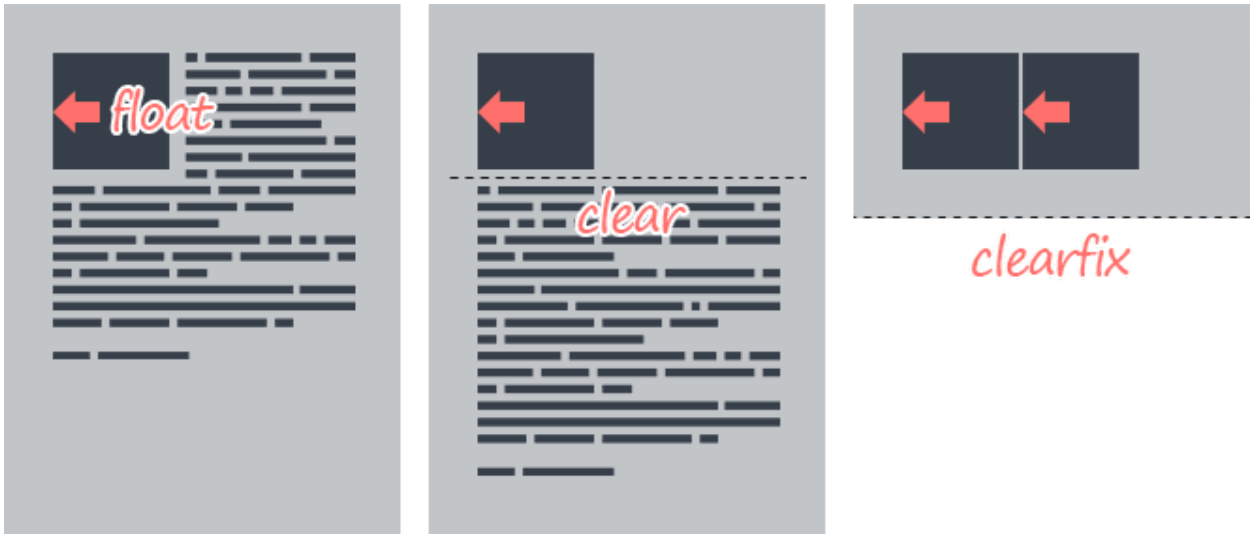
Entendendo float, clear e clearfix de uma vez por todas

22 DE MAIO DE 2014 / 4 COMENTÁRIOS

O que é **float**? Pra que serve? E **clear**? Onde e quando posso aplicar? Por que quando se usa um às vezes é necessário usar o outro?
E esse tal de **clearfix** que a gente vê em todo código? Pra que serve?

De tanto ver aplicações equivocadas, achei que pudesse ser necessário explicar de uma forma que fique [bem] clara cada uma dessas propriedades.



Float

A propriedade CSS `float` determina a retirada de um elemento do fluxo normal

do documento, “flutuando-o” à **esquerda** ou à **direita**, fazendo com que texto e elementos em linha o contornem. Devem sempre vir antes de qualquer outro elemento.

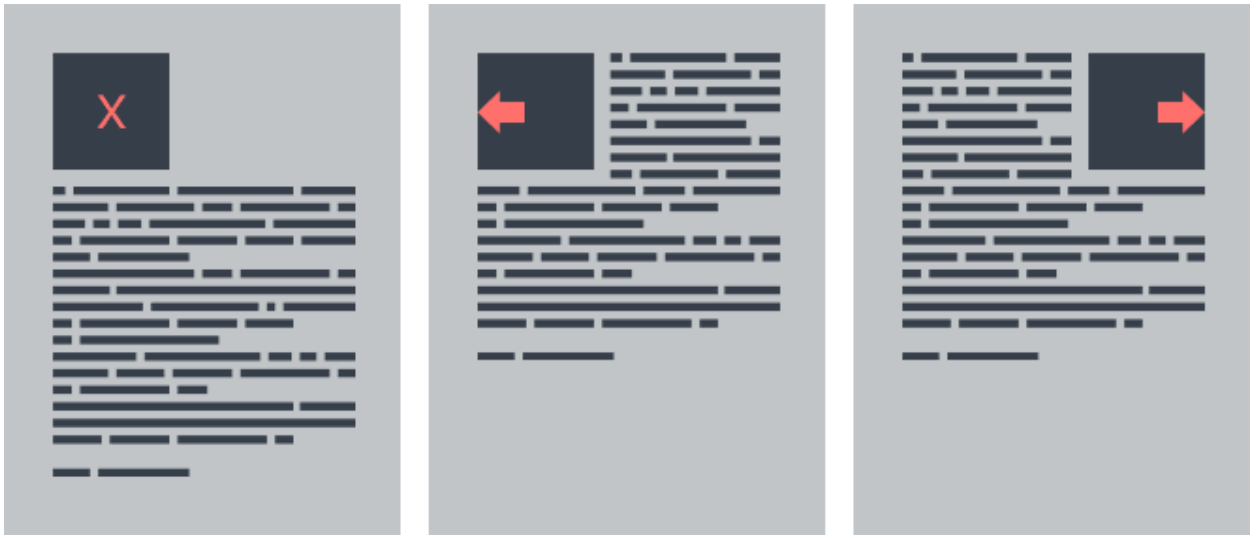
```
<div class="parent">
    <div class="float">Elemento flutuante</div>
    <div class="content">
        ...
    </div>
</div>
```

Pode ser atribuído a qualquer elemento HTML, desde que este não esteja com posicionamento absoluto (`position: absolute;`).

Sintaxe:

```
float: none | left | right | inherit
```

Valor	Comportamento
none	O elemento não flutua (*default*).
left	O elemento gera uma caixa (bloco) que flutua à esquerda. O conteúdo flui no seu lado direito, começando do topo (desde que caiba dentro do container, caso contrário começa logo abaixo).
right	Idem ao `left`, porém flutuando à direita e o conteúdo fluindo no seu lado esquerdo.
inherit	Herda o valor de float do elemento pai, caso seja declarado..



Elementos float são, em sua essência, elementos de caixa.
Não é necessário atribuir `display: block;` pois a propriedade **float** sobrepõe a propriedade **display**.

```
.classe {
    float: left;
    display: block; /* <-- desnecessário */
}
```

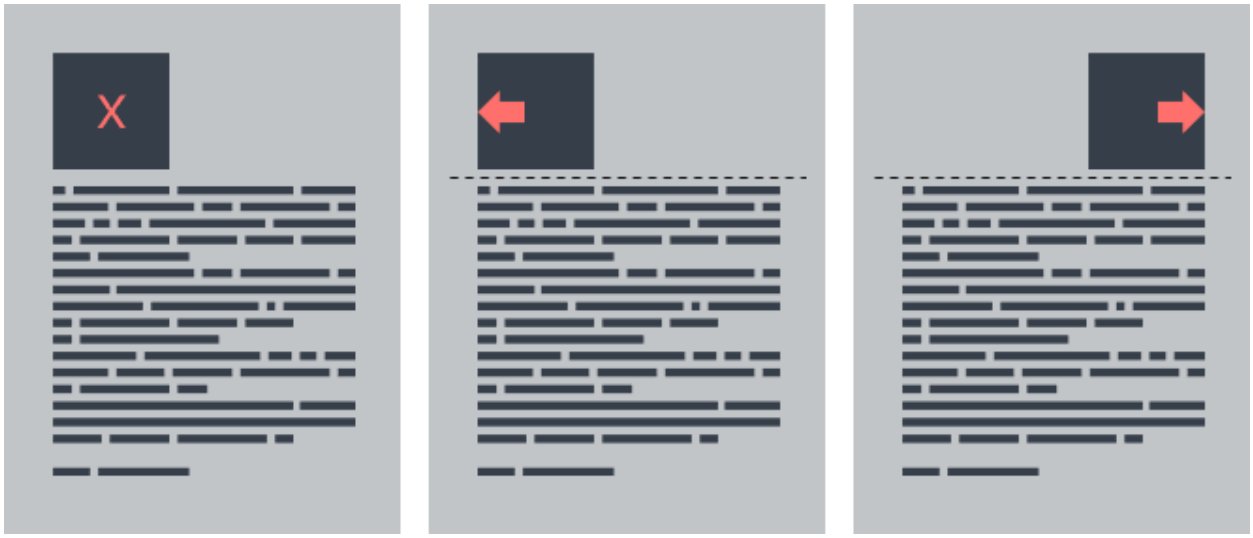
Curiosidade: Na época do IE6 tínhamos que atribuir `display: inline;` para a maioria dos elementos flutuantes. Era uma maneira de evitar um bug indesejado apelidado de "double margin bug". Coisas de IE6…

Clear

A regra é quase sempre válida: Se você usou `float`, muito provavelmente vai precisar usar `clear`.

`clear` não é para o elemento que flutua, e sim para o próximo elemento na ordem do documento.

Dito isso, qualquer conteúdo subsequente a um elemento flutuante irá contorná-lo até que este use a propriedade clear.



Na figura acima o texto nas páginas central e direita usa a propriedade `clear`.

```
<style>
.clear {
    clear: both;
}
</style>

<div class="parent">
    <img class="float" alt="" />
    <div class="clear">
        ... /* <-- conteúdo permanece abaixo da imagem */
    </div>
</div>
```

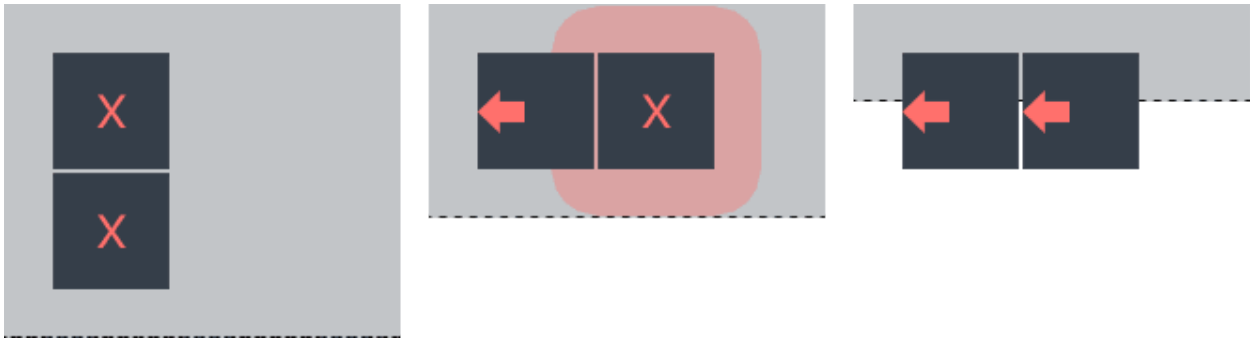
Sintaxe:

```
clear: none | left | right | both | inherit
```

Valor	Comportamento
none	Não impede que seu conteúdo contorne elementos flutuantes (*default*).
left	Evita fluir seu conteúdo à direita de elementos com `float: left;` permanecendo abaixo dele.
right	Idem ao `left`, evitando fluir seu conteúdo à esquerda de elementos com `float: right`
both	Idem `left` e `right`, o conteúdo permanece abaixo de elementos flutuantes

	independentemente para que lado flutuem.
inherit	Herda o valor de clear do elemento pai, caso seja declarado.

Você já percebeu que, ao usar elementos flutuantes dentro de um container este "perde" sua altura, deixando de contornar seus elementos filhos?



- Na figura da esquerda os blocos não flutuam.
- Na do meio o primeiro bloco flutua à esquerda mas o segundo não. Note que o container tem **a mesma altura do bloco que não flutua**.
- Na da direita os dois blocos flutuam e, como são "retirados" da ordem do documento, o container é computado com altura zero.

Para que isso não aconteça basta aplicar um elemento com `clear` logo após os elementos flutuantes e tudo se resolve.

```
<style>
.left {
    float: left;
}
.right {
    float: right;
}
.clear {
    clear: both;
}
</style>

<div class="parent">
    <div class="left">Para a esquerda</div>
    <div class="right">Para a direita</div>
    <br class="clear" />
```

```
</div>
```

Ou, aplicar `overflow: hidden;` ou `overflow: auto;` ao container.

```
<style>
.parent {
    overflow: hidden;
}
</style>

<div class="parent">
    <div class="left">Para a esquerda</div>
    <div class="right">Para a direita</div>
</div>
```

Porém é preciso alertar que `overflow: hidden|auto;` pode causar efeitos colaterais indesejados, normalmente **com elementos posicionados relativa ou absolutamente dentro do container**.

Se esse for o caso, o que você faz?

Resposta: Usa um **clearfix**.

Clearfix

Clearfix é apenas uma maneira de fazer com que um elemento tenha, ele próprio, a propriedade `clear`, sem que seja necessário usar código extra.

Clearfix não é uma propriedade CSS. É um nome comum de classe que caiu no gosto de programadores de interface.

Nos dias atuais (falo de 2014) podemos usar um "micro clearfix" para simular essa característica nos containers.

```
.clearfix:before, .clearfix:after {
    content: "";
    display: table;
}
.clearfix:after {
```

```
    clear: both;
}
```

Ainda em dúvida sobre **floats** e **clears**?

Teste o exemplo abaixo e veja se compreendeu tudo.

Não ficou claro? Quer tirar alguma dúvida ou comentar? Fique à vontade!

Outros artigos:

- Lista de itens horizontal
- Radial shadow background
- Seu CSS é bom ou ruim?