# 详说 Block Formatting Contexts (块级格式化上下文)

在上文[《详说清除浮动》](http://kayosite.com/remove-floating-style-in-detail.html" \o "详说清除浮动" \t "_blank)中，Kayo 较为详细地介绍了 BFC ，也就是本文的主角 Block Formatting Contexts (块级格式化上下文)，本文会基于上文关于 BFC 的部分进行拓展，进一步说明 BFC 的特性。

但在进一步说明 BFC 特性之前，Kayo 首先要介绍另一个在 CSS 的可视化格式模型 (Visual Formatting Model) 中具有非常重要地位的概念——定位方案。定位方案是控制元素的布局，在 CSS 2.1 中，有三种定位方案——普通流 (Normal Flow) 、浮动 (Floats) 和绝对定位 (Absolute Positioning) ，下面分别对这三种布局简略说明一下。

普通流(Normal Flow)

在普通流中，元素按照其在 HTML 中的先后位置至上而下布局，在这个过程中，行内元素水平排列，直到当行被占满然后换行，块级元素则会被渲染为完整的一个新行， 除非另外指定，否则所有元素默认都是普通流定位，也可以说，普通流中元素的位置由该元素在 HTML 文档中的位置决定。

浮动 (Floats)

在浮动布局中，元素首先按照普通流的位置出现，然后根据浮动的方向尽可能的向左边或右边偏移，其效果与印刷排版中的文本环绕相似。

绝对定位 (Absolute Positioning)

在绝对定位布局中，元素会整体脱离普通流，因此绝对定位元素不会对其兄弟元素造成影响（如果看了上文的童鞋，会发现这点与浮动元素会影响兄弟元素是不同的），而元素具体的位置由绝对定位的坐标决定。

BFC 正是属于普通流的，因此它对兄弟元素也不会造成什么影响。

**一. BFC 是什么？**

有了上面的基础后，可以正式介绍 BFC 了。从样式上看，具有 BFC 的元素与普通的容器没有什么区别，但是从功能上，具有 BFC 的元素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，上文中的第二类清除浮动的方法（如 overflow 方法）就是触发了浮动元素的父元素的 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。

简单来说，BFC 就是一种属性，这种属性会影响着元素的定位以及与其兄弟元素之间的相互作用。

**二.如何触发 BFC**

上面介绍了 BFC 的定义，那么如何触发 BFC 呢？

满足下面任一条件的元素，会触发为 BFC ：

* 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）

但是，"display:table" 本身并不产生 BFC，而是由它产生匿名框，匿名框中包含 "display:table-cell" 的框会产 BFC。 总之，对于 "display:table" 的元素，产生 BFC 的是匿名框而不是 "display:table"。

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。

值得注意的是，在前面 Kayo 已经说明过了，BFC 并不是元素，而是某些元素带有的一些属性，因此，是上面这些元素产生了 BFC ，而它们本身并不是 BFC ，这个概念需要区分清楚。

**三. BFC 的特性**

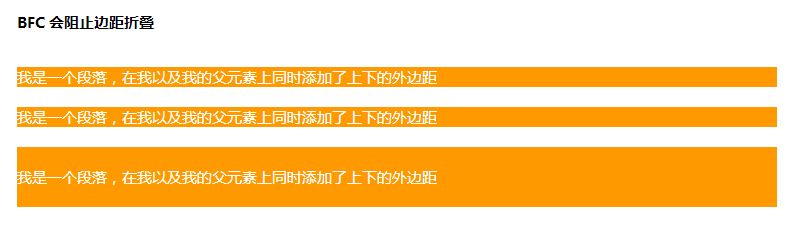
从整体上看，BFC 是隔离了的容器，这个具体可以表现为三个特性：

**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。

在举例说明 BFC 如何阻止外边距折叠之前，首先说明一下外边距折叠的规则：仅当两个块级元素相邻并且在同一个块级格式化上下文时，它们垂直方向之间的外边距才会叠加。也就是说，即便两个块级元素相邻，但当它们不在同一个块级格式化上下文时它们的边距也不会折叠。因此，阻止外边距折叠只需产生新的 BFC 。

效果如图：

[](http://kayosite.com/wp-content/uploads/2012/10/bfc-1.png)

也可以看 [Demo](http://kayosite.com/demo/bfc.html#bfc-1) 。

如上图的例子，三个 div 各包含一个 p 元素，三个 div 及其包含的 p 元素都有顶部和底部的外边距，但只有第三个 div 的边距没有与它的子元素 p 的外边距折叠。这是因为第三个 div 创建了新的 BFC ，由此可见：创建了 BFC 的元素，不和它的子元素发生外边距折叠。

**2. BFC 可以包含浮动的元素**

这也正是使用 overflow: hidden 与 overflow: auto 方法闭合浮动的原理，使用 overflow: hidden 或 overflow: auto 触发浮动元素的父元素的 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”，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。

效果如图：

[](http://kayosite.com/wp-content/uploads/2012/10/bfc-2.png)

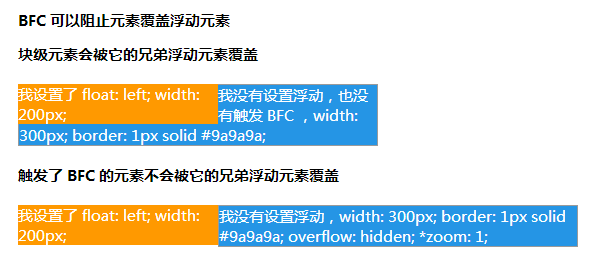
也可以看 [Demo](http://kayosite.com/demo/bfc.html#bfc-2) 。

上面的例子中，有两个 div ，它们各包含一个设置了浮动的 p 元素，但第一个 div 出现了“高度塌陷”，这是因为内部的浮动元素脱离了普通流，因此该 div 相当于一个空标签，没有高度和宽度，即高度为 0 ，上下边框也重叠在一起。而第二个 div 使用 overflow: hidden 触发了 BFC ，可以包含浮动元素，因此能正确表现出高度，其边框位置也正常了。

**3. BFC 可以阻止元素被浮动元素覆盖**

如上文所说，浮动元素的块级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。

效果如图：

[](http://kayosite.com/wp-content/uploads/2012/10/bfc-31.png)

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

如上图的例子，蓝色背景的 div 使用 overflow: hidden 触发了 BFC ，它并不会被它的兄弟浮动元素覆盖，而是处于它的旁边。值得注意的是，以上的情况仅仅是元素宽度之和没有超出父元素宽度的情况，假设浮动元素宽度和它的非浮动兄弟元素宽度都没有超过父元素宽度，但两个元素的宽度加起来超出了父元素宽度的时候，非浮动元素会下降到下一行，即处于浮动元素下方，效果如下图：

[](http://kayosite.com/wp-content/uploads/2012/10/bfc-4.png)

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

**四. BFC 与 hasLayout**

细心的童鞋会发现，在上面的例子中，除了使用 overflow: hidden 触发 BFC 外，还使用了一个 \*zomm: 1 的属性，这是 IEhack ，因为 IE6-7 并不支持 W3C 的 BFC ，而是使用私有属性 hasLayout 。从表现上来说，hasLayout 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。

这时我们需要注意一个问题：既然 hasLayout 有着跟 BFC 相似的功能，那么在实际开发中，就要为需要触发 BFC 的元素同时触发 hasLayout ，这样 BFC 和 hasLayout 具有的一些特殊性质可以在现代浏览器和 IE 中同时产生，避免一个元素在不同浏览器间的表现因为 BFC 或 hasLayout 出现差异。事实上，在实际开发中很多莫名其妙的问题其实都是因此而产生的。当然同样地，如果一个元素没有触发 BFC ，也要尽量保证它没有触发 hasLayout 。