# 详说清除浮动

浮动，从诞生那天起，它就是个特别的属性——既为网页布局带来新的方法，却又随之产生一系列的问题。当然，随着时间的推移，这些问题终究有了一些出色的解决方案，Kayo 要在这里详细介绍的，除了是这些解决方案，还有其中的原理。（温謦提示：文章信息量大且篇幅长，请各位自备瓜子，饮料，音乐）。

**一.什么是清除浮动？**

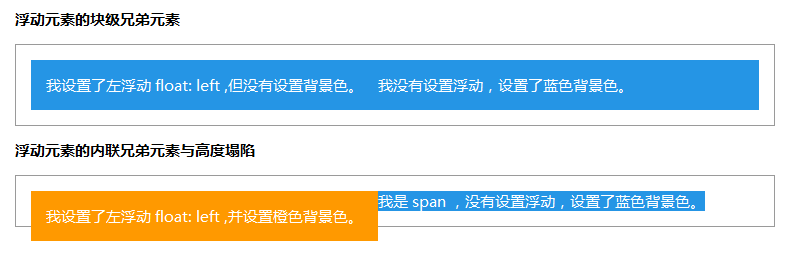
**1.浮动的缺陷**

在了解如何清除浮动之前，Kayo 先介绍为什么需要清除浮动。如本文开头所说的，浮动虽然可以便于页面布局，但同时会产生一些问题，也就是我们常说的“副作用”。而一个元素设置了浮动（即 float 值为 left, right 或 inherit 并从父元素上继承 left 或 right 值）的常见缺陷是——影响它的兄弟元素的位置和父元素产生高度塌陷，下面对这两个问题展开说明。

一个元素设置了浮动后，会影响它的兄弟元素，具体的影响方式较为复杂，这要视乎这些兄弟元素是块级元素还是内联元素，若是块级元素会无视这个浮动的块框，也就是我们平时看到的效果——使到自身尽可能与这个浮动元素处于同一行，导致被浮动元素覆盖，除非这些 div 设置了宽度，并且父元素的宽度不足以包含它们，这样兄弟元素才会被强制换行；若是内联元素，则会尽可能围绕浮动元素。

另外，浮动的元素脱离了普通流，这样使得包含它的父元素并不会因为这个浮动元素的存在而自动撑高，这样就会造成高度塌陷。

下面是演示效果图（点击图片打开大图，下同）

[](http://kayosite.com/wp-content/uploads/2012/10/clearfloat1.png)

关于这几点的更多说明，请看 [Demo](http://kayosite.com/demo/clearfloat.html#float-1) 。

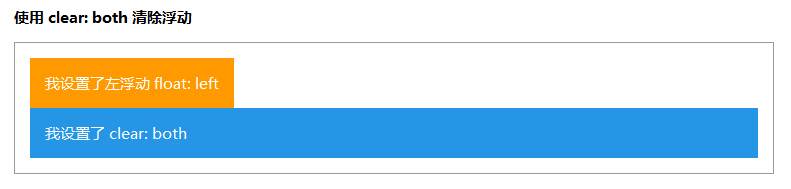
很显然，无论是影响兄弟元素还是高度塌陷的问题，都不是我们使用浮动的目的，设置浮动，只是为了改变一个元素的布局，但最终的结果却造成了更多不必要的影响，这不利于布局，因此我们需要清除这些额外的影响，也就是本文要介绍的清除浮动，其实更加准确的说，是清除浮动带来的额外影响。

**2.清除浮动的常见方法**

了解了为什么要清除浮动后，这里可以开始介绍清除浮动的常见方法了，不过这里并不急于探讨这些方法的原理，首先列出几种常见清除浮动的方法，再作探讨。

说起清除浮动，大家肯定会想起 clear: both ，的确，这是 CSS 中清除浮动的属性，clear 有 both/left/right/none/inherit 几个属性值，分别代表在元素左右两侧不允许出现浮动元素/左侧不允许出现浮动元素/右侧不允许出现浮动元素/不清除浮动/继承父元素的值。

如下图为清除浮动的例子：

[](http://kayosite.com/wp-content/uploads/2012/10/clearfloat5.png)

也可以看 [Demo](http://kayosite.com/demo/clearfloat.html#float-2) 。

从例子中可以看出，设置了 clear: both （当然在该例子中也可以为 clear: left）的元素不会跟浮动元素同行，并且会占据新的一整行，而不是根据内容来自动调整宽度。之所以会这样，要从 clear 的原理说起，clear 会为元素添加足够的空白空间，使到该元素的位置会放置在它前一个浮动元素之下，这跟增加元素外边距使到元素占据满行而强制换行的效果是一样的，事实上在 CSS1 和 CSS2 中，清除浮动正是通过自动为清除元素（即设置了 clear 属性的元素）增加外边距实现的，从 CSS 2.1 开始改为增加额外的空白空间，不改变外边距。现在大家应该清楚了，既然是增加足够的空间使到元素换行，那么最稳妥的办法就是使到该元素占据一整行，也就是 Demo 中的效果。

现在清除了浮动，但是，这只是清除了浮动对于兄弟元素的影响，而高度塌陷的问题还没有解决，因此，我们需要更高级的清除浮动——闭合浮动。

为什么叫闭合浮动？因为浮动的元素脱离了普通流，因此对于它的父元素，它并没有闭合，这时候就需要闭合浮动了。这个问题的解决方法经过多年的发展，已经有了比较完善的方法，下面 Kayo 为大家详细介绍三种常用方法。

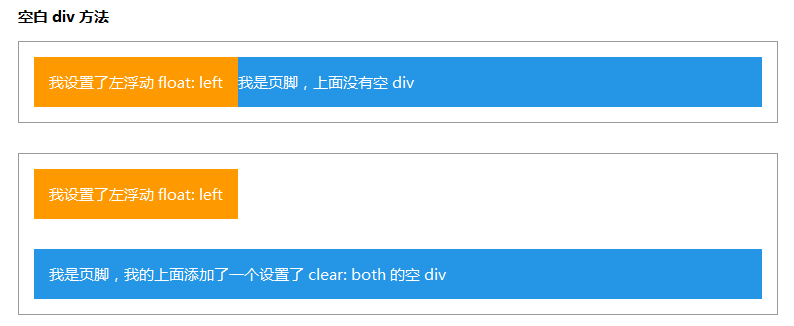
**(1) 空 div 方法**

这是较为古老的方法了，除了 div ，也有使用其他标签的，但 div 更为适用，因为除了浏览器赋予它的 display: block 外，它没有其他的样式了，也不会有特殊的功能，干干净净。这里插一段题外话，display: block 是浏览器赋予 div 的，存在于浏览器的 user agent stylesheet ，而不是 div 默认 display 的值就为 block ，在 W3C 中，所有的 HTML 标签 display 的默认值都为 inline 。

下面代码中使用到的 box main left aside 为预先设置了相关 CSS 的类，具体可以查看 Demo 的源码，在其他例子中也是如此。

|  |  |
| --- | --- |
| 1  2  3  4  5 | <div class="box">      <div class="main left">我设置了左浮动 float: left</div>      <div style="clear: both;"></div>      <div class="aside">我是页脚，我的上面添加了一个设置了 clear: both 的空 div</div>  </div> |

效果如图：

[](http://kayosite.com/wp-content/uploads/2012/10/clearfloat2.png)

也可以看 [Demo](http://kayosite.com/demo/clearfloat.html#float-3) 。

空 div 方法很方便，但是加入了没有涵义的 div ，这违背了结构与表现分离的原则，并且后期维护也不方便。

**(2) overflow 方法**

在浮动元素的父元素上设置了 overflow 的值为 hidden 或 auto ，可以闭合浮动。另外在 IE6 中还需要触发 hasLayout ，例如为父元素设置容器宽高或设置 zoom：1

|  |  |
| --- | --- |
| 1  2  3  4 | <div class="box" style="overflow: hidden; \*zoom: 1;">      <div class="main left">我设置了左浮动 float: left</div>      <div class="aside left">我是页脚，但是我也设置了左浮动。</div>  </div> |

效果如图：

[](http://kayosite.com/wp-content/uploads/2012/10/clearfloat3.png)

也可以看 [Demo](http://kayosite.com/demo/clearfloat.html#float-4) 。

这个方法相对前者更加方便，也更加符合语义要求，只是 overflow 并不是为了闭合浮动而设计的，因此当元素内包含会超出父元素边界的子元素时，可能会覆盖掉有用的子元素，或是产生了多余的滚动条。这也是在 overflow 方法诞生后依然需要寻找更佳方法的原因。

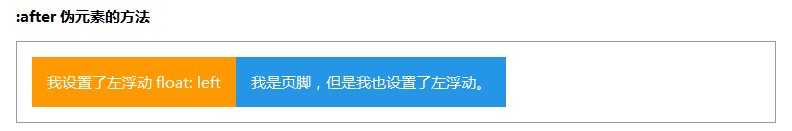
**(3) 使用 :after 伪元素的方法**

该方法来源于 [positioniseverything](http://www.positioniseverything.net/easyclearing.html" \o "How To Clear Floats Without Structural Markup" \t "_blank)， 结合 :after 伪元素（注意这不是伪类，而是伪元素，代表一个元素之后最近的元素）和 IEhack ，可以完美兼容当前主流的各大浏览器，这里的 IEhack 指的是触发 hasLayout ，具体请看下面的方法。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <style>      .clearfix {/\* 触发 hasLayout \*/ zoom: 1; }      .clearfix:after {content: &quot;.&quot;; display: block; height: 0; clear: both; visibility: hidden; }  </style>  <div class="box clearfix">      <div class="main left">我设置了左浮动 float: left</div>      <div class="aside left">我是页脚，但是我也设置了左浮动。</div>  </div> |

显然，相对来说，这个办法不但完美兼容主流浏览器，并且也很方便，使用重用的类，可以减轻代码编写，另外网页的结构也会更加清晰。

效果如图：

[](http://kayosite.com/wp-content/uploads/2012/10/clearfloat4.png)

也可以看 [Demo](http://kayosite.com/demo/clearfloat.html#float-5) 。

**二.清除浮动方法的实质 —— CSS clear 与 BFC 特性**

通过上面的例子，我们不难发现清除浮动的方法可以分成两类：

一是利用 clear 属性，包括在浮动元素末尾添加一个带有 clear: both 属性的空 div 来闭合元素，其实利用 :after 伪元素的方法也是在元素末尾添加一个内容为一个点并带有 clear: both 属性的元素实现的。

二是触发浮动元素父元素的 BFC (Block Formatting Contexts, 块级格式化上下文)，使到该父元素可以包含浮动元素，关于这一点，下面 Kayo 为大家进行详细的介绍。

BFC 在 CSS 的可视化格式模型 (Visual Formatting Model) 中具有非常重要的地位，很多开发者因为不了解 BFC 的特性而在实际开发中产生很多让人感到莫名其妙的问题。尽管如此，因为 BFC 涉及 CSS 中很少接触的部分，因此国内的相关介绍很少，这里 Kayo 展开说明一下。

**1.BFC 是什么？**

BFC (Block Formatting Contexts) 即块级格式化上下文，从样式上看，它与普通的容器没有什么区别，但是从功能上，BFC 可以看作是隔离了的独立容器，容器里面的元素不会在布局上影响到外面的元素，并且 BFC 具有普通容器没有的一些特性，例如可以包含浮动元素，上面的第二类方法（如 overflow 方法）就是触发了父元素的 BFC ，使到它可以包含浮动元素，从而防止出现高度塌陷的问题。

**2.如何触发 BFC**

触发 BFC 的条件如下：

* 浮动元素，float 除 none 以外的值
* 绝对定位元素，position（absolute，fixed）
* display 为以下其中之一的值 inline-blocks，table-cells，table-captions
* overflow 除了 visible 以外的值（hidden，auto，scroll）

在 CSS3 中，BFC 叫做 Flow Root，并增加了一些触发条件：

* display 的 table-caption 值
* position 的 fixed 值，其实 fixed 是 absolute 的一个子类，因此在 CSS2.1 中使用这个值也会触发 BFC ，只是在 CSS3 中更加明确了这一点。

**3.BFC 的特性**

BFC 主要有三个特性：

**(1) BFC 会阻止外边距折叠**

两个相连的 div 在垂直上的外边距会发生叠加，有些书籍会把这个情况列作 bug ，这里 Kayo 并不同意，这种折叠虽然会给不熟悉 CSS 布局的开发者带来一些不便，但实际上它具有完整且具体的折叠规则，并且在主流浏览器中都存在，因此 Kayo 更认为这应该是 CSS 的特性。当然，在实际开发中，或许我们有时会不需要这种折叠，这时可以利用 BFC 的其中一个特性——阻止外边距叠加。

**(2) BFC 可以包含浮动的元素**

这也正是上面使用 overflow: hidden 与 overflow: auto 方法闭合浮动的原理，使用 overflow: hidden 或 overflow: auto 触发浮动元素父元素的 BFC 特性，从而可以包含浮动元素，闭合浮动。

W3C 的原文是“['Auto' heights for block formatting context roots](http://www.w3.org/TR/CSS2/visudet.html#root-height)”，也就是 BFC 会根据子元素的情况自动适应高度，即使其子元素中包括浮动元素。

但是 IE6-7 并不支持 W3C 的 BFC ，而是使用自产的 hasLayout 。从表现上来说，它跟 BFC 很相似，只是 hasLayout 自身存在很多问题，导致了 IE6-7 中一系列的 bug 。触发 hasLayout 的条件与触发 BFC 有些相似，具体情况 Kayo 会另写文章介绍。这里 Kayo 推荐为元素设置 IE 特有的 CSS 属性 zoom: 1 触发 hasLayout ，zoom 用于设置或检索元素的缩放比例，值为“1”即使用元素的实际尺寸，使用 zoom: 1 既可以触发 hasLayout 又不会对元素造成其他影响，相对来说会更为方便。

**(3) BFC 可以阻止元素被浮动元素覆盖**

如上面所说，浮动元素的块状兄弟元素会无视浮动元素的位置，尽量占满一整行，这样就会被浮动元素覆盖，为该兄弟元素触发 BFC 后可以阻止这种情况的发生。

为了说明 BFC 对于清除浮动的本质，Kayo 在这里使用了较大的篇幅介绍了 BFC ，但 BFC 并不是本文要说明的中心，并且 BFC 的机制也相当复杂，因此 Kayo 会基于以上 BFC 的内容另写一篇更详细的文章介绍 BFC ，并制作 Demo 说明实际情况。

**三.清除浮动的更多方法**

在了解了清除浮动的实际原理后，我们不难想象，清除浮动的方法并不只上面提到的三种，例如，利用 BFC 特性，还可以使用以下的方法触发 BFC 并清除浮动。

给浮动元素的父元素添加浮动，但是这样需要一直浮动到 body ，不建议采取该方法。

给浮动元素的父元素添加 display: table-cells ，但是这样无疑改变了盒子模型，也不建议使用。

可以看出，以上这些方法虽然也比较方便，但同时也有很明显的缺点，这也导致了上面三种方法会更加适合实际项目中使用。

结合语义化的要求，在实际的项目中，Kayo 建议可以使用 overflow 方法或 :after 伪元素方法。使用 overflow 方法，较为方便，在如内部元素全部为浮动元素，并且内容不会超出父元素框等情况可以直接采用 overflow 方法，但该方法毕竟会触发 BFC ，上面已经提到，BFC 的特性是有多个的，为了避免不必要的影响，如果实际需要清除浮动元素的布局比较复杂，可以直接采用 :after 伪元素方法。