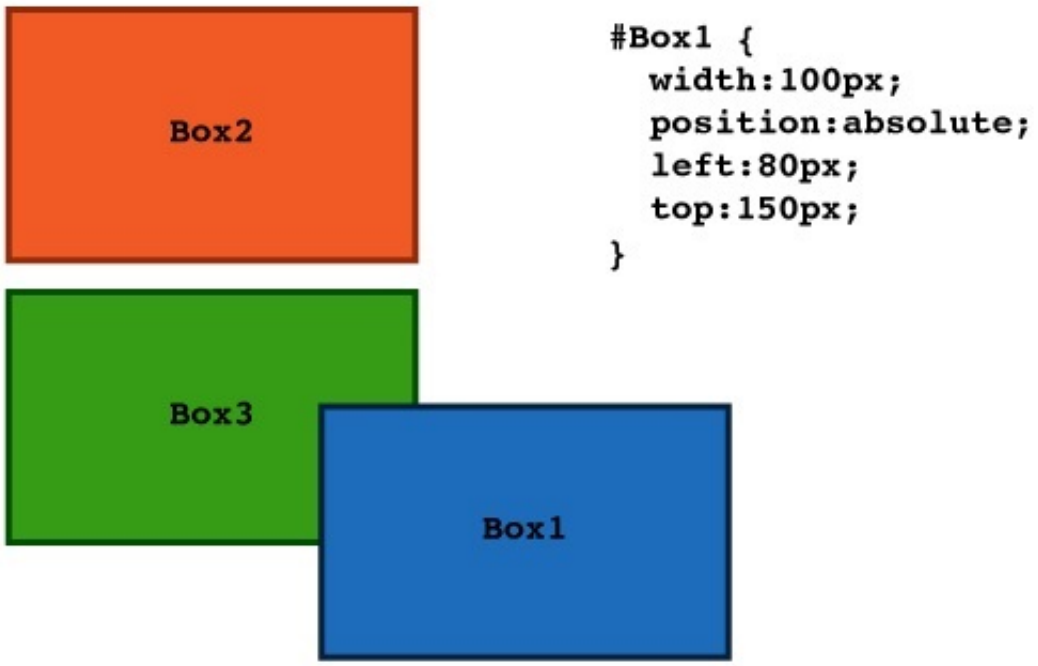
Poziţionarea absolută

Elementele cu proprietatea *position:absolute*; sunt eliminate, respectiv

© Copyright Link group

mutate din fluxul normal şi nu influenţează celelalte elemente. Poziţia elementelor este determinată pe baza valorilor *Offset,* care pot fi *Left*, *Right*, *Top* şi *Bottom*. Aceste valori se comportă aproape la fel ca şi în cazul poziţionării relative, doar că la poziţionarea absolută elementul este mutat în raport cu elementul părinte, care trebuie și el poziţionat, respectiv trebuie să i se atribuie proprietatea *position.* Cel mai simplu este să adăugăm *position:relative* elementului părinte, dacă nu vrem să-l mutăm.

Poziţionarea absolută ne permite să definim unde ar trebui să poziţionăm, în mod absolut, elementul în raport cu pagina sau în raport cu elementul părinte.



Imaginea 12.2. Poziţionarea absolută

Box 1 este poziţionat absolut, proprietăţile *Left* şi *Top* îl mută la dreapta şi în jos, în timp ce Box 2 şi 3 se comportă ca și când Box 1 nici

© Copyright Link group

nu există şi îi ocupă poziţia în fluxul normal. Din cauza noii sale poziţii, Box 1 se suprapune peste Box 3 şi orice modificare a conţinutului lui Box 3 nu influenţează poziţia celorlalte două elemente.

O capcană frecventă pentru începători constă tocmai în folosirea poziţionării absolute pentru toate elementele. Deşi poate părea atrăgător, acest lucru poate crea mari probleme (şi le va crea cu siguranţă) mai târziu, în timpul activității.

Acest tip de poziţionare trebuie folosit doar uneori, respectiv atunci când este necesar, însă cu mare atenție.

Poziţionarea fixă

Poziţionarea fixă este un tip special de poziţionare absolută. Când derulăm pagina, elementul fix nu se mută la fel ca restul elementelor. Acest mod de poziţionare nu este suportat de Internet Explorer 6, precum şi de versiunile mai vechi ale acestuia. Totuşi, acest lucru nu mai este important, fiindcă Internet Explorer 6 aproape că nici nu se mai foloseşte.

```
#myElement {
  position:fixed;
  left:10%;
}
```

Elementul este mutat la dreapta cu 10% din lăţimea totală a ferestrei browserului şi este fixat în acel loc, astfel încât, atunci când derulăm pagina, *myElement* rămâne în acelaşi loc, indiferent de restul elementelor.

De asemenea, în următorul material video puteţi vedea

© Copyright Link group

comportamentul elementului poziţionat fix:

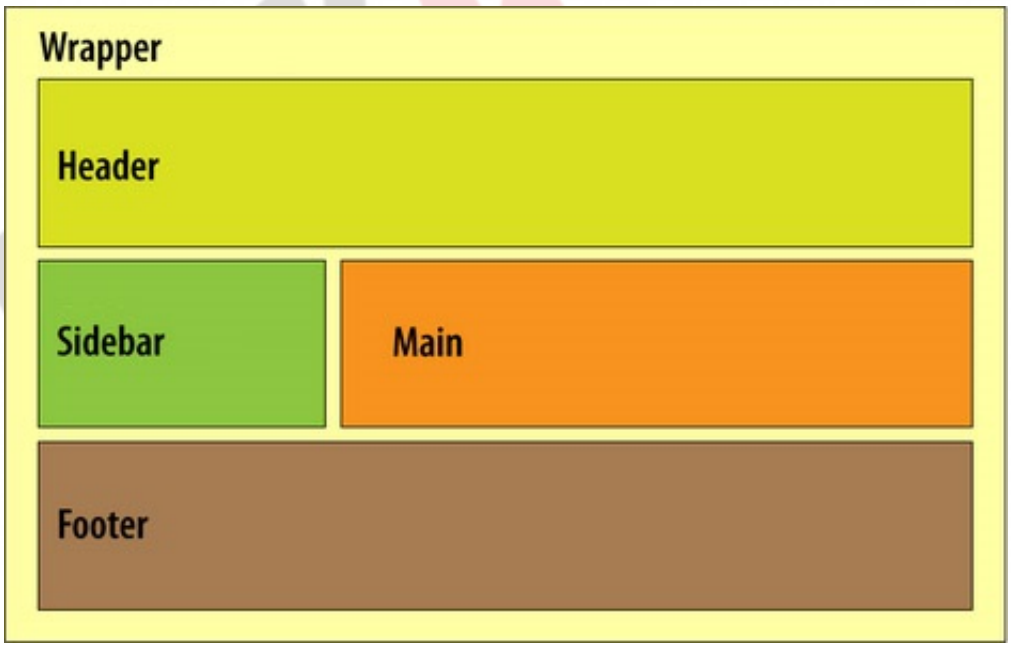
Materialul video 12.1 Comportamentul elementului poziţionat fix

Utilizarea elementelor *Block* pentru layout

Pentru a crea structura de bază a paginii, folosim elementele *Block* (de obicei, DIV-urile). Am menţionat că în trecut, în acest scop, se utilizau tabelele invizibile, însă astăzi nu mai sunt recomandate.

Prin urmare, creând elementele DIV şi aranjându-le în unităţi, setăm aspectul de bază (layout-ul) al paginii noastre, respectiv al site-ului.

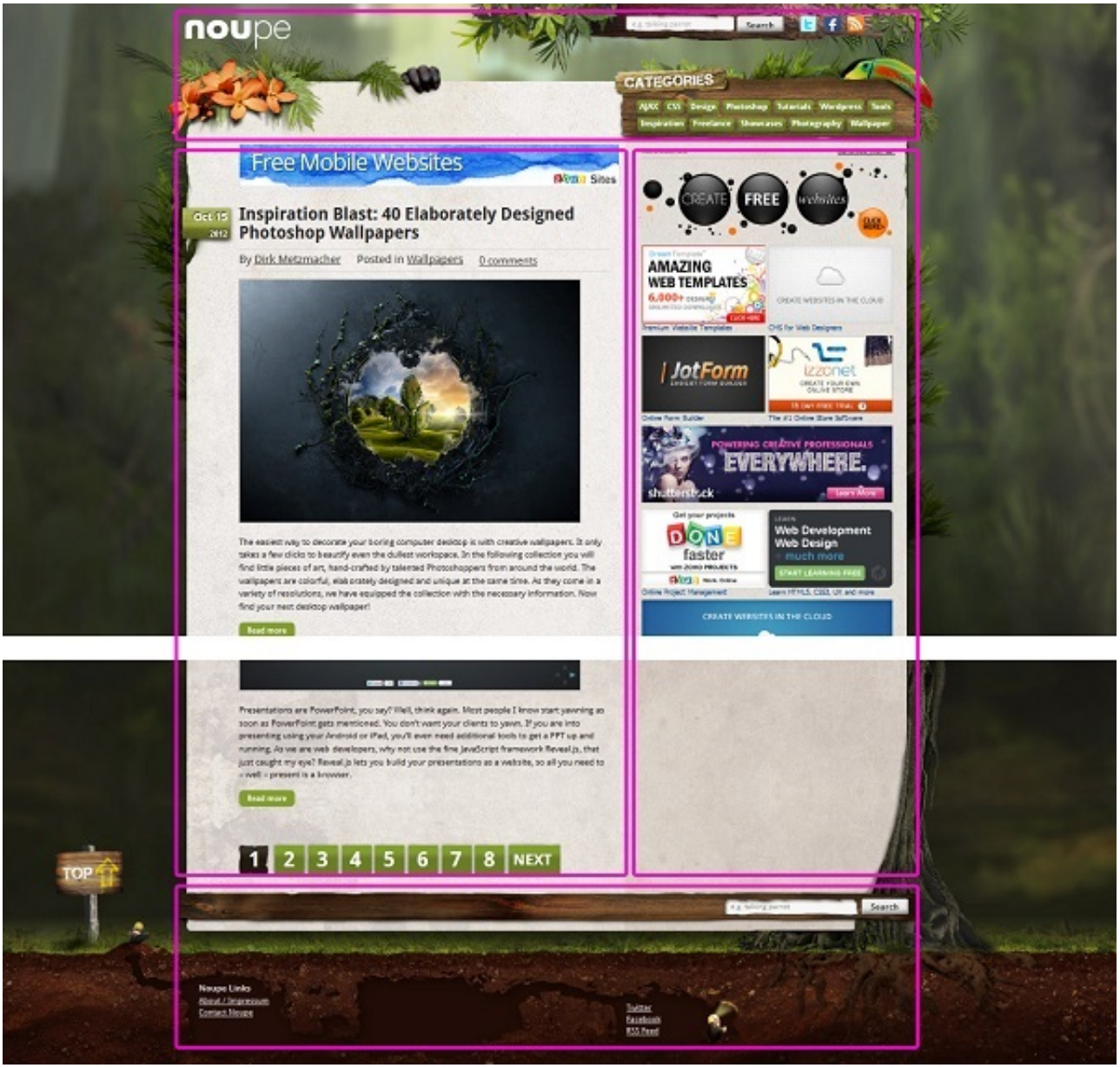
Pentru a ilustra asta, vom analiza exemplul de mai jos.



Imaginea 12.3 - Un layout întâlnit frecvent pe site-uri

În imagine, nu este prezentată pagina site-ului, ci un layout frecvent întâlnit în zilele noastre pe site-uri. Haideţi să vedem un blog de design web popular în zilele noastre – www.noupe.com – şi principalul său layout. Liniile mov au fost adăugate pentru a ne ajuta să identificăm

mai uşor regiunile. În mod normal, aceste linii nu există pe site.



Imaginea 12.4. www.noupe.com

În acest exemplu, vedem că este foarte similar cu conceptul nostru din imaginea precedentă, deşi, din cauza designului, acest lucru nu se observă la prima vedere. Deşi utilizatorul nici nu este conştient de acest fapt, regiunea principală a site-ului a fost creată astfel încât să

© Copyright Link group

putem seta elementele DIV (sau echivalentele lor din limbajul HTML5).

Setarea layout-ului

Există patru reguli sau aspecte pe care trebuie să le respectăm atunci când setăm elementele Block pentru layout, şi anume.

1. **Dimensiunile** - spaţiul ocupat de element şi care este influenţat.
2. **Float** - efectul asupra comportamentului *Block*-ului. Separarea de fluxul normal.
3. **Clear**: restabilirea fluxului normal.
4. **Embedding**: elementul se află fie în interiorul unui alt element, fie la acelaşi nivel al ierarhiei, fără să existe nimic între.

Important: Toate elementele se referă la fluxul normal (la poziţionarea *Static* şi *Relative).* Poziţionările *Absolute* şi *Fixed* sunt diferite din punct de vedere conceptual, iar aceste noţiuni nu se referă la ele.

1. **Lăţimea totală**

Lăţimea totală depinde de mai multe proprietăți ale elementului, după cum am explicat deja în lecţia precedentă. Dimensiunile totale sunt influenţate de lăţime/înălţime, dar şi de padding şi border. În afară de aceasta, marginile elementului se află, din punct de vedere vizual, în afara elementului, însă afectează și spaţiul pe care îl ocupă şi pe care îl influenţează.

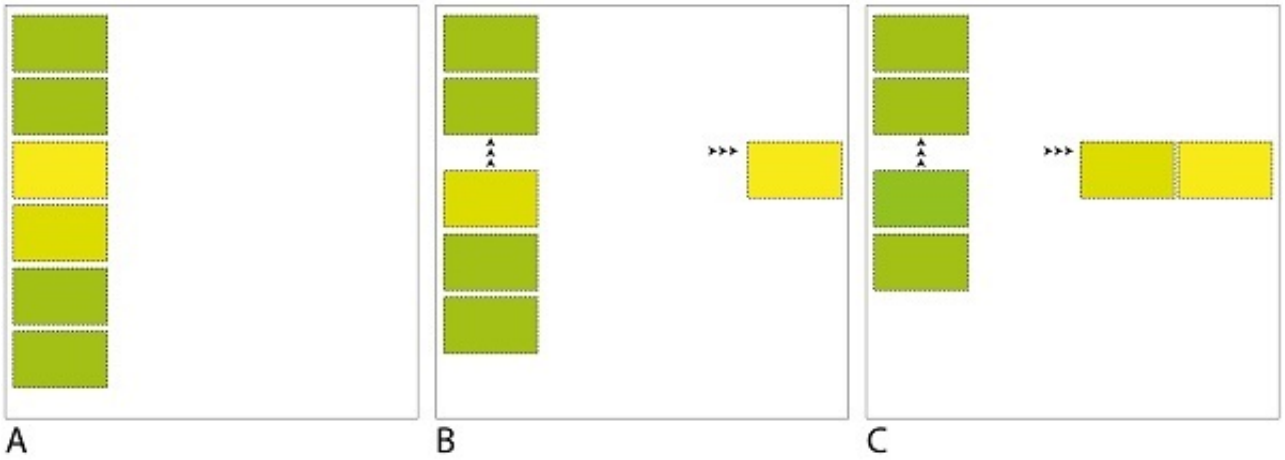
Dacă adunăm greşit dimensiunile, se vor semnala erori. De exemplu, dacă postăm un DIV cu o lăţime de 1000 px, iar în cadrul lui vrem să punem alte două DIV-uri, unul lângă altul, le putem atribui o lăţime de 500 px numai dacă valorile pentru margin, padding şi border sunt 0. Dacă modificăm (mărim) dimensiunile respective, trebuie să reducem

© Copyright Link group

lăţimea pentru ca suma să rămână la 1000 px (sau mai puţin de 1000 px).

2. Float

Floating se realizează prin proprietatea *float,* cu valorile *left* (stânga) şi right (dreapta). Când atribuim această valoare unui element, acesta iese din fluxul normal pe care l-am menţionat mai devreme şi se comportă diferit. Pe verticală, este poziţionat la fel (ca la fluxul normal - static sau relativ), în timp ce pe orizontală este poziţionat mult spre stânga sau dreapta, cât de departe este posibil în cadrul *Block*-ului care îl conţine. Spre deosebire de box-urile din fluxul normal, marginile verticale ale unui element *floated* nu se suprapun niciodată cu marginile elementelor de deasupra sau dedesubt.



Imaginea 12.5 - Afişarea comportamentului elementelor float

În imaginea de mai sus, în partea marcată cu **A** putem vedea mai multe elemente Block, care se află în fluxul normal (nu le-a fost setat float*).* În imagine, **B** este un element cu proprietatea *float* setată la *float:right*; poziţia lui se schimbă imediat spre dreapta la înălţimea la care a şi fost pus la început. Toate elementele care urmează ignoră elementul float şi îi ocupă locul. Dacă punem float şi pe următorul element (cel marcat cu **C**), el se va muta spre dreapta până la

© Copyright Link group

elementul floated, însă aceste două elemente nu se vor suprapune. Celelalte elemente se mută din nou, fără să fie influențate de cele două.

Float se aplică foarte simplu, adăugând proprietatea *float* şi valorile *left* şi *right* la regula CSS dorită.

```
#sidebar{
float:right;
width:200px;
}
```

Vă recomandăm ca valoarea *float* să fie urmată întotdeauna de lăţimea elementului.

Notă:

O eroare care apare deseori, în special la începutul lucrului cu poziţionarea, este folosirea proprietăţii float asupra elementului şi încercarea de a centra acest element în partea centrală. Acest lucru nu este posibil, deoarece elementul cu proprietatea float se poate muta doar în stânga sau dreapta, în niciun caz nu se poate găsi în partea centrală. Pentru aceste scopuri se foloseşte setarea marginilor pe elementul propriu-zis (cel mai des se foloseşte proprietatea margin:auto) sau introducerea manuală a marginilor pentru setarea elementului în partea centrală a paginii.

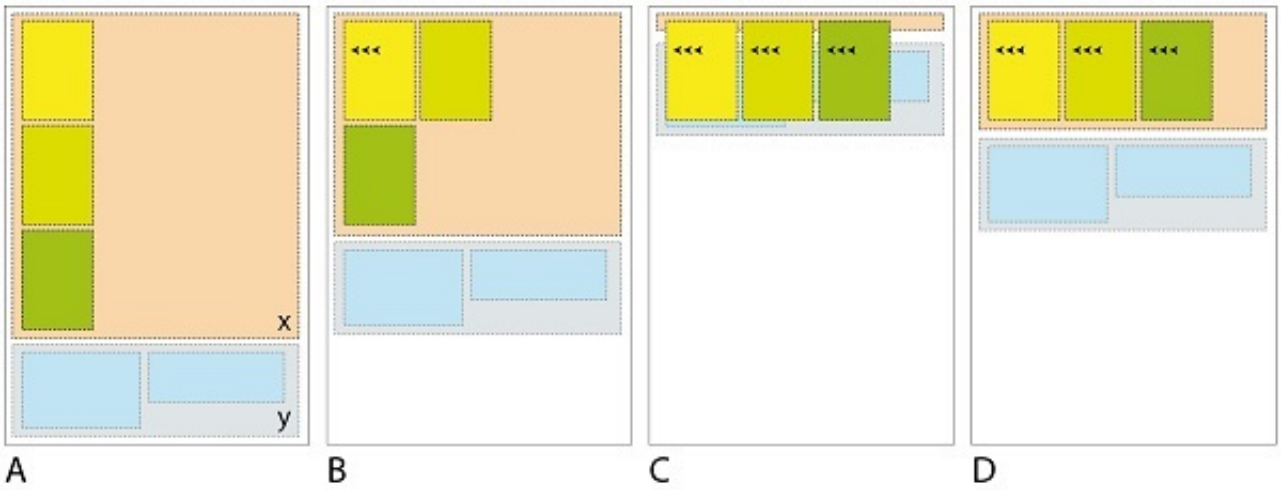
Trebuie să menţionăm că float este, tradiţional, proprietatea generală, folosită pentru layout. În cadrul acestui curs ne ocupăm cu această variantă generală a layout-ului, în timp ce în următoarele cursuri vom învăţa câteva tehnici avansate, care, încetul cu încetul, elimină proprietatea float din utilizare. Desigur, float nu este eliminat în totalitate, doar că se foloseşte mai puţin, aşadar, este foarte important să cunoaşteţi tehnicile de gestionare ale elementelor float, deoarece

© Copyright Link group

ele reprezintă baza pentru învăţarea layout-urilor mai avansate.

3. **Clear**

Una dintre caracteristicile (putem spune inconvenabilă) proprietăţii float este că elementul părinte, care conţine elementul float, nu-i recunoaşte înălţimea (Height). Prin urmare, înălţimea elementului float nu influenţează comportamentul (înălţimea) elementului părinte. Elementul părinte se străduieşte să cuprindă întotdeauna toate elementele sale, în afară de cele float. De ce este important acest lucru? Este important deoarece, dacă elementul părinte conţine doar elemente floated, nu va avea înălţime, respectiv va fi egală cu zero (dacă nu este setată manual prin proprietatea Height). Pe o pagină în care majoritatea elementelor se află în fluxul normal, acest lucru poate crea un adevărat haos. De asemenea, elementele care nu sunt floated, dar care sunt inserate după elementele floated, pur şi simplu nu vor recunoaşte aceste elemente floated. Haideţi să observăm ilustrația de mai jos:



Imaginea 12.6 - Afişarea comportamentului elementelor float

La început (**A**), sunt setate două elemente DIV, X şi Y, cărora nu li s-a specificat înălţimea, așa că depind de elementele interne. X conţine

trei elemente *Block.*

Dacă aplicăm proprietatea float pe unul dintre cele trei elemente X (**B**), celelalte două se vor plasa lângă el şi doar înălţimea lor (a elementelor care nu sunt float) influenţează DIV-ul X (vedem că i se micşorează înălţimea), iar elementul Y se mută în sus.

Dacă toate cele trei elemente din cadrul elementului X sunt *float:left;* (imaginea **C**), elementul X îşi va pierde complet înălţimea. Elementele sale sunt în continuare vizibile, însă următorul element Y s-a mutat până la X. Elementele se suprapun şi se creează un adevărat haos.

Soluţia este simplă și constă în adăugarea proprietăţii ***clear*** pe elementul care urmează imediat după elementele floated. În acest caz, este vorba de elementul Y. Proprietatea clear menţionată poate avea valorile *left, right* sau *both*. Cu ele marcăm tipul de elemente float de deasupra, pe care vrem să-l rezolvăm, respectiv ca elementul nostru să-l recunoască. În exemplul nostru, dacă toate cele trei elemente din cadrul lui X au *float:left*; atunci *clear:left* sau *clear:both;* pe elementul Y, soluţionează problema.

```
#footer{
clear:both;
}
```

Poate vă puneţi întrebarea de ce am folosit clear. Am făcut-o pentru a seta înălţimea (Height) pe elementele din jur. Aceasta ar putea fi o soluţie, însă astfel am limita multe aspecte ale propriei noastre pagini. De obicei, aceste elemente se folosesc pentru elementele de bază ale structurii, ceea ce înseamnă că diferite pagini de pe același site (cu elemente identice) pot avea conţinuturi diferite. Printre designeri web circulă o vorbă: *„Dacă puteţi să dublaţi întregul conţinut al paginii (să*

© Copyright Link group

aveți un text de două ori mai lung sau două imagini în loc de una etc.) fără să-i distrugeți aspectul, atunci aţi creat un layout bun."

Informații suplimentare despre proprietatea *float*

Deşi concepută cu intenţia de a organiza float, proprietatea clear nu reprezintă întotdeauna cea mai bună soluţie. În general, se utilizează pentru a adăuga cel mai simplu element Block direct după elementele floated (de exemplu, tag-ul
), cu o înălţime de 1 px, vizibilitatea 0 etc. (deoarece nu trebuie să fie vizibil pe pagină), însă i se şi adaugă *clear:both*. Poate aceasta este cea mai elegantă soluţie, însă necesită tag-ul suplimentar, care nu are un alt scop sau conţinut.

O altă soluţie, poate mai bună, care a apărut mai mult sau mai puţin întâmplător, este utilizarea proprietăţii *overflow,* despre care am vorbit mai devreme. De fapt, dacă elementul care înconjoară elementele float conţine proprietatea *overflow:hidden;*, va lua în considerare înălţimile lor şi se va seta corect în jurul acestor elemente, ca și când nu ar avea *float* și astfel n-ar mai fi nevoie de proprietatea clear. Utilizând această soluţie, evităm tag-urile suplimentare, iar elementele care urmează nu trebuie să aibă rânduri CSS suplimentare. În exemplul nostru de mai sus, am fi putut să adăugăm *overflow:hidden;* pe elementul X şi astfel am fi soluţionat problema.

4. Embedding

Când planificăm, trebuie să separăm şi să planificăm clar elementele, respectiv trebuie să definim ce elemente se vor afla la acelaşi nivel, unul după altul, şi care se vor afla unul într-altul. Vă reamintim că toate elementele se află în elementul root (body), însă atunci, pe toate celelalte le putem pune în cadrul acestor elemente.

Exemplu de layout

© Copyright Link group

Având în vedere că am învăţat deja ce este float şi cum putem insera elementele, haideţi acum să vedem un exemplu general. De exemplu, să zicem că vrem să inserăm două coloane în partea centrală a paginii, apoi main div (pentru conţinut) şi sidebar div (pentru conţinutul secundar). În HTML am fi adăugat, pur şi simplu, următoarele:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8" />
 <title>Float</title>
</head>
<body>

 <div id="wrapper">

  <div id="main">
  </div>

  <div id="sidebar">
  </div>

 </div>

</body>
</html>
```

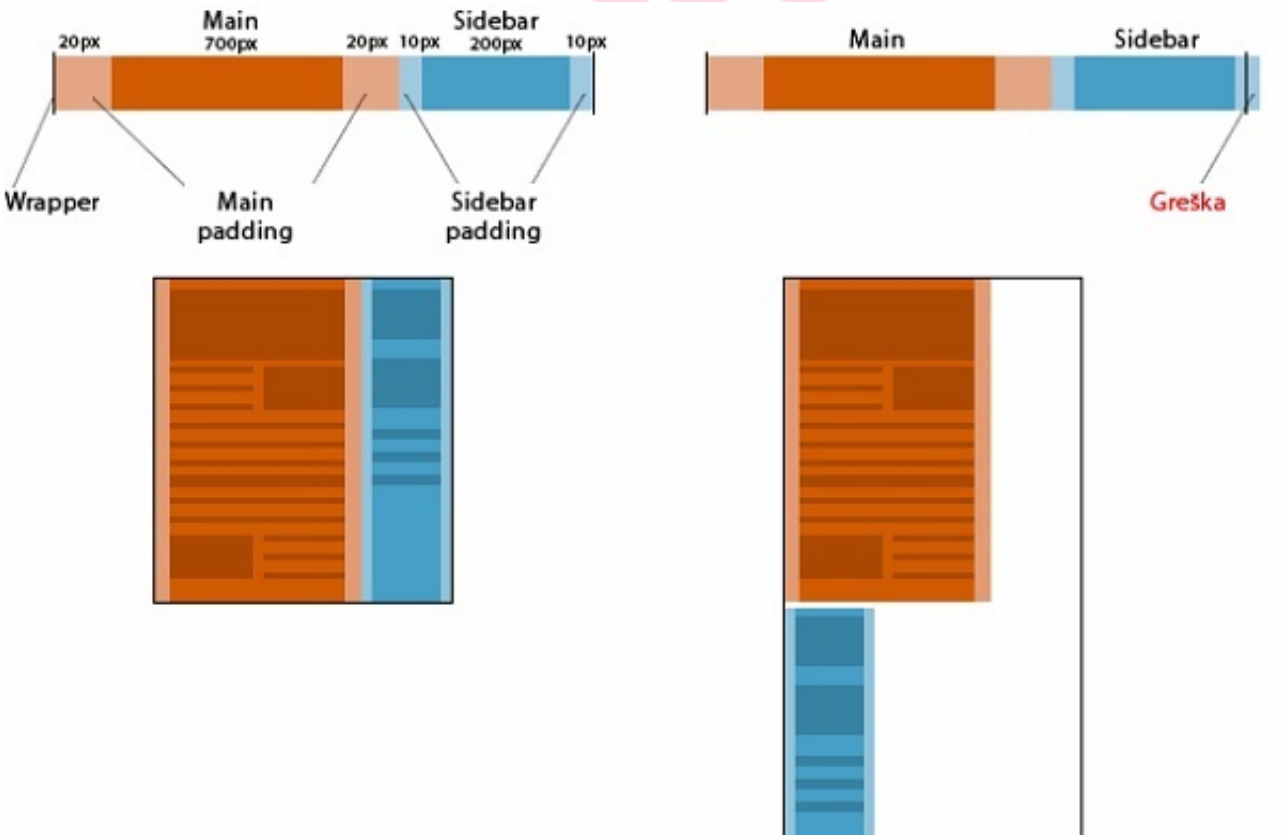
În timp ce în CSS am fi introdus:

```
#wrapper {
 width:960px;
}

#main {
 float:left;
 width:700px;
```

© Copyright Link group

```
 padding:0 20px 0 20px;
}

#sidebar {
 float:right;
 width:200px;
 padding:0 10px 0 10px;
}
```

Dacă introducem conţinutul în elementele *main* şi *sidebar,* obţinem ceva similar ilustrației de mai jos (partea stângă a imaginii):



Imaginea 12.7 - Float al principalelor elemente

© Copyright Link group

Dacă privim cu mai multă atenție partea stângă a ilustraţiei şi codul nostru, vom observa că suma totală a lăţimilor este exact de 960px, ceea ce corespunde lăţimii elementului din jur.

700px (main) + 2 x 20px (main padding) + 200px (sidebar) + 2 x 10px (sidebar padding) = 960px

Dacă am fi mărit măcar cu un pixel una dintre aceste valori, am fi obţinut ceva asemănător cu partea dreaptă a ilustraţiei. Deoarece nu există suficient spaţiu (lăţime), sidebar ar fi trecut pe rândul următor, respectiv sub secțiunea main.

Bineînţeles, acestea sunt doar două elemente. Acum, putem adăuga un div header înainte de partea main (în cadrul lui wrapper) şi footer după partea sidebar (înainte de a închide wrapper div). Singura diferenţă constă în faptul că acestor două elemente nu le trebuie float, aşadar le putem lăsa să se extindă pe întreaga lăţime (în mod implicit, pentru elementele block).

Z index

Dacă elementele sunt inserate cu ajutorul poziţionării relative, absolute sau fixe şi dacă se suprapun, elementul care apare mai târziu în codul documentului va fi vizibil deasupra elementului care a apărut mai devreme în cod, ascunzându-l astfel într-o anumită măsură.

Totuşi, putem influenţa această suprapunere utilizând valoarea *z-index*. Aceasta este similară cu opţiunile *send to back, bring forward* sau cu alte opţiuni similare pe care le putem găsi în Word, Photoshop etc.

Ca valoare, introducem orice valoare numerică la libera alegere, iar elementul cu valoarea mai mare va fi vizibil. Valorile pot fi și negative.

```
.elementuldinFata {
```

© Copyright Link group

```
 z-index:100;
}

.elementuldinSpate {
z-index:-20;
}
```

Prin urmare, nu valorile sunt importante, ci relaţia dintre ele.

Notă:

În materialul multimedia ataşat acestei lecţii, prezentăm un exemplu al layout-ului unui site, aşadar, obligatoriu să urmăriţi întregul proces, deoarece teoria prezentată în această lecţie va fi mult mai clară după ce finalizaţi exemplul din materialul multimedia.

© Copyright Link group