# CSS学习笔记

1、id作为单独的选择器。

#sidebar {

border: 1px dotted #000;

padding: 10px;

}

老版本的 Windows/IE 浏览器可能会忽略这条规则，除非你特别地定义这个选择器所属的元素：

div#sidebar {

border: 1px dotted #000;

padding: 10px;

}

选择器的两种写法：对id和class都是：

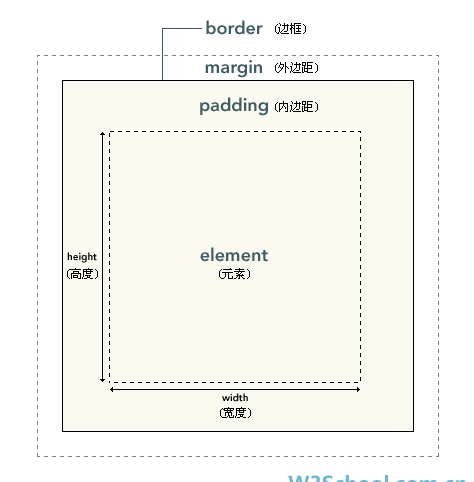
div.selector{ } ：解释为：“其 class 属性值为 selector的所有的div层”。

.selector div{ } ：解释为：“其 class 属性值为 selector的所有的元素的子元素div层”.（这其实算是一种后代选择器。）

总结： 就是说一个选择器后面以空格隔开再跟一个选择器的情况是一种父子关系（包含关系）。如div p{ } :表示选择div元素中的所有p元素。

2、CSS的盒模型。

元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。



3、CSS定位：

绝对定位与float浮动不能同时使用，比如一个大盒子里有的是绝对定位，有的是使用css float浮动定位，这样IE6浏览器将不会显示改大对象里的这些绝对定位与相对定位，这也算是IE6 CSS HACK吧，注意不要混用即可。  
  
position:absolute；position:relative绝对定位使用通常是父级定义position:relative定位，子级定义position:absolute绝对定位属性，并且子级使用left或right和top或bottom进行绝对定位。

相对定位是“相对于”元素在文档中的初始位置，而绝对定位是“相对于”最近的已定位祖先元素，如果不存在已定位的祖先元素，那么“相对于”最初的包含块。

注释：根据用户代理的不同，最初的包含块可能是画布或 HTML 元素。

提示：因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 z-index 属性来控制这些框的堆放次序。

我们尝试不用 position 属性, 但为节点加上 z-index 属性. 发现 z-index 对节点没起作用. z-index 属性仅在节点的 position 属性为 relative, absolute 或者 fixed 时生效.left和top也是。

如果所有节点都定义了 position:relative. z-index 为 0 的节点与没有定义 z-index 在同一层级内没有高低之分; 但 z-index 大于等于 1 的节点会遮盖没有定义 z-index 的节点; z-index 的值为负数的节点将被没有定义 z-index 的节点覆盖. 浮动会让z-index失效。

**1.第一种情况（z-index无论设置多高都不起作用情况）：**这种情况发生的条件有三个：   
1、父标签 position属性为relative；   
2、问题标签无position属性（不包括static）；   
3、问题标签含有浮动(float)属性。   
  
eg:z-index层级不起作用，浮动会让z-index失效

复制代码

代码如下:

<div style="position:relative; z-index:9999;">   
<img style="float:left;" src="http://image.zhangxinxu.com/image/study/s/s256/mm2.jpg" />   
</div>

解决办法有三个（任一即可）：   
1、position:relative改为position:absolute；   
2、浮动元素添加position属性（如relative，absolute等）；   
3、去除浮动。   
  
**2.第二种情况**   
IE6下，层级的表现有时候不是看子标签的z-index多高，而要看整个DOM tree（节点树）的第一个relative属性的父标签的层级。   
  
eg:IE7与IE6有着同样的bug，原因很简单，虽然图片所在div当前的老爸层级很高(1000)，但是由于老爸的老爸不顶用，可怜了9999如此强势的孩子没有出头之日啊！

复制代码

代码如下:

<div style="position:relative;">   
<div style="position:relative; z-index:1000;">   
<div style="position:absolute; z-index:9999;">   
<img src="http://image.zhangxinxu.com/image/study/s/s256/mm3.jpg" />   
</div>   
</div>   
</div>

解决办法： 在第一个relative属性加上一个更高的层级（z-index:1）

复制代码

代码如下:

<div style="position:relative; z-index:1;">   
<div style="position:relative; z-index:1000;">   
<div style="position:absolute; z-index:9999;">   
<img src="http://image.zhangxinxu.com/image/study/s/s256/mm3.jpg" />   
</div>   
</div>   
</div>

**绝对定位**

设置为绝对定位的元素框从文档流完全删除，并相对于其包含块定位，包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像该元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

绝对定位的元素的位置相对于最近的已定位祖先元素，如果元素没有已定位的祖先元素，那么它的位置相对于最初的包含块。

因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 z-index 属性来控制这些框的堆放次序。

4、float和 clear：

**浮动**

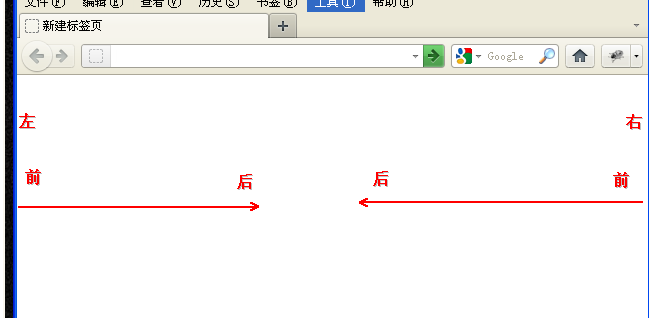
浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

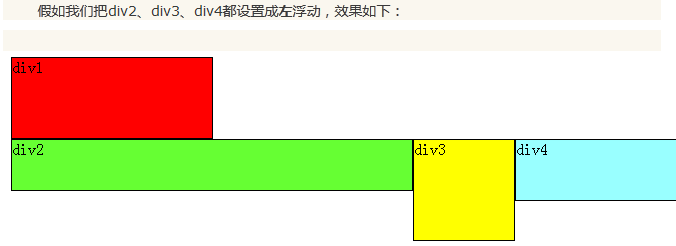
由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。这句话怎么理解呢：就是，就是说浮动框中的内容会放在最近后一个普通流的框中，但是这个内容又会在普通流框中占据其所属浮动框的宽高。而这个最近的普通流框中原来的内容会放在浮动框的后面。

**假如某个div元素A是浮动的，如果A元素上一个元素也是浮动的，那么A元素会跟随在上一个元素的后边(如果一行放不下这两个元素，那么A元素会被挤到下一行)；如果A元素上一个元素是标准流中的元素，那么A的相对垂直位置不会改变，也就是说A的顶部总是和上一个元素的底部对齐。**

**div的顺序是HTML代码中div的顺序决定的。**

**靠近页面边缘的一端是前，远离页面边缘的一端是后。**





如果把div2、div3、div4都设置成右浮动，则这三个元素的顺序是倒过来的。即div2在最右边。

Float不会覆盖原来的元素，但是会把原来的元素挤开放在其后面显示。

清除浮动的是clear元素.

清除浮动，要想阻止行框围绕浮动框，需要对该框应用 clear 属性。clear 属性的值可以是 left、right、both 或 none，它表示框的哪些边不应该挨着浮动框。

语法：

       clear : none | left | right | both

       取值：

       none  :  默认值。允许两边都可以有浮动对象

       left   :  不允许左边有浮动对象 ，前提是它的左边有左浮动的框。

       right  :  不允许右边有浮动对象 ，前提是它的左边有右浮动的框。

       both  :  不允许有浮动对象

**对于CSS的清除浮动(clear)，一定要牢记：这个规则只能影响使用清除的元素本身，不能影响其他元素。**

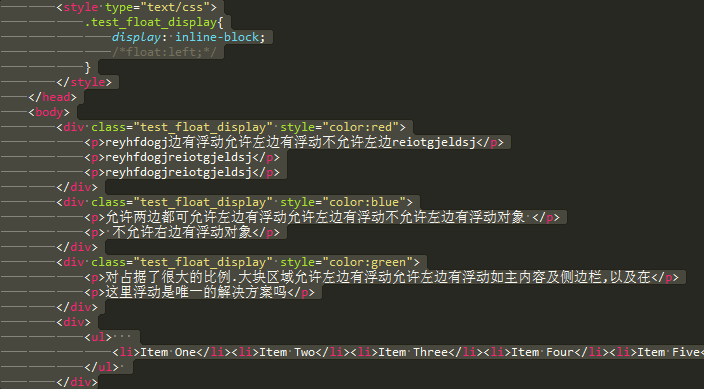
清除浮动是一个时常会遇到的问题，不少人的解决办法是添加一个空的 div 应用 clear:both; 属性。现在，无需增加没有意义的元素，仅需要以下样式即可在元素尾部自动清除浮动：

.clear-fix { \*overflow: hidden; \*zoom: 1; }

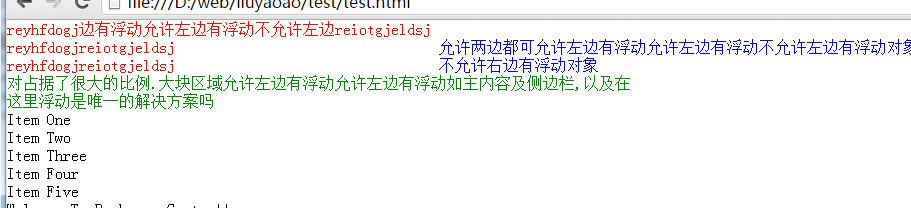
.clear-fix:after { display: table; content: ""; width: 0; clear: both; }

其实Zoom属性是IE浏览器的专有属性，Firefox等浏览器不支撑。它可以设置或检索对象的缩放比例。除此之外，它还有其他一些小作用，比如触发ie的hasLayout属性，清除浮动、清除margin的重叠等。overflow:hidden;和\_zoom:1;是连起来用的，作用是清除border内部浮动。

5、inline-block和float的区别？



下面是使用inline-block时的显示：这里还有其他显示：如果绿色字很短的话会直接排在蓝色字后面。因为这三个div层其实是一行的。



总结：1、三种颜色的div都设置了inline-block。这就把他们都改成了行内块元素。如果其中一块比较高，默认是底部对齐的，如上图的红色字体块和蓝色字体块。可以使用vertical-align属性来改变对齐方式。

2、使用inline-block布局的元素间默认是有间距的。如图红色和蓝色之间有间距。产生“空白间隙”的问题。其实这个问题是我们写代码的习惯所造成的。平时我们写代码，为了使代码看上去“层级分明”，通常会在标签结束符后顺手打个回车， 而回车会产生回车符，回车符相当于空白符，通常情况下，多个连续的空白符会合并成一个空白符，而产生“空白间隙”的真正原因就是这个让我们并不怎么注意的 空白符。

既然我们已经知道产生问题的原因了，那么就很好解决该问题。空白符也是字符，只要是字符就会联想到字体，字体大小之类的。所以去除空白符的存在只需要设置字体大小为零就行了（font-size:0;）。至此还有部分浏览器有问题（safari），再增加样式：letter-spacing: -8px;/\*根据不同字体字号或许需要做一定的调整\*/  
word-spacing: -8px;

**inline-block元素的4px空白间距解决方案：**

<http://www.jb51.net/css/68785.html>

3、还可以增加两个样式，因为ie6/ie7默认是不兼容块级元素的display:inline-block的：

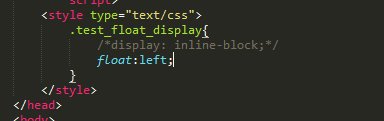
\*display: inline;

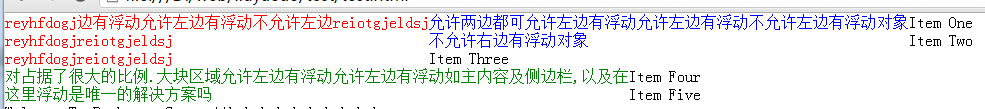
\*zoom: 1; /\*这是为了兼容IE7和6低版本不会显示inline-block的问题。\*/

那么，ie6/ie7下块元素如何实现display:inline-block的效果？解决方法如下：  
让标准浏览器识别display:inline-block;让ie6/7识别 display:inline;来覆盖上面的display:inline-block;然后通过zoom:1;来触发haslayout让inline 元素在ie中表现得和inline-block元素一样。  
.list1 li{display:inline-block; width:150px; **\*zoom:1;\*display:inline;**}  
加上**\*zoom:1**(触发ie6和ie7下的haslayout)**;\*display:inline**(只有ie6和ie7识别)**;  
注：一定要加在display:inline-block;后面。**

=======================

使用float:left布局时的显示：





总结：

1. 使用float浮动时的布局显示有时候会有很多种（根据浮动元素的宽高的不同。）这里有个很明显的特点。（1）、浮动元素本身会顺序排列；（2）、浮动元素会覆盖在未浮动元素之上。但是浮动元素的内容会占据未浮动元素原来的位置，就是说会把未浮动元素原来位置上的内容挤开。
2. 两个浮动元素间是没有空白间隙的。

6、CSS的伪类和伪元素：

**如a:link,a:hover,a:visited,a:active，分别为用于链接尚未被用户访问的状态，用于用户移动他们的鼠标在元素上，而尚未触发或点击它的时候，用于用户访问过的链接，用于用户点击元素的情况。**

**:link**

伪类将应用于未被访问过的链接，与:visited互斥。只能用于a标签。

**:visited**

伪类将应用于已经被访问过的链接，与:link互斥。只能用于a标签。

**:hover**

伪类将应用于有鼠标指针悬停于其上的元素。 所有标签都可。

**:active**

伪类将应用于被激活的元素，如被点击的链接、被按下的按钮等。所有标签都可。

设置时一定要按照上面的顺序排列。

**:focus**

伪类将应用于拥有键盘输入焦点的元素。一般只应用于输入框。

**:first-child**

伪类将应用于元素在页面中第一次出现的时候。

**:lang**

伪类将应用于元素带有指定lang的情况。要在相关的标签上定义了lang属性才行。

# 伪元素

与伪类针对特殊状态的元素不同的是，伪元素是对元素中的特定内容进行操作，它所操作的层次比伪类更深了一层，也因此它的动态性比伪类要低得多。实际上，设计伪元素的目的就是去选取诸如元素内容第一个字（母）、第一行，选取某些内容前面或后面这种普通的选择器无法完成的工作。它控制的内容实际上和元素是相同的，但是它本身只是基于元素的抽象，并不存在于文档中，所以叫伪元素。

**:first-letter**

伪元素的样式将应用于元素文本的第一个字（母）。

**:first-line**

伪元素的样式将应用于元素文本的第一行。

**:before**

在元素内容的最前面添加新内容。

**:after**

在元素内容的最后面添加新内容。

不过，在 CSS3 中为了区别伪元素和伪类为伪元素使用了双冒号，因此如果使用了 display 或者 width 等属性时使得显示脱离了原本元素后，建议按照标准双写。过于老的浏览器可能会存在支持问题，不过伪元素大多是配合 CSS3 使用，就无所谓向下兼容了：

img::after {}

before和after的一些应用举例：

这两个伪类下特有的属性 content ，用于在 CSS 渲染中向元素逻辑上的头部或尾部添加内容。注意这些添加不会改变文档内容，不会出现在 DOM 中，不可复制，仅仅是在 CSS 渲染层加入。比较有用的是以下几个值：

* **[String]** – 使用引号包括一段字符串，将会向元素内容中添加字符串。示例：

a:after { content: "↗"; }

* **attr()** – 调用当前元素的属性，可以方便的比如将图片的 Alt 提示文字或者链接的 Href 地址显示出来。示例：

a:after { content:"(" attr(href) ")"; }

* **url() / uri()** – 用于引用媒体文件。示例：

h1::before { content: url(logo.png); }

* **counter()** –  调用计数器，可以不使用列表元素实现序号功能。具体请参见 counter-increment 和 counter-reset 属性的用法。示例：

h2:before { counter-increment: chapter; content: "Chapter " counter(chapter) ".

样式示例:

a {

position: relative;

display: inline-block;

outline: none;

text-decoration: none;

color: #000;

font-size: 32px;

padding: 0 10px;

}

/\* 大框 \*/

a:hover::before, a:hover::after {

content: "";

display: block;

position: absolute;

top: -15%%;

left: -14%%;

width: 120%;

height: 120%;

border-style: solid;

border-width: 4px;

border-color: #DDD;

}

/\* 小框 \*/

a:hover::after {

top: 0%;

left: 0%;

width: 100%;

height: 100%;

border-width: 2px;

}

可以看到在使用before和after添加内容时有几个属性是必须添加的：

 float: left; //left或right；

 position: relative; //或者是absolute。

 top: 30px; //也可以是left: 40px;等。

或者是如上面一样：

display: block;

position: absolute;

top: -15%%; //这里为什么要两个百分号？？

left: -14%%;