1.普通流

元素按照HTML中的位置顺序决定排布的过程。

position:

relative相对布局：相对于元素本身在文档中应该出现的位置来移动这个元素，可以通过TRBL来移动元素的位置，实际上该元素依然占据文档中原有的位置，只是视觉上相对原来的位置有移动。

absolute绝对布局：是生成绝对定位的元素，脱离了文本流（即在文档中已经不占据位置），参照浏览器的左上角通过top,right,bottom,left（简称TRBL） 定位。**如果一个元素绝对定位后，其参照物是以离自身最近元素是否设置了相对定位，如果有设置将以离自己最近元素定位，如果没有将往其祖先元素寻找相对定位元素，一直找到html为止。**

fixed固定布局：不管页面滚动条怎么移动，页面上下是否移动，fixed布局的div使用在同一个位置，有点像微信扫一扫窗口。

static静态布局：默认，设置和相对布局的效果相同，但是left，right等偏移量失效了,float是有效的。

2.十步图解CSS的position

CSS的[positon](http://www.w3.org/TR/CSS2/)，我想做为一个Web制作者来说都有碰到过，但至于对其是否真正的了解呢？那我就不也说了，至少我自己并不非常的了解其内核的运行。今天在[Learn CSS Positioning in Ten Steps](http://www.barelyfitz.com/screencast/html-training/css/positioning/)一文中分十步介绍了CSS的“**position**”中的“static、relative、absolute、float”使用，觉得蛮有意思的。整理了一下贴上来与大家一起分享。希望大家能喜欢。

在图解这十个过程之前，我将实例的代码放上来，好让大家一个实体参考：

**HTML Markup**

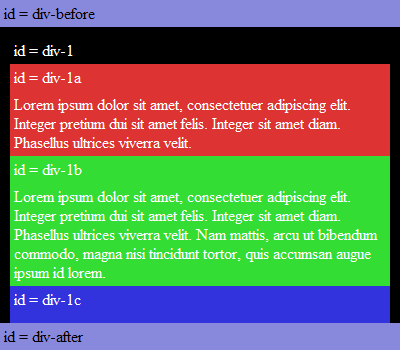
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <**div** id="example">                  <**div** id="div-before">                      <**p**>id = div-before</**p**>                  </**div**>                  <**div** id="div-1">                      <**div** id="div-1-padding">                          <**p**>id = div-1</**p**>                          <**div** id="div-1a">                              <**p**>id = div-1a</**p**>                              <**p**>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. Phasellus ultrices viverra velit.</**p**>                          </**div**>                          <**div** id="div-1b">                              <**p**>id = div-1b</**p**>                              <**p**>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. Phasellus ultrices viverra velit. Nam mattis, arcu ut bibendum commodo, magna nisi tincidunt tortor, quis accumsan augue ipsum id lorem.</**p**>                          </**div**>                          <**div** id="div-1c">                              <**p**>id = div-1c</**p**>                          </**div**>                      </**div**>                  </**div**>                  <**div** id="div-after">                      <**p**>id = div-after</**p**>                  </**div**>              </**div**> |

**CSS Code**

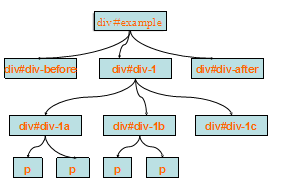
简单的运用一点样式：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | #example {  **float**: right;              }                #example p {  **margin**: 0 0.25em;  **padding**: 0.25em 0;              }              #div-before,              #div-after {  **background-color**: #88d;  **color**: #000;              }                #div-1 {  **width**: 400px;  **background-color**: #000;  **color**: #fff;              }                #div-1-padding {  **padding**: 10px;              }                #div-1a {  **background-color**: #d33;  **color**: #fff;              }                #div-1b {  **background-color**: #3d3;  **color**: #fff;              }                #div-1c {  **background-color**: #33d;  **color**: #fff;              } |

初步效果：

[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-1.png)

为了后面能更好的了解相关知识点，我特将此例的DOM草图画出来：

[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-2.png)

上面的DOM图，我想大家一定非常容易的理解，下面就一起来主要看position的使用。

第一步：position: static

在CSS中所有元素的“position”属性的默认值都是“**static**”，因为不需要显式的为每个元素设置“**position:static**”。此时大家会问，那这个属性值是不是没有任何意义呢？其实不是的，他在CSS中也会起着很大的作用。我们来看一个实例：

比如说你的两个页面，同时存在“div#div-1”，那么此时你在A面中需要对“div#div-1”进行绝对定位；而在B页面中“div#div-1”又不需要进行绝对定位。

**A页面中“div#div-1”绝对定位：**

|  |  |
| --- | --- |
| 1  2  3 | #div-1 {  **position**: absolute;              } |

此时在B页面中不想在进行绝对定位，那么我们就必须在你的样式中显式的重新设置“#div-1”的postion属性为“static”

|  |  |
| --- | --- |
| 1  2  3 | body.B #div-1 {  **position**: static;              } |

**第二步：相对定位position:relative**

**relative**称为相对定位，如果你给某个元素指定了**postion**的值为“**relative**”，那么你就可以通过“T-R-B-L”(也就是top,right,bottom,left)来设置元素的定位值。

使用relative时有几点需要注意：

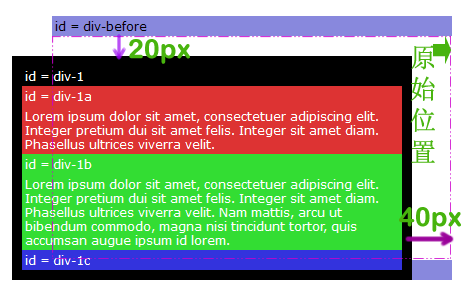
* 1. 元素设置了relative时，是相对于元素本身位置进行定位；
  2. 元素设置了relative后，可以通过“T-R-B-L”改变元素当前所在的位置，但元素移位后，同样点有当初的物理空间位；
  3. 元素设置了relative后，如果没有进行任何的“T-R-B-L”设置，元素不会进行任何位置改变。

上面三点第一点和第三点来说都是比较好理解，那么现在针对第二点，我们来看一个实例的操作：

在上面的基础上，我们对“div-1”进行向下移动20px；向左移动40px:

|  |  |
| --- | --- |
| 1  2  3  4  5 | #div-1 {  **position**:relative;  **top**:20px;  **left**:-40px;                  } |

我们来看看效果：

[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-3.png)

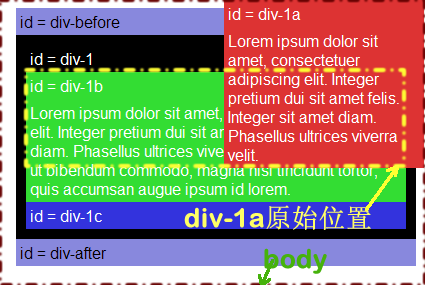
从效果图可以再次印证上面说的第二点。元素“div-1”向下移动了20px，向左移动了40px，本身位置变化了，而元素最初所占的物理空间依然还是存在，另外一点元素相对定位后并没有影响其他相邻的元素。

第三步：绝对定位position:absolute

**absolute**是position中的第三个属性值，如果你给元素指定了**absolute**，整个元素就会漂出文档流，而且元素自身的物理空间也同时消失了。不像“**relative**”还具有原先的物理空间。

我们来看一个实例,在div-1a元素上进行绝对定位：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #div-1a {  **position**:absolute;  **top**:0;  **right**:0;  **width**:200px;              } |

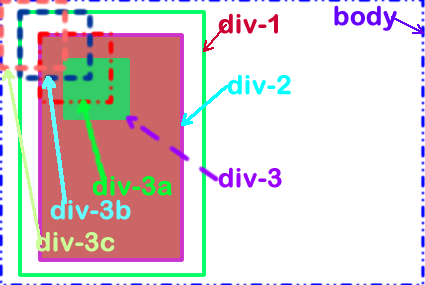
[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-4.png)

此时元素“div-1a”不在当初的文档流中，而且其此时定位也是相对于html来进行定位，那么我们有时候并不是需要这样的效果，哪果我们元素div-1a还想在div-1是进行绝对定位，那要怎么办呢？此时就要发挥前面第二步的“relative”作用了。

第四步：relative和absolute的结合

第二步中大家知道元素相对定位“relative”是相对于元素自身定位，而在第三步中大家知道元素绝对定位“absolute”是相对于html。但这种说法只有满足这样的条件才是正常的：“**绝对定位元素的任何祖先元素没有进行任何的“relative”或者“absolute”设置，那么绝对定位的元素的参考物就是html**”，这样一来，“relative”和“absolute”的结合就能起到很大的作用。

我们接下来看一个截图：

[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-5.png)

上图做为一个实例来说明“relative”和“absolute”的关系，首先上图中共有三个div放在body内，而且他们三个div的关系是“div-1>div-2>div-3”,而且在div-3有这么一个绝对定位：

|  |  |
| --- | --- |
| 1  2  3  4  5 | .div-3 {  **position**: absolute;  **left**:0;  **top**:0;              } |

下面分几个情况来说明上图的意思：

**1、div-1与div-2都没有设置“position:relative”**，此时我们的div-3绝对定位后就漂到了上图中“div-3c”的位置上；

**2、现在我们在div-2元素中加设置一个“position: relative”**，此时我们的div-3绝对定位后就漂到了上图中的“div-3a”的位置;

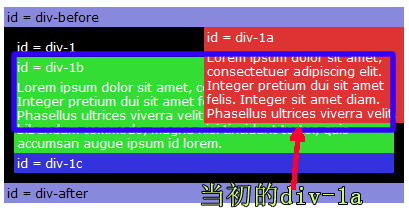
**3、接下来把相对定位的设置换到div-1元素上，**此时div-3绝对定位后就到了div-3b的位置。

花这么多心思，我只想说明一点：**如果一个元素绝对定位后，其参照物是以离自身最近元素是否设置了相对定位，如果有设置将以离自己最近元素定位，如果没有将往其祖先元素寻找相对定位元素，一直找到html为止。**这句话说起起来好像有点拗口，不知道大家能否明白我说的是什么？如果不明白大家可以参考上图或者下面这个实例效果：

回到上面的实例中，如果我们在“div-1”加一个“relative”：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #div-1 {  **position**:relative;              }              #div-1a {  **position**:absolute;  **top**:0;  **right**:0;  **width**:200px;              } |

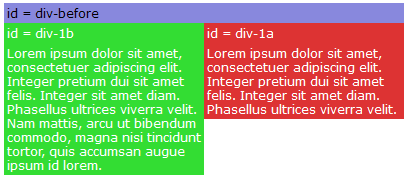
现在我们相对点不在是第三步中的body了，而是“div-1”了，大家看看与第三步的变化：

[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-6.png)

**第五步：relative和absolute实现布局效果**

这一步只要想演示一下使用相对定位和绝对定位实现的两例布局。在前面的基础上，div-1进行相对定位，而div-1a和div-1b进行绝对定位，从而实现两列布局的效果：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #div-1 {  **position**:relative;              }              #div-1a {  **position**:absolute;  **top**:0;  **right**:0;  **width**:200px;              }              #div-1b {  **position**:absolute;  **top**:0;  **left**:0;  **width**:200px;              } |

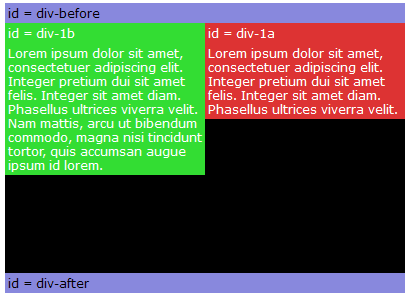
[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-7.png)

这样的制作只是用来说明absolute的作用，如果只能实现上面的效果，可能在实际制作中并不完美，为了让其更完美一些，在这个基础上我们在来看下面这一步。

第六步：设置固定高度

为了让布局更适用一些，可以在div-1元素上设置固定高度，如：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #div-1 {  **position**:relative;  **height**:250px;              }              #div-1a {  **position**:absolute;  **top**:0;  **right**:0;  **width**:200px;              }              #div-1b {  **position**:absolute;  **top**:0;  **left**:0;  **width**:200px;              } |

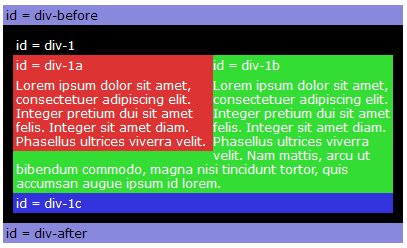
[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-8.png)

相比之下好一点，但我们并不知道元素内容高度将会是多少，所以在此设置一个固定高度也是我们实际中的一个死穴，个人不建议这样使用。如果为了需要，我们可以通过别的办法来实现。

第七步：float

前两步，使用绝对定位都并不是很理想，那么我们可以考虑使用float来解决。我们可以在一个元素上使用float，让元素向左或向右，而且还可以使用文本围绕在这个元素的周边（这个作用在文本围绕图片特别有用）。下面来模拟一下：

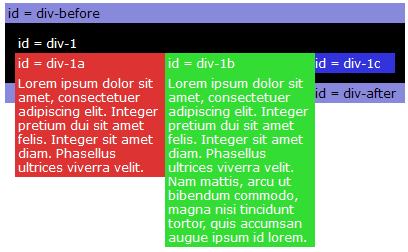
|  |  |
| --- | --- |
| 1  2  3  4 | #div-1a {  **float**:left;  **width**:200px;              } |

[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-9.png)

**第八步：多列浮动**

上面展示的是一个列浮动，接下来看看多列的变化：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #div-1a {  **float**:left;  **width**:150px;              }              #div-1b {  **float**:left;  **width**:150px;              } |

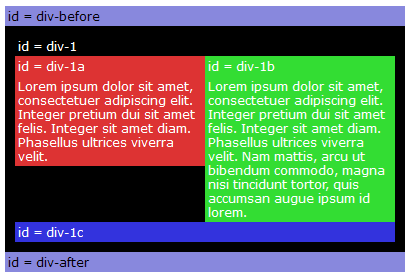
[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-11.png)

浮动与绝对定位来相比，现在解决了其高度自适应的问题，但也存在一个问题，浮动也破坏了元素当初的文档流，使其父元素塌陷了，那么为了解决这个问题，我们有必要对其进行清除浮动。

**第九步：清除浮动**

为了让浮动元素的父元素不在处于塌陷状态下，我们需要对浮动元素进行清除浮动：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #div-1a {  **float**:left;  **width**:190px;              }              #div-1b {  **float**:left;  **width**:190px;              }              #div-1c {  **clear**:both;              } |

[](http://jbcdn2.b0.upaiyun.com/2013/10/ten-css-position-10-1.png)

浮动是在css中是一个很深的课题，这里只是轻描淡写了一下。前面在《[CSS的Float之一](http://www.w3cplus.com/css/css-containing-floats)》和《[CSS的Float之二](http://www.w3cplus.com/css/css-containing-floats-part-2)》也介绍了一些有关于float的相关知识，有关于清除浮动的，大家也可以点阅《[Clear Float](http://www.w3cplus.com/css/clear-float)》。

第十步：扩展阅读

上面只是简单的介绍了一下CSS中的position的基础知识。如果大家对这一块知识感兴趣的话，可以点击下面的相关连接：

* 1. [The position declaration](http://www.quirksmode.org/css/position.html)
  2. [Absolute Positioning Inside Relative Positioning](http://css-tricks.com/absolute-positioning-inside-relative-positioning/)
  3. [CSS Positioning 101](http://www.alistapart.com/articles/css-positioning-101/)
  4. [Css position: position static, absolute, relative and fixed of an element](http://css.flepstudio.org/en/css-properties/position-static-absolute-relative-static.html)
  5. [Making the absolute, relative](http://stopdesign.com/archive/2003/09/03/absolute.html)
  6. [floatutorial](http://css.maxdesign.com.au/floatutorial/)
  7. [CSS Floats 101](http://www.alistapart.com/articles/css-floats-101/)
  8. [All About Floats](http://css-tricks.com/all-about-floats/)
  9. [float](https://developer.mozilla.org/en/CSS/float)