Css实现倒三角图标

网页中一些小图标可以用css实现，而不是切图。

分别试了几种效果

.triangle{

width: 0px;

height: 0px;

border-left: 150px solid red;

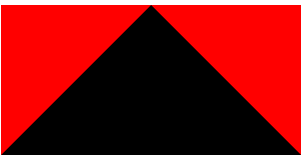
border-right: 150px solid red;

border-bottom: 50px solid black;

}



将border-bottom改为 100px solid black;



.triangle{

width: 0px;

height: 0px;

border-top: 50px solid black;

border-left: 150px solid red;

border-right: 150px solid red;

}



.triangle{

width: 0px;

height: 0px;

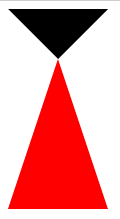
border-top: 50px solid black;

border-left: 150px solid red;

}



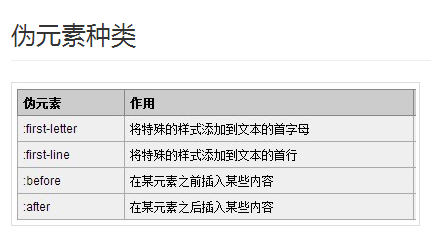
如果只写border-top: 50px solid black;border-bottom: 150px solid red; 网页上不能显示，将border-left和border-right的边框颜色设置为transparent就可以显示了，我分别设置成了50px。



综上，我对css实现倒三角原理的理解是，因为宽高为0，所以中间没有元素，如果设置了边框，就会把所有边框挤在一起，最后的形状也会因为边框大小不同而不同。

关于伪类和伪元素





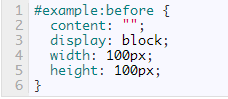
区别：伪类需要在样式中定义一个样式，伪元素只需要在样式里定义一个元素即可。

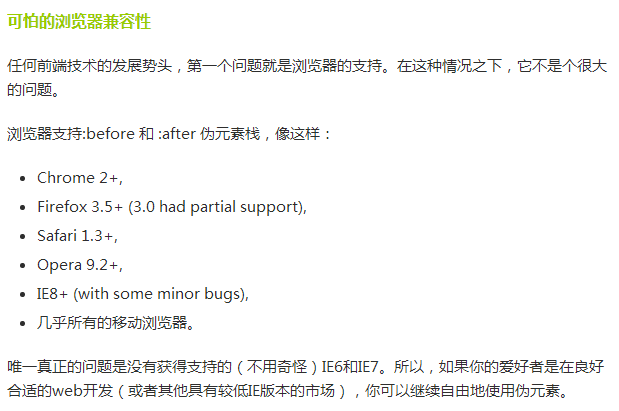
为什么要使用伪类？

利用伪元素装饰内容 (无论是装饰图片还是音效) 不需要更改 HTML 的内容，从而帮助内容与样式更好地分离。因为如果仅仅为了画一个装饰用的三角就在 HTML 里多加一个元素，这上对于实际内容来说其实是多余的，对自动分析网页的语义也可能会产生不好的影响 (这取决于分析程序的具体实现)。   
应该说，::before 和 ::after 伪元素的初衷还是用于插入内容——不受文档约束，也不影响文档本身（比如，不影响 DOM），只影响最终的样式。  
CSS 规范中给的例子很典型：p.note:before { content: "Note: " } ——这种插入的内容本身其实不是真正的内容，其实是身为文本的样式，所以没必要在 HTML 中重复出现，交给 CSS 来生成会很不错。

使用伪类必须与content属性结合，伪元素如果没有设置“content”属性，伪元素是无用的。

你可以设置content属性值为空，并且仅仅把他当做一个内容很少的盒子。像这样:





伪元素不是决定性的

幸运的是，缺少伪元素不会造成大问题。大多数情况下，伪元素一般修饰（或者帮助）内容，不会给不支持的浏览器造成问题。所以，如果你的支持者具有较高的IE版本，你仍然可以在某种程度上使用它们。

一些提醒

正如前面提到的，伪元素不会出现在DOM中。这些元素不是真正的元素。因此，它们[不是可用的](http://cssgallery.info/testing-the-accessibility-of-the-css-generated-content" \t "http://blog.jobbole.com/49301/_blank)。所以，不要使用伪元素生成内容，是您的网页的可用性和可访问性的关键。

另外一件需要记住的是，开发工具，例如火狐，[不要用伪元素显示内容](http://meyerweb.com/eric/thoughts/2009/11/03/pseudo-phantoms/" \t "http://blog.jobbole.com/49301/_blank)。所以，如果使用了，伪元素会造成难以维护和调试缓慢。

（更新：在[评论](http://coding.smashingmagazine.com/2011/07/13/learning-to-use-the-before-and-after-pseudo-elements-in-css/" \l "comment-544285" \t "http://blog.jobbole.com/49301/_blank)中提到的，你可以使用谷歌的开发工具来查看一个伪元素相关联的风格，但不会出现在DOM元素里。同时，[火狐在1.8版](http://getfirebug.com/wiki/index.php/Firebug_Release_Notes" \l "CSS" \t "http://blog.jobbole.com/49301/_blank)加入伪元素支持它。）

**用到的伪类：设置最后一个footer-col的属性**

.public-footer .footer-col:last-child{

margin-right: 0px;

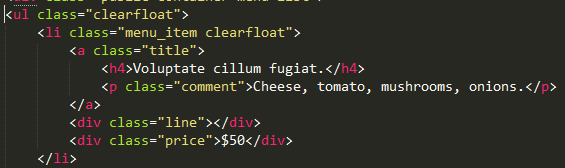
}

习惯在列表ul-li前面套一个div，设置一些属性。



在块级元素中，不建议把块级元素和内联元素写在同一级。

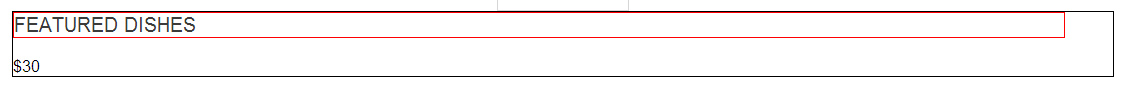
在a标签前面套一层div，a标签内容会不显示。

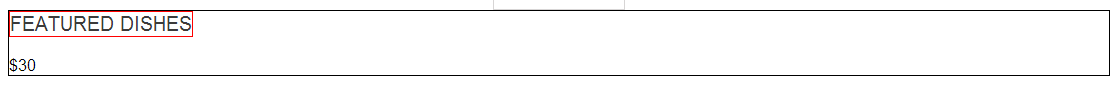


一般设置div中字体的垂直居中用设置padding的方式。

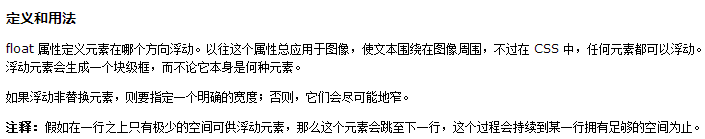
在设置div里h3标签的样式时发现一个问题，给h3标签设置float：left与不设置，h3宽度不同。因为本身位置就是靠左的，所以一开始没有设置浮动，但是发现后面的div位置不对。

设置浮动前后：





去W3School查了float属性：



注意：它们会尽可能的窄。如果宽度不变的话，还是设置float比较好。

正好碰到了float的其他问题

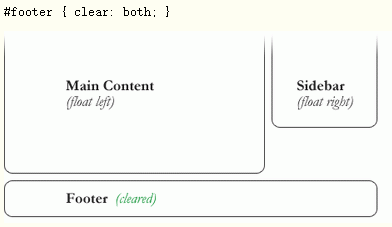
清除浮动：清除(clear)是浮动(float)的相关属性.一个设置了清除Float的元素不会如浮动所设置的一样，向上移动到Float元素的边界，而是会忽视浮动向下移动。

清除浮动前：



上图中，侧栏向右浮动，并且短于主内容区域。页脚(footer)于是按浮动所要求的向上跳到了可能的空间。要解决这个问题，可以在页脚(footer)上清除浮动，以使页脚(footer)待在浮动元素的下面。

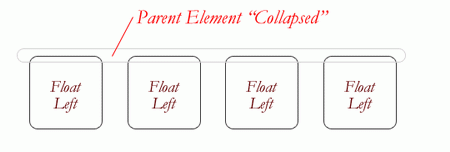
清除浮动后：



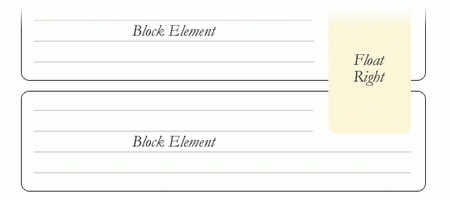
清除(clear)也有4个可能值。最常用的是 both，清楚左右两边的浮动。left 和 right 只能清除一个方向的浮动。none 是默认值，只在需要移除已指定的清除值时用到。inherit 应该时第五个值，不过很奇怪的是 IE 不支持。

伟大的塌陷

使用浮动(float)的一个比较疑惑的事情是他们怎么影响包含他们的父元素的。如果父元素只包含浮动元素，那么它的高度就会塌缩为零。如果父元素不包含任何的可见背景，这个问题会很难被注意到，但是这是一个很重要的问题。



塌陷的直观对立面更不好，看看下面的情况：



当上面的块级元素自动扩展以适应浮动元素时，段落间的文本流中会出现非自然的空白换行，而且没有有效的方法来修正这个问题。对于这种情况，设计师的抱怨会更甚于对塌陷的抱怨（没理解，不是设计完成之后才会进行页面编码吗？－ 糖伴西红柿）。

为了防止怪异的布局和跨浏览器的问题，塌陷问题几乎总是被要处理的。我们在容器中的浮动元素之后，容器结束之前来清除浮动。

清除浮动的技术

如果你很明确的知道接下来的元素会是什么，可以使用 clear:both; 来清除浮动。这个方法很不错，它不需要 hack，不添加额外的元素也使得它有良好的语义性。当然事情并不是都可以这样解决的，工具箱中还是需要另外几个清除浮动的工具。

空div方法：从字面来看，是一个空的 div。  
有时可能会用一些其他元素，但是 div 是最常用的，因为它没有浏览器默认样式；没有特殊功能，而且一般不会被 css 样式化。

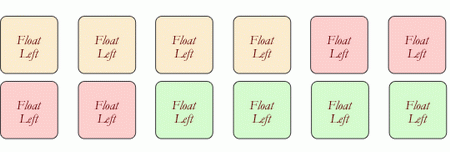
overflow 方法：在父元素上设置 overflow 这个 css 属性。如果父元素的这个属性设置为 auto 或者 hidden，父元素就会扩展以包含浮动。这个方法有着较好的语义性，因为他不需要额外元素。但是，如果需要增加一个新的 div 来使用这个方法，其实就和空 div 方法一样没有语义了。而且要记住，overflow 属性不是为了清除浮动而定义的。要小心不要覆盖住内容或者触发了不需要的滚动条。

简单清除方法（常用）：使用了一个聪明的 css 伪选择符(:after)来清除浮动。比起在父元素上设置 overflow，只需要给它增加一个额外的类似于”clearfix”的类。这个类使用如下 css:

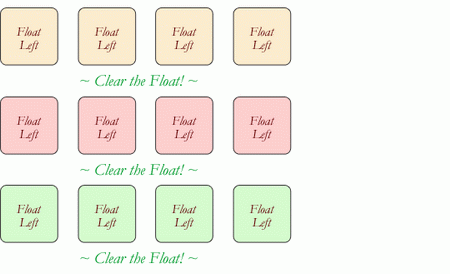
.clearfix:after { content: "."; visibility: hidden; display: block; height: 0; clear: both; }

这会在清除浮动的父元素之后应用一点看不见的内容。这不是[全部内容](http://www.positioniseverything.net/easyclearing.html)，还需要一些额外的代码来适应那些老旧的浏览器。

不同的情况需要不同的浮动清除方法。以一个具有不同样式块的网格为例。



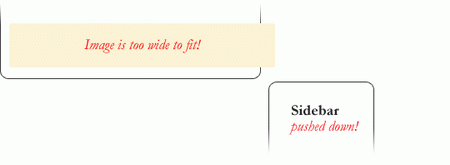
为了从视觉上较好的把相似的块联系起来，需要在必要的地方开启新行，这里是颜色改变的地方。如果每个颜色组都有一个父元素的话，我们可以使用 overflow 或者 简单清除方法。或者，在每组之间用一个空div方法。额外的 div 之前并不存在，可以自己试试来看看哪个方法好。



浮动的问题

浮动因脆弱而饱受诟病。大多数的脆弱性来自于 IE6 及其一系列的浮动相关 bug。因为越来越多的设计师不再支持 IE6 了，你也可以不关注它了。不过对于那些要关注的人来说，这里有些大概。

推倒是浮动元素内的元素（大多是图片）比浮动元素本身宽造成的现象。大多数的浏览器会在浮动之外渲染图片，但是不会有伸出来的部分影响其他布局。IE 会扩展浮动来包含图片，精彩大幅度地影响布局。一个普遍的例子是突破伸出主内容之外把侧栏推到下面。



快速修正：确保不是图片造成这种情况，使用 overflow:hidden 来切除多余的部分。

双倍边距bug处理 IE6 时，另一个需要记住的事情是，如果在和浮动方向相同的方向上设置外边距(margin)，会引发[双倍边距](http://www.cssnewbie.com/double-margin-float-bug/)。快速修正：给浮动设置 display:inline; 而且不用担心，它依然是块级元素。

3像素间距是指挨着浮动元素的文本会神奇的被踢出去3像素，好像浮动元素的周围有一个奇怪的力场一样。快速修正：在受影响的文本上设置宽度或高度。

IE7 中，底边距 bug是当浮动父元素有浮动子元素时，这些子元素的底边距会被父元素忽略掉。快速修正：用父元素的底内补白(padding)代替。

（http://www.cnblogs.com/hnyei/archive/2011/09/27/2193133.html）

还有一个问题，当元素设置了浮动但是后面的元素没有设置浮动时，后面的元素会占据前面元素的位置。float使元素脱离了文档流，其他盒子会无视这个元素，但其他盒子内的文本依然会为这个元素让出位置，环绕在周围。

这是因为当初float被设计的时候就是用来完成文本环绕的效果，所以文本不会被挡住，这是float的特性。

而对于使用position：absolute 脱离文档流的元素，其他盒子与其他盒子内的文本都会无视它。

圆形小图标可以用border-radius圆形边框实现。

display：inline和inline-block都会换行，但是区别是，后者可以改高宽，前者由图片本身决定，如果设置了宽高会失效。

实现

.index-panel-header .btn-group{

float: right;

font-size: 0;

}

.index-panel-header .btn{

display: inline-block;

width: 10px;

height: 10px;

margin-left: 11px;/\*你会发现实际的间距是15px，想要去掉元素与元素之间的间距，用font-size: 0实现。\*/

background: #dedede;

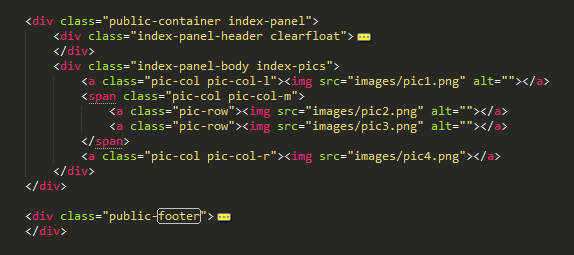
border-radius: 50%;

}

margin-top失效最笨的方法，改成内边距padding-top。

形状是不能直接切图的，必须先转换成智能对象。

注意划虚下划线的span，一开始用的是div，内部几个元素的宽度加起来等于总div的宽度，但就是把最后一个元素挤下去了，超出了2px，怎么都不知道为什么，换成了span之后就没有问题了。果然div和a标签放在一起就有毒。



还有一个问题，按照老师的写法，中间上面的图片不能覆盖下面的图片的上部分，我给中间下面的图片设置了absolute和z-index：-1，搞定。

