(/)

CSS float: considerações, dicas e macetes para bons layouts na web

Aprenda sobre CSS float e veja dicas e macetes para fazer bons layouts para a web usando tableless

CSS float: considerações, dicas e macetes para bons layouts na web

11/07/2008
CSS (http://desenvolvimentoparaweb.com/categoria/css/)
Interface (http://desenvolvimentoparaweb.com/tag/interface/), Layout (http://desenvolvimentoparaweb.com/tag/layout/), Planejamento (http://desenvolvimentoparaweb.com/tag/planejamento/), Web Design (http://desenvolvimentoparaweb.com/tag/web-design/)
Tárcio Zemel (https://plus.google.com/114896277363762090056?rel=author)

Entender o correto funcionamento e a dinâmica da propriedade CSS “float” é fundamental para o desenvolvimento e estruturação de bons *layouts* para *web*. Saber usar corretamente *float* em *web sites* e projetos *web*, em geral, é de extrema importância, já que, basicamente, é através das regras CSS que utilizam *float* que é possível compor *layouts* sem o uso de tabelas (***tableless***), o que traz inúmeros benefícios, como economia no tempo de carregamento, adequação às normas W3C (http://www.w3.org/) e, consequentemente, um aumento de performance, em geral.

Este é um artigo traduzido do original “All About Floats (http://css-tricks.com/all-about-floats/)“, do blog CSS-Tricks (http://css-tricks.com/), e a tradução foi feita com autorização do autor (http://chriscoyier.net/).

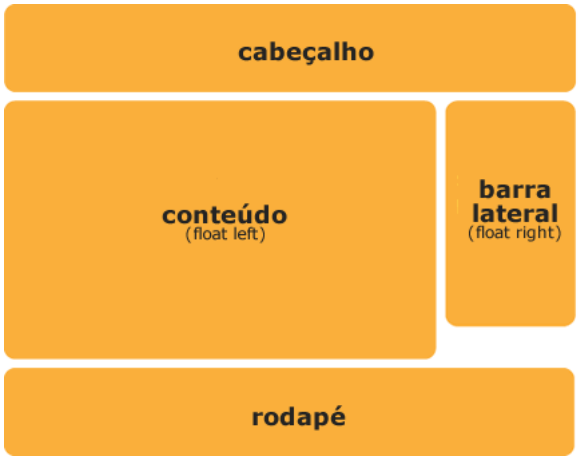
O que é “Float”?

Float é uma propriedade CSS de posicionamento. Se você está familiarizado com projetos para mídia impressa, você pode pensar, de forma semelhante, numa imagem em um *layout* onde o texto a circunda quando necessário.

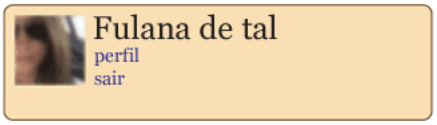
No *design* para *web*, uma imagem inserida continua a fazer parte do fluxo da página. Isto significa que, se forem feitas alterações no tamanho ou se elementos ao seu redor mudarem, a página irá automaticamente ser reajustada ("*reflow*"). Isso difere da página onde os elementos são posicionados de forma absoluta. Elementos posicionados de forma absoluta são removidos do fluxo da página *web*. Elementos posicionados absolutamente não afetarão quaisquer outros elementos da página, quer estes estejam em contato, ou não.

Para que são usados os floats?

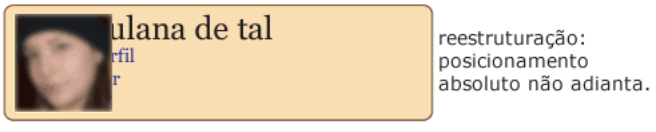
Além de simples exemplos, como posicionar uma imagem ao lado de um bloco de texto, floats são usados para criar *layouts* para *web*.



Floats são igualmente úteis para *layouts* de pequenas instâncias. Veja, por exemplo, esta parte de uma *web page*:



Esses tipos de *layouts* podem ser manipulados usando posicionamento absoluto dentro de posicionamento relativo, mas elementos que flutuam (propriedade CSS "float") são mais flexíveis. Vamos supor que o tamanho da imagem de um avatar precise ser alterado. Com *floats*, o *box* pode ser reestruturado para acomodar um tamanho maior, enquanto um posicionamento absoluto criaria problemas:

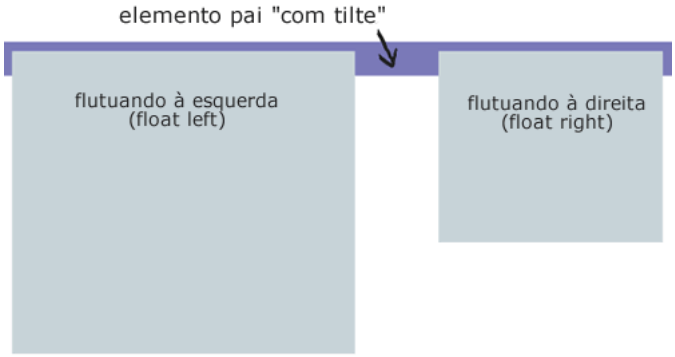


Problemas com floats

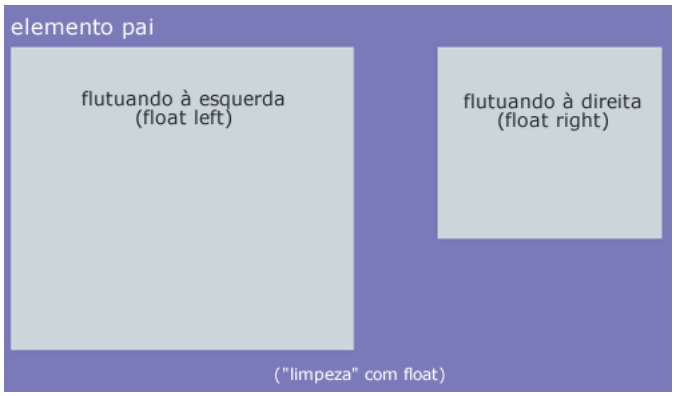
"*Floats* são frágeis". Eles são cheios de "contratempos" e *cross-browser quirks (http://emanuelfelipe.net/blog/quirks-mode-e-standards-mode-entendendo-os-modos-de-renderizao/)*. Talvez o mais significativo seja a necessidade de "limpar" *floats* (propriedade CSS "clear") em algumas situações. Primeiro, vejam alguns exemplos de porque alguns *floats* precisam ser "limpos" e, depois, como fazer esta "limpeza".

Arrumar o *float* para ajustar a altura do elemento pai

Elementos que contêm elementos *float* não calculam sua altura como é de se esperar. De fato, se o elemento pai contém apenas elementos flutuadores, navegadores vão renderizar a altura em zero (como se fosse "**height:0**").

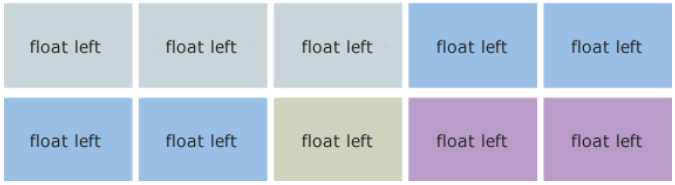


Se você der um "*clear*" antes de fechar a *tag* do elemento pai, você conserta isso.

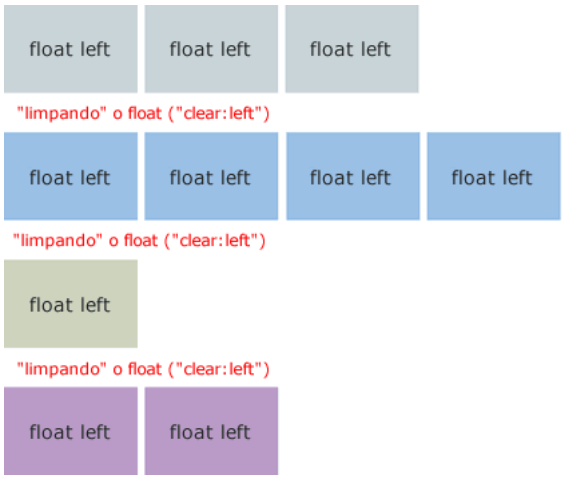


Limpar o *float* para começar uma nova linha

Vamos dizer que você tem uma série de elementos flutuantes.

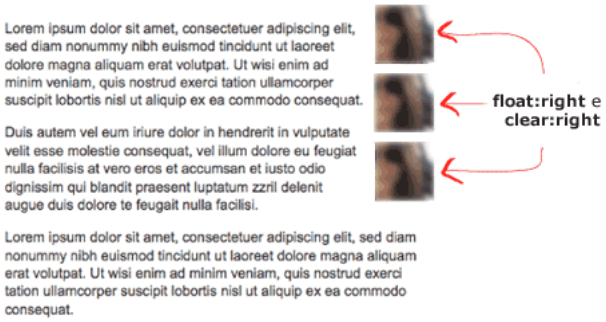


Então, digamos que você quer criar uma pausa nesta grade de elementos, a fim de iniciar uma nova linha. Porque, você sabe, iss faz sentido apenas visualmente falando.



Dando "*clear*" somente à esquerda ou à direita

Os dois exemplos acima são exemplos genéricos de como fazer o uso de "clear" para fazer a "limpeza", ou seja, dar *clear* tanto à direita, quanto à esquerda. Porque *floats* podem ser tanto à direita, como à esquerda, e você pode, evidentemente, dar um *float* somente à esquerda ou somente à direita. Isso pode ser útil quando fazer a limpeza de ambos os lados ("**clear:both**") seja problemático.



Se tivesse sido dado um clear em ambos os lados ("**clear:both**") no exemplo acima, a segunda imagem teria sido empurrada para baixo, onde termina o bloco de texto.

Diferentes técnicas de clear

Assim como tudo em CSS, há mais de um jeito para fazer isso.

Usando "*clear*" exatamente onde você precisa

A propriedade CSS "*clear*" faz exatamente o que "diz na caixa". O problema reside em onde e como aplicar um elemento à página com a propriedade *clear* correta.

- **Aplicar “clear:both” ao elemento imediatamente seguinte ao que você precisa que seja “limpo”.** Tomemos um exemplo perto do topo de um *layout* de *web page* com o cabeçalho e rodapé de largura total, com o conteúdo principal flutuando à esquerda e uma barra lateral à direita. A fim de que o rodapé aparece no lugar certo, você **deve** limpar a flutuação antes dele. Neste simples exemplo, você pode aplicar aplicar **clear:both** na própria div do rodapé.

Esta técnica é maravilhosa, já que ela funciona bem sem a necessidade de códigos supérfluos. No entanto, ela às vezes falha em sites dinâmicos. E se, por exemplo, você teve que acrescentar um novo elemento de página acima do rodapé e, consequentemente, abaixo dos outros conteúdos da página? Agora você precisaria dar um *clear* neste elemento ao invés de no rodapé. Muitas vezes é mais fácil pensar sobre *onde* é preciso dar um *clear* ao invés de *em qual elemento* é preciso dar um *clear*.

- **Aplicar “*clear*” e limpar o *float* em um elemento vazio.** DIVs são muito boas porque, geralmente, você não tem nenhum estilo aplicado a elas (como em um elemento de parágrafo “p”, por exemplo) e elas não têm nenhuma funcionalidade especial (como um “
”). Onde for preciso limpar o *float*, apenas insira: **<div style=”clear: both;”> **. Usar estilos *in-line* não é muito “atraente”, então é melhor fazer uma classe “*clear*” e criar uma regra que faça a limpeza do *float*, mas isso é apenas uma questão de gosto.

O método da DIV vazia:

```
...parte de elemento flutuante.
</div>

<div class="clear"></div>

<p> ... ahhhh, recebi um "clear"! </p>
```

CSS:

```
div.clear { clear: both; }
```

“overflow:auto” no elemento pai

É difícil explicar o motivo, mas a aplicação da propriedade CSS “**overflow:auto**” no elemento pai fará com que sua altura seja calculada corretamente. Ele irá se expandir para englobar os elementos flutuadores (*floats*) ao invés de “dar pau”. Isso pode ser muito útil e é muito “limpo”, mas tem alguns inconvenientes. O maior deles é que, frequentemente, não faz sentido usar a regra CSS no elemento pai. Pense nas vezes em que você precisa dar um *clear* em vários elementos que têm um elemento pai em comum; nesse caso, isso não vai ajudá-lo.

Outro problema é que você pode querer usar a propriedade “*overflow*” para outros fins. E se você quiser esconder o *overflow* de uma *div*, em especial? Você não pode. Você terá que englobar a *div* com outra div para conseguir isso.

O *hack* “*clearfix*” (dar o clear com a pseudo-classe “:after”)

Embora antigo, o artigo do positioniseverything (http://www.positioniseverything.net/easyclearing.html) sobre como usar uma pseudo-classe CSS para dar o *clear* ainda é válido. Explicando rapidamente, a técnica consiste em adicionar um pouco de conteúdo **depois** do elemento. Este pouco de conteúdo (um espaço, geralmente) é que faz o clear funcionar, mas fica oculto do visitante.

Aqui está um código, aplicado em uma classe CSS para qualquer elemento que precise de um *clear*:

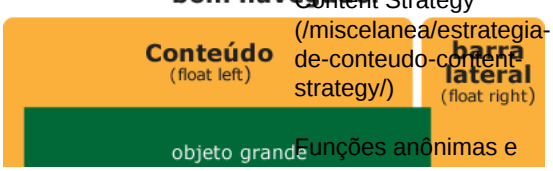
```
/* This needs to be first because FF3 is now supporting this */
.clearfix {display: inline-block;}

.clearfix:after {
content: " ";
display: block;
height: 0;
clear: both;
font-size: 0;
visibility: hidden;
}

/* Hides from IE-mac */
* html .clearfix { height: 1%; }
.clearfix { display: block; }
/* End hide from IE-mac */
```

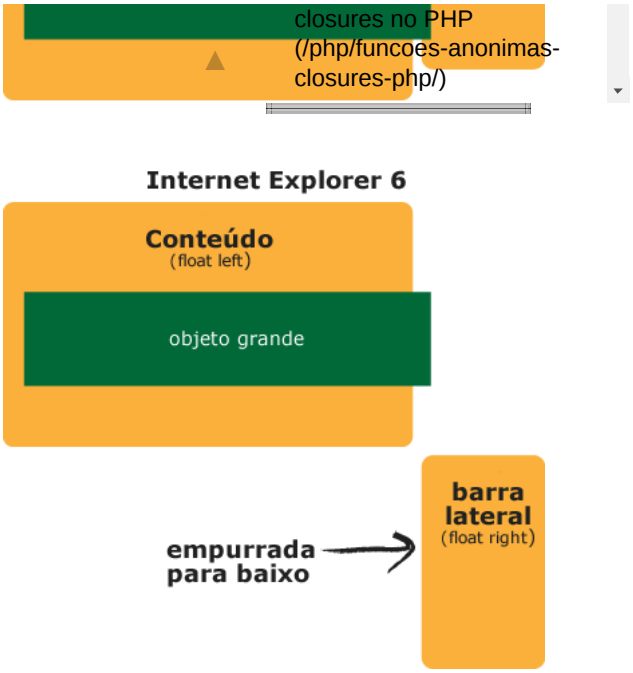
O artigo citado alerta que a técnica já está ficando velha e fala sobre "**overflow:auto**". Eu não compartilho da opinião de que a técnica está ficando ultrapassada. Esta é completamente diferente da outra técnica de *overflow*. Com a "*clearfix*", você aplica a classe no **próprio elemento**, não no elemento pai. Isso significa que você pode usá-la mesmo quando não se trata de um elemento pai, isto significa que poderá continuar a usá-la, ainda que não exista um elemento pai que faça sentido, e aplicá-la pensando **onde** o elemento com *float* precisa de um *clear*.

Outro problema com *float*: empurrões para baixo ("*pushdown*")

Esta questão merece uma atenção especial, porque frequentemente as pessoas têm problemas com questões de "empurrões". Olhando novamente o exemplo dado no início deste artigo, a *div* do conteúdo principal flutua à esquerda da *div* da barra lateral. Esta é uma estrutura comum em muitos *blogs*.

Quer sejam elementos fixos ou fluidos, ambas as DIVs têm uma largura especificada. Eles supostamente comportam-se como elementos flutuantes, ou seja, se um elemento dentro da área do conteúdo principal tem a largura maior que toda esta área (por exemplo, uma imagem muito larga) ele simplesmente se extende através e cobre qualquer coisa que esteja "no caminho". A forma como o IE6 trata isso é bem diferente. Se um elemento tem a largura maior que a de seu elemento pai, no IE6 a barra lateral vai ser simplesmente empurrada para baixo ("*pushdown*"), quebrando completamente o *layout*.





A solução? A melhor solução é não colocar elementos com largura maior que a de seu elemento pai. Para se protejer melhor, e se seu *layout* funciona de outra forma (não especifique alturas!), você pode trabalhar com "**overflow:hidden**" para esconder algo que seja preciso. Ainda há outra solução, que seria usar posicionamento absoluto para posicionar a barra lateral mais à direta. Lembre-se, entretanto, que posicionamento absoluto retira o elemento do fluxo da página – algo para se levar em consideração.

Há algum tempo saiu no A List Apart (http://alistapart.com/) um artigo sobre fazer um "falso" posicionamento absoluto, Faux Absolute Positioning (http://alistapart.com/articles/fauxabsolutepositioning), que é uma leitura interessante e aborda uma nova técnica de *layout* que traz inúmeros benefícios do posicionamento absoluto, conservando o fluxo da página e não indo de encontro à "fragilidade" dos elementos com *float*.

"Quirks" sobre elementos float

Outra coisa para se lembrar quando se lida com IE6 é que, se você aplicar uma margem (CSS "margin") no mesmo sentido que o float ("left" ou "right"), ela deve ser o dobro da margem (http://www.cssnewbie.com/double-margin-float-bug/).

Para o IE7, existe um pequeno truque sobre sua maneira peculiar de não respeitar margens inferiores (http://www.maratz.com/blog/archives/2006/11/11/ie-7-quirks-floats-and-margins/) (CSS "margin-bottom") em elementos filhos dentro de objetos flutuantes.

Artigos relacionados

- Centralizar suas páginas web com CSS: como deixar o site sempre no centro com folhas de estilo

(http://desenvolvimentoparaweb.com/css/centralizar-suas-paginas-web-com-css-como-deixar-o-site-sempre-no-centro-com-folhas-de-estilo/)

- Diferenças entre IDs e Classes (http://desenvolvimentoparaweb.com/css/diferencas-entre-ids-e-classes/)
- Textarea: dicas e truques que você sempre quis saber (http://desenvolvimentoparaweb.com/html/textarea-dicas-truques-textarea/)
- URLs longas apresentadas corretamente com CSS (http://desenvolvimentoparaweb.com/css/urls-longas-apresentadas-corretamente-com-css/)
- Como fazer loading em conteúdos do site (http://desenvolvimentoparaweb.com/jquery/como-fazer-loading-site-conteudos/)

(https://twitter.com/intent/tweet?url=<%= url %><%= via %>&text=<%= text %>&hashtags=<%= hashtags %>&related=ivyapp%3AGet the latest news and updates on Ivy.filament_io%3AApps updates and tips for making your visitors happier)
(https://www.facebook.com/sharer/sharer.php?u=<%= url %>) (mailto:?subject=<%= subject %>&body=<%= text %>)