# 字体大小px、em、rem

这里的字体大小指的是用户自己设置或者设备的默认字体大小。同样是12px大小的字符在不同设备上对用户的观感是不一样的，但是设备总会提供一个观感还不错的默认字体大小（或者用户自己指定的）。所以为了让我们的网站的文字在不同设备上也能有不错的观感，网站应该使用em代替px。em定义的是一个相对大小，设计者通过它可以定义哪些地方的字应该大一些，哪些地方的字应该小一些，但是不规定哪些地方的字必须多大。

任意浏览器的默认字体高都是16px

rem始终是基于根元素（html） 的。

比如默认的 html font-size=16px，那么我想设置12px 的文字就是：12÷16=0.75（rem）

PX特点

1. IE无法调整那些使用px作为单位的字体大小；

2. 国外的大部分网站能够调整的原因在于其使用了em或rem作为字体单位；

3. Firefox能够调整px和em，rem，但是96%以上的中国网民使用IE浏览器(或内核)。

px像素（Pixel）。相对长度单位。像素px是相对于显示器屏幕分辨率而言的。(引自CSS2.0手册)

em是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。(引自CSS2.0手册)

 EM特点

1. em的值并不是固定的；

2. em会继承父级元素的字体大小。

如示例所示：

1、

<div style="font-size: 20px;">

<div style="width:50px;height:50px;border: 1em solid red;font-size:12px;"></div>

</div>

border宽度为12px

2、

<div style="font-size: 20px;">

<div style="width:50px;height:50px;border: 1em solid red;"></div>

</div>

Border宽度为20px

       任意浏览器的默认字体高都是16px。所有未经调整的浏览器都符合: 1em=16px。那么12px=0.75em,10px=0.625em。为了简化font-size的换算，需要在css中的body选择器中声明Font-size=62.5%，这就使em值变为 16px\*62.5%=10px, 这样12px=1.2em, 10px=1em, 也就是说只需要将你的原来的px数值除以10，然后换上em作为单位就行了。

# Chrome下font-size最小12px

Chrome浏览器限制了字体的最小值为12px！

因此，在使用chrome模拟手机测试时，字体最小为12px。

但是来往手机展示时，因为是在webView（webkit内核渲染引擎）中展示的，所以没有字体的最小限制！

因此，应该将页面发布，然后直接在手机上测试字体的显示！

**另外附上解决chrome浏览器最小字号的限制的方法：**

使用-webkit-transform: scale(xxx)来缩小字体（字体所占位置不变）（只对块级元素有用）

经查询以前大家经常用-webkit-font-size-adjust:none;新版浏览器已不再支持该属性；

# line-height 因子，设置行高

如果line-height属性值有单位，那么继承的值则是换算后的一个具体的px级别的值；而如果属性值没有单位，则浏览器会直接继承这个“因子（数值）”，而非计算后的具体值，此时它的line-height会根据本身的font-size值重新计算得到新的line-height 值。

深入学习：[css行高line-height的一些深入理解及应用](http://www.zhangxinxu.com/wordpress/2009/11/css%E8%A1%8C%E9%AB%98line-height%E7%9A%84%E4%B8%80%E4%BA%9B%E6%B7%B1%E5%85%A5%E7%90%86%E8%A7%A3%E5%8F%8A%E5%BA%94%E7%94%A8/)

# Fixed水平居中

position: fixed;

left: 0;

right: 0;

margin-left: auto;

margin-right: auto;

注意！fixed布局还需要设置top或bottom，否则不会显示出来！

# 垂直居中

使用display:table-cell，使得子元素垂直居中：

外层嵌套一个div，设置为display:table

单行文字垂直居中

先要知道css里vertical-align无效，W3C官方对vertical-align做了下面的解释： “ This property affects the vertical positioning inside a line box of the boxes generated by an inline-level element.”   
      
     实际上，一个Box中由很多行很多元素组成，vertical-align只作用于在同一行内的元素，它的垂直并不是相对于整个Box而言的。前面定义了一个60px的高度，但是这个Box中存在很多行，那段文本并不能对齐到中央。因此希望那段文本对齐中间，需要给它定义一个line-height的属性，让line-height为60px，作用于一行的vertical-align就可以工作了。

* + 为什么将height和line-height设置为相同百分比，不能实现垂直居中？

这是因为height的百分比是“基于包含它的块级对象的百分比高度”，而line-height的百分比是“其百分比取值是基于字体的高度尺寸”的，两者不相同！

# 如何使用css切精灵图片区域

background-size

设置background-size:width（数值） height（数值）时，设置的是背景图片的大小。

　1、Background-size的语法

[css]

background-size: auto(原始图片大小) || number(数值) || percentage(百分比) || cover(放大铺满) || contain(缩小铺满)

　　2、Background-size的属性值

　　1、auto：此值为默认值，保持背景图片的原始高度和宽度;

　　2、number：此值设置具体的值，可以改变背景图片的大小;

　　3、percentage：此值为百分值，可以是0%～100%之间任何值，但此值只能应用在块元素上，所设置百分值将使用背景图片大小根据所在元素的宽度的百分比来计算。

　　4、cover：此值是将图片放大，以适合铺满整个容器，这个主要运用在，当图片小于容器时，又无法使用background-repeat来实现时，我们就可以采用cover;将背景图片放大到适合容器的大小，但这种方法会使用背景图片失真;

　　5、contain：此值刚好与cover相反，其主要是将背景图片缩小，以适合铺满整个容器，这个主要运用在，当背景图片大于元素容器时，而又需要将背景图片全部显示出来，此时我们就可以使用contain将图片缩小到适合容器大小为止，这种方法同样会使用图片失真。

　　当background-size取值为number和percentage时可以设置两个值，也可以设置一个值，当只取一个值时，第二个值相当于auto，但这里的auto并不会使背景图片的高度保持自己原始高度，而会与第一个值相同。

#### 问题描述

现在有个精灵图片，如下图所示：



（注：为了方便说明，以下数据为伪造数据）

现在我想在页面上以40px\*30px的大小显示小人图片：



精灵图片大小spriteSize为1280px\*500px

该小人图片在精灵图片中的位置为：

imgOffset：640px\*160px

imgSize：80px\*60px

#### 解决方案

使用background-size、background-position、width、height来从精灵图片中切出图片，将该图片设为背景图片。

思想：

首先计算显示大小与要切出的图片在精灵图片中大小的比例，按照比例设置background-size，对精灵图片大小进行缩放

然后使用backgournd-position（位移offset也要按比例缩放）、width和height（显示区域的大小，也是从精灵图片中切出的size大小。因为已经使用background-size对精灵图片大小进行了缩放，使得显示区域大小正好等于切出的size大小！）切出图片。

具体到本示例中，

宽度比例wRatio=40 / 80 = 0.5

高度比例hRatio= 30 / 60 = 0.5

背景图片宽度wBackgroundSize = wSpriteSize \* wRatio = 720

背景图片高度hBackgroundSize =hSpriteSize \* hRatio = 250

背景位移宽度wBackgroundPos = wImgOffset \* wRatio = 320

背景位移高度hBackgroundPos = hImgOffset \* wRatio = 80

所以，最终css为：

.img {

background-image: url(sprite.png);

background-size: 720px 250px;

background-position: 320px 80px;

width: 40px;

height: 30px;

}

#### 总结

**已知：**

显示图片大小为width、height

精灵图片大小为wSpriteSize、hSpriteSize

目标图片在精灵图片中的位置wImgOffset、hImgOffset

目标图片在精灵图片中的区域大小wImgSize、hImgSize

求解：

背景图片大小wBackgroundSize、hBackgroundSize

背景位移wBackgroundPos、hBackgroundPos

**解**：

/ wSpriteSize = width / wImgSize

/ hSpriteSize = height / hImgSize

= wSpriteSize \* (width / wImgSize)

= hSpriteSize \* (height / hImgSize)

= width / wImgSize

= height / hImgSize

= \* (width / wImgSize)

= \* (height / hImgSize)

**所以：**

wRatio = width / wImgSize

hRatio = height / hImgSize

= wSpriteSize \* wRatio

= hSpriteSize \* hRatio

= \* wRatio

= \* hRatio

img元素下方出现几像素的空白间隙  
问题描述

如：

<div style="">

<img src=”…” style="

max-width: 200px;

max-height: 350px;

"/>

</div>

Img下方会出现几个px的空白

## 原因分析

不清楚

## 解决方案

这个也有多种解决方案，如将img定义为display:block,或定义父容器为font-size:0，个人更推荐使用vertical-align的方式，它的值可以是text-top | text-bottom | middle等

或者设置div的width和height，然后设置img为width:100%;height:100%

# 浏览器元素默认样式

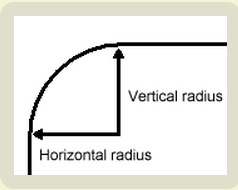
参考[h​t​m​l​默​认​样​式​和​浏​览​器​默​认​样​式](http://wenku.baidu.com/view/ef98bbfe910ef12d2af9e71e)

# css3

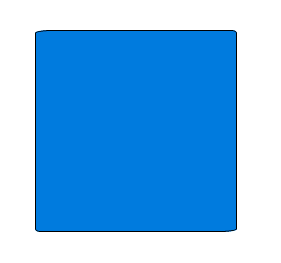
## border-radius研究

#### 基本用法

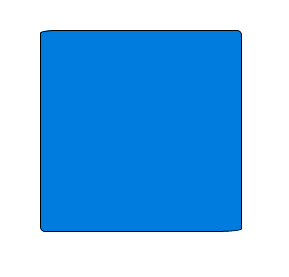
border-radius还可以用斜杠设置第二组值。这时，第一组值表示水平半径，第二组值表示垂直半径。第二组值也可以同时设置1到4个值，应用规则与第一组值相同。



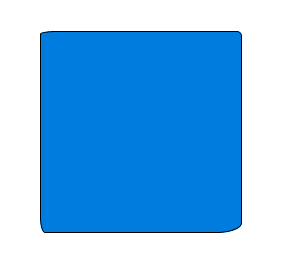
　　border-radius: 15px 5px / 3px;



　　border-radius: 15px 5px 25px / 3px 5px;



　　border-radius: 15px 5px 25px 5px / 3px 5px 10px 15px;



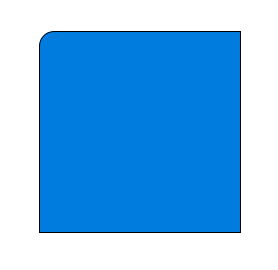
**三、单个圆角的设置**

除了同时设置四个圆角以外，还可以单独对每个角进行设置。对应四个角，CSS3提供四个单独的属性：

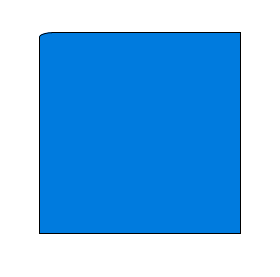
　　\* border-top-left-radius  
　　\* border-top-right-radius  
　　\* border-bottom-right-radius  
　　\* border-bottom-left-radius

这四个属性都可以同时设置1到2个值。如果设置1个值，表示水平半径与垂直半径相等。如果设置2个值，第一个值表示水平半径，第二个值表示垂直半径。

　　border-top-left-radius: 15px;



　　border-top-left-radius: 15px 5px;



#### 内外半径

border-radius的内径值是等于外径值减去边框厚度值,例如下面的实例

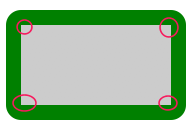
.border-big {

border: 15px solid green;

border-radius: 15px;

}

效果：



外半径=15px

内半径=border\_radius – border\_width = 15 – 15 = 0px

因此内部没有圆角效果

我们接着上面这个例子，把 border-radius半径值改成比边框值大一点：

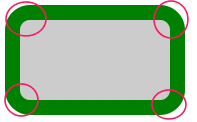
.border-small {

border: 15px solid green;

border-radius: 25px;

}

效果：



为何当border-radius的半径小于元素边框的厚度时，内部没有圆角效果？我在这里说一下，因为我们的border-radius的内径值是等于外径值减去边框厚度值，当他们的值为负时，内径默认为0,最前面讲border-radius取值时就说过其值不能为负值。同时也说明border-radius的内外曲线的圆心并不一定是一致的。只有当边框厚度为0时，我们内外曲线的圆心才会在同一位置。

#### 实例

##### 画圆

Width和height相等，border-radius>=width/2即可

## Box-sizing

**content-box**

默认值，标准盒模型。 [width](https://developer.mozilla.org/zh-CN/docs/CSS/width) 与 [height](https://developer.mozilla.org/zh-CN/docs/CSS/height) 是内容区的宽与高， 不包括边框，内边距，外边距。

**padding-box**

[width](https://developer.mozilla.org/zh-CN/docs/CSS/width) 与 [height](https://developer.mozilla.org/zh-CN/docs/CSS/height) 包括内边距，不包括边框与外边距。

**border-box**

[width](https://developer.mozilla.org/zh-CN/docs/CSS/width) 与 [height](https://developer.mozilla.org/zh-CN/docs/CSS/height) 包括内边距与边框，不包括外边距。这是IE 怪异模式（Quirks mode）使用的 [盒模型](https://developer.mozilla.org/en-US/docs/CSS/Box_model) 。

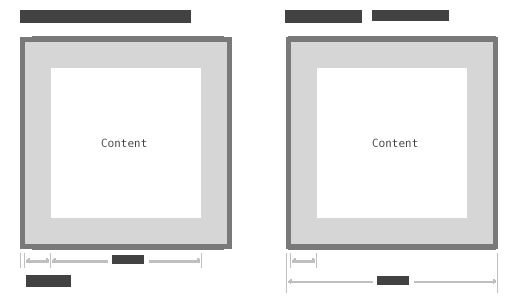
说到 IE 的 bug，一个臭名昭著的例子是它对于“盒模型”的错误解释：在 IE5.x 以及 Quirks 模式的 IE6/7 中，将 border 与 padding 都包含在 width 之内。这为前端工程师的工作平添了不少麻烦，几户每个需要定义尺寸的 box 都要思量一下：是否触发了“盒模型 bug”？

同时，由于另一撮浏览器对标准的遵从，我们在精确定义一个在有限空间内显示的 box 时，也需要计算一下：留给它的空间只有那么大，刨去 border 和 padding，我们该把它的 width 写成多少呢？

这种情况在 CSS3 时代有了改善，得益于这个叫做 box-sizing 的属性，它具有“content-box”和“border-box”两个值。

定义 box-sizing: content-box; 时，浏览器对盒模型的解释遵从我们之前认识到的 W3C 标准；

定义 box-sizing: border-box; 时，浏览器对盒模型的解释与 IE6 相同；



## 字体

### -webkit-font-smoothing

-webkit-font-smoothing: antialiased;

-webkit-text-stroke-width: 0.2px;

-moz-osx-font-smoothing: grayscale;

**webkit内核的私有属性： -webkit-font-smoothing**

          --------------------   用于**字体抗锯齿,**使用后字体看起来会更**清晰舒服**

它有3个**可选**的值：

* **none**  
  **------ 对低像素的文本比较好**
* **subpixel-antialiased             
  ------默认值**
* **antialiased                           
   ------抗锯齿很好**

### **text-stroke**

**复合属性。设置或检索对象中的文字的描边。**

参考<http://www.w3chtml.com/css3/properties/text/text-stroke.html>

### font- family

可以指定多个字体，以逗号隔开。按优先顺序排列

**font-family**：[ [<family-name>](http://css.doyoe.com/properties/font/font-family.htm#dfn-syntax) | [<generic-family>](http://css.doyoe.com/properties/font/font-family.htm#dfn-syntax2) ] [, [<family-name>](http://css.doyoe.com/properties/font/font-family.htm#dfn-syntax) | [<generic-family>](http://css.doyoe.com/properties/font/font-family.htm#dfn-syntax2)]\*

**<family-name>** = arial | georgia | verdana | helvetica | simsun and etc.

**<generic-family>** = cursive | fantasy | monospace | serif | sans-serif

**默认值**：由user agent决定

**适用于**：所有元素

**继承性**：有

**动画性**：否

**计算值**：指定值

取值：

**<family-name>：**

字体名称。按优先顺序排列。以逗号隔开。如果字体名称包含空格或中文，则应使用引号括起

**<generic-family>：**

字体序列名称。

说明：

**设置或检索用于对象中文本的字体名称序列。**

* 序列可包含嵌入字体。请参阅[@font-face](http://css.doyoe.com/rules/@font-face.htm)规则。
* 一般字体引用可以不加引号，如果字体名包含了空格、数字或者符号（如连接符）则需加上引号，避免引发错误。
* user agent会遍历定义的字体序列，直到匹配到某个字体为止。
* **指定字体序列：**

**写法：**

body{font-family:helvetica,verdana,sans-serif;}

## -webkit-touch-callout

Description

Disables the default callout（标注） shown when you touch and hold a touch target.

On iPhone OS, when you touch and hold a touch target such as a link, Safari displays a callout containing information about the link. This property allows you to disable that callout.

Syntax

-webkit-touch-callout: <behavior>;

Values

<behavior>

The touch callout behavior.

|  |  |
| --- | --- |
| default | Restore the default callout. |
| none | The callout is disabled. |
| inherit |  |

## -webkit-user-select:none

禁止选择文本

或许你常常不希望用户在你的网站上选择文本，无论是否是出于版权的原因。通常大家会有js来实现，另一个方案就是，将-webkit-user-select 和-moz-user-select 的值设为none。

## -webkit-tap-highlight-color

这个属性只用于iOS (iPhone和iPad)。当你点击一个链接或者通过Javascript定义的可点击元素的时候，它就会出现一个半透明的灰色背景。要重设这个表现，你可以设置-webkit-tap-highlight-color为任何颜色。

想要禁用这个高亮，设置颜色的alpha值为0即可。

**示例**

设置高亮色为50%透明的红色:

## backface-visibility

backface-visibility 属性定义当元素transform变换不面向屏幕时是否可见。

实例

隐藏被旋转的 div 元素的背面：

div

{

backface-visibility:hidden;

-webkit-backface-visibility:hidden; /\* Chrome 和 Safari \*/

-moz-backface-visibility:hidden; /\* Firefox \*/

-ms-backface-visibility:hidden; /\* Internet Explorer \*/

}

[亲自试一试](http://www.w3school.com.cn/tiy/t.asp?f=css3_backface-visibility)

浏览器支持

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IE | Firefox | Chrome | Safari | Opera |
|  |  |  |  |  |

只有 Internet Explorer 10+ 和 Firefox 支持 backface-visibility 属性。

Opera 15+、Safari 和 Chrome 支持替代的 -webkit-backface-visibility 属性。

定义和用法

backface-visibility 属性定义当元素transform变换不面向屏幕时是否可见。

如果在旋转元素不希望看到其背面时，该属性很有用。

|  |  |
| --- | --- |
| **默认值：** | visible |
| **继承性：** | no |
| **版本：** | CSS3 |
| **JavaScript 语法：** | *object*.style.backfaceVisibility="hidden" |

语法

backface-visibility: visible|hidden;

|  |  |
| --- | --- |
| **值** | **描述** |
| visible | 背面是可见的。 |
| hidden | 背面是不可见的。 |

min-height

最小高度

max-height

最大高度

## content

### 一、哗啦哗啦的简介

zxx://这里“哗啦哗啦”的作用是为了渲染一种氛围。  
content属性早在CSS2.1的时候就被引入了，可以使用:before以及:after伪元素生成内容。此特性目前已被大部分的浏览器支持：(Firefox 1.5+, Safari 3.5+, IE 8+, Opera 9.2+, Chrome 0.2+)。另外，目前Opera 9.5+ 和 Safari 4已经支持所有元素的content属性，而不仅仅是:before和:after伪元素。

在[CSS 3 Generated Content工作草案](http://www.w3.org/TR/css3-content/)中，content属性添加了更多的特征，例如：插入以及移除文档内容的能力，以创建脚注，结语，及段落注释。但是目前还没有浏览器支持content的扩展功能。

本文将展示content内容生成的基本使用，以及一些您可以应用的特别的技术。

### 二、还有一点小小的提醒

在进入正题之前，我要做一回啰嗦的老女人，讲点可有可无无伤大雅的提醒。

* 仅在CSS可用的现代浏览器下起作用。
* 不可通过DOM使用，它只是纯粹的表象。在特殊情况下，从一个访问的角度来看，当前屏幕阅读不支持生成的内容。

### 三、最基本的 – 生成内容

content使用如下：

h2:before {

content: "我是额外文字！";

}

此段样式的作用是在每个h2标签的前面添加文字“我是额外文字”。  
您可以狠狠地点击这里：[content最基本使用Demo](http://www.zhangxinxu.com/study/201004/css-content-basice.html)

content属性的值除了使用文本值外，还可以使用标签的一些属性值，方法是attr()，参见如下代码：

a.content:after {

content: attr(href);

}

<a class="content" href="http://www.zhangxinxu.com/">此标签的href值是：</a>

结果如下截图（截自Firefox3.6）：



对应a标签内的文字值的最后添加了此标签的href属性的值。  
您可以狠狠地点击这里：[标签属性值生成content内容Demo](http://www.zhangxinxu.com/study/201004/css-content-basics-attr.html)

**注意：**当使用attr()获取标签属性名的时候，千万不要添加引号!

您还可以使用计数器产生动态数字或运用url(/path/to/file)路径插入图片。要想获得更多信息，请继续看下面的应用实例。

### 四、使用计数器创建号码内容

如果您想插入一系列递增的值，比如说“美女1，美女2，美女3，美女4……”，您就可以使用计数器在每个列表项的前面插入递增的序号值。CSS代码如下：

ol {

list-style-type:none;

counter-reset:sectioncounter;

}

ol li:before {

content:"美女" counter(sectioncounter) "： ";

counter-increment:sectioncounter;

}

HTML代码如下：

<ol>

<li><img src="http://image.zhangxinxu.com/image/study/s/s128/mm1.jpg" width="128" height="96" /></li>

<li><img src="http://image.zhangxinxu.com/image/study/s/s128/mm3.jpg" width="128" height="96" /></li>

</ol>

结果如下图（Firefox3.6下截图）：



您可以狠狠地点击这里：[content产生递增序号demo](http://www.zhangxinxu.com/study/201004/css-content-number.html)

**说明：**  
1. ol标签应用counter-reset属性，li标签应用counter-increment属性，值均为sectioncounter，再配合content属性，以及counter计数器就可以实现递增的效果了。  
2. 如果您指定了元素display:none，则content下的计数器不会产生递增值。  
3. 当然，如果浏览器不支持此方法，那么就不会有数字出现，您使用firebug或是之类的工具查看真实的HTML时不会看到产生的数值，不要有疑惑，content属性动态生成的内容是纯粹的装饰而已，虚假的表象。  
4. 如果您对CSS计数器感兴趣，想深入了解，可以参见David Storey一篇优秀的文章：[Automatic numbering with CSS Counters](http://dev.opera.com/articles/view/automatic-numbering-with-css-counters/)。

### 五、为多语言内容插入正确的引号

不同的语言使用不同的引号字符。比如英文的引号就是""，而中文引号就是“”。使用content可以使得不同的语言使用对应的一些字符。例如，我们需要对下面的些文字添加其对应语言的引号，该怎么办呢？

<p lang="en"><q>It’s only work if somebody makes you do it.</q></p>

<p lang="no"><q>Hvis du forteller meg nok en vits, så skal jeg slå deg til jorden.</q></p>

<p lang="ch"><q>欢迎来到上海，欢迎参观世博会！</q></p>

我们不妨试试下面的代码：

/\* 为不同语言指定引号的表现 \*/

:lang(en) > q {quotes:'"' '"';}

:lang(no) > q {quotes:"«" "»";}

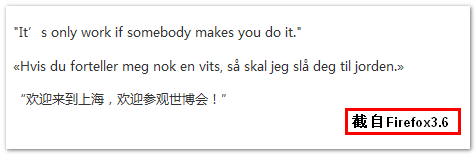
:lang(ch) > q {quotes:"“" "”";}

/\* 在q标签的前后插入引号 \*/

q:before {content:open-quote;}

q:after {content:close-quote;}

结果如下图所示：



您可以狠狠地点击这里：[content生成对应语言字符demo](http://www.zhangxinxu.com/study/201004/css-content-language-quot.html)

此技术可以应用于各类元素，而不仅仅是q标签，另外，Safari 3以及IE7（以及以下）是不支持此技术的。

### 六、用图片替换文字

关于图片替换技术，您可以参见这里：[several image replacement techniques](http://www.mezzoblue.com/tests/revised-image-replacement/)，里面提供了几种方法。而这里展示的则是另外不同的用图片替换方法，使用的是content。

您可以参见下面的代码：

div.logo {

content:url(logo.png);

}

这种图片替换技术的优点在于文字内容确实被替换了。因此，您没有必要设定高宽为图片的显示创造空间，或是使用text-indent或是padding隐藏最初的文字。

然而，就目前而言，还是有不少需要注意的：  
1. 你不能对图片进行重复或平铺，或是使用image sprite。  
2. 目前仅在Opera 9.5+ 和 Safari 4+浏览器下起作用，因为这些浏览器支持所有元素的content方法，而不仅仅局限于:after或是:before。  
替换的图片无法应用alt属性，所以一些特殊的使用屏幕阅读器的用户可能无法理解您替换的图片的含义。

### 七、显示相对应的链接图标

指的是针对不同的链接类型，在链接的后面显示对应链接类型的图标，例如，链接的对象是一个图片，则显示图片的小图标，如果链接对象是视频，则显示视频小图标，如果链接就是个URL网页链接，则显示链接的小图标。您可以参见下面的代码：

CSS代码：

p a[href $=".pdf"]:after {

content:url(../image/icon\_pdf.png);

}

p a[rel = "external"]:after {

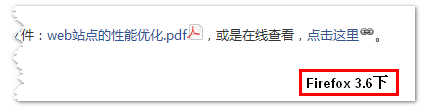
content:url(../image/icon\_link.png);

}

HTML代码：

<p>您可以查看此PDF文件：<a href="/sample.pdf">web站点的性能优化.pdf</a>，或是在线查看，<a href="http://www.zhangxinxu.com/wordpress/" rel="external">点击这里</a>。</p>

结果如下图所示：



您可以狠狠地点击这里：[CSS content显示链接图标demo](http://www.zhangxinxu.com/study/201004/css-content-link-image.html)

### 八、使用属性值作为content内容

使用属性值当作content内容指的是将标签的title,src,href,alt等值当作内容显示在标签的内部，这是很强大的一个功能。由于此技术目前远未到普及的程度，所以这里简单介绍一下，您可以参见下面的两小段样式代码：

a:after {

content: "(" attr(href) ")";

}

abbr:after {

content: "(" attr(title) ")";

}

这里理解和使用都不难，关键就是CSS3 的attr()方法。

**前瞻：强大的CSS3 attr()方法。**  
[CSS3 Values and Units](http://www.w3.org/TR/css3-values/#attribute)草案扩展了attr()表达式的范围，除了返回字符串，还可以返回诸如CSS colors, CSS integer, length, angle, time, frequency以及其他一些单元。

通过使用自定义的data属性，可以实现一些非常强大的效果，诸如简单图表图形的渲染，动画效果的实现。例如我们可以设置根据元素的attribute值设定background-color的颜色值，这在显示在线调色板上会大放异彩；我们还可以根据attribute值指定元素的大小，就像定义图表每个条形的长度一样。总之，attr()的潜力惊人。

### 九、最后一点唠叨

相信本文对您理解content属性会有不少的帮助。只是在中国，在IE6依旧盛行的中国，由于IE6/7对content的不支持，限制了这一强大功能的流行，但我相信，在未来几年内，会迎来它的春天的。随着对CSS3的逐步深入的了解，会发现，CSS3的潜力与强大真是超乎想象。我可以预见再未来几年，会有类似于寒武纪爆发的CSS新技术涌现出来，那时，可真是前端开始走向兴盛的时代。

**参考文章：**  
1. [CSS generated content techniques](http://dev.opera.com/articles/view/css-generated-content-techniques/)

## outline

outline （轮廓）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

**注释：**轮廓线不会占据空间，也不一定是矩形。

outline 简写属性在一个声明中设置所有的轮廓属性。

可以按顺序设置如下属性：

* outline-color
* outline-style
* outline-width

**实例**

设置 4 个边框的样式：

p

{

outline:#00FF00 dotted thick;

}

## pointer-events

使用该属性可以决定是否能穿透绝对定位元素去触发下面元素的某些行为

pointer-events: auto | none | visiblePainted | visibleFill | visibleStroke | visible | painted | fill | stroke | all | inherit

pointer-events属性有很多值，但是对于浏览器来说，只有auto和non两个值可用，其它的几个是针对SVG的(本身这个属性就来自于SVG技术)。

**pointer-events属性值详解**

auto——效果和没有定义pointer-events属性相同，鼠标不会穿透当前层。在SVG中，该值和visiblePainted的效果相同。

none——元素不再是鼠标事件的目标，鼠标不再监听当前层而去监听下面的层中的元素。但是如果它的子元素设置了pointer-events为其它值，比如auto，鼠标还是会监听这个子元素的。

其它属性值为SVG专用，这里不再多介绍了。

## 使用 text-overflow:ellipsis 实现 CSS 文本溢出省略号

.entry\_title{

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

}

上面 CSS 第一行是设置强制文本在一行内输出，第二行是设置溢出处理方式，这里是隐藏，第三行是设置文本溢出的处理方式，这里是末尾加上省略号。这样简单处理下，就能完美实现文本溢出省略号了。

## appearance

[CSS 参考手册](http://www.w3school.com.cn/cssref/index.asp)

### 实例

使 div 元素看上去像一个按钮：

div

{

appearance:button;

-moz-appearance:button; /\* Firefox \*/

-webkit-appearance:button; /\* Safari 和 Chrome \*/

}

[亲自试一试](http://www.w3school.com.cn/tiy/t.asp?f=css3_appearance)

页面底部有更多实例。

### 浏览器支持

所有主流浏览器都不支持 appearance 属性。

Firefox 支持替代的 -moz-appearance 属性。

Safari 和 Chrome 支持替代的 -webkit-appearance 属性。

### 定义和用法

appearance 属性允许您使元素看上去像标准的用户界面元素。

|  |  |
| --- | --- |
| **默认值：** | normal |
| **继承性：** | no |
| **版本：** | CSS3 |
| **JavaScript 语法：** | *object*.style.appearance="button" |

### 语法

appearance: normal|icon|window|button|menu|field;

|  |  |
| --- | --- |
| **值** | **描述** |
| normal | 将元素呈现为常规元素。 |
| icon | 将元素呈现为图标（小图片）。 |
| window | 将元素呈现为视口。 |
| button | 将元素呈现为按钮。 |
| menu | 将元素呈现为一套供用户选择的选项。 |
| field | 将元素呈现为输入字段。 |

## @font-face

[**@font-face**](http://www.w3.org/TR/css3-fonts/)是[CSS3](http://www.w3.org/TR/CSS/#css3)中的一个模块，可引入自定义的字体。

详见[CSS3 @font-face](http://www.w3cplus.com/content/css3-font-face)