# 第一章

## 学习CSS最好的方式

学习CSS最好的一种方式是直接开始使用它，先从博客上看到了一些出色的效果，通过查看源代码研究他们是如何实现的，然后就在自己的个人网站上大胆尝试。人们几乎不会先去读完整的CSS规范，这些规范能把任何人送入梦乡。

## 弃用表格布局的原因：

* 结构太复杂，嵌套表格和分割线GIF把代码弄得非常混乱
* 很容易被破坏
* 难以维护
* 重用性低

## shim GIF

## HTML包含丰富的有意义元素

HTML4元素表：<http://www.w3school.com.cn/tags/index.asp>

按功能类别划分：<http://www.w3school.com.cn/tags/html_ref_byfunc.asp>

HTML5元素表：<http://www.w3school.com.cn/html5/html5_reference.asp>

HTML5为开发人员提供更丰富的元素：header\nav\article\section\footer。为了准备迎接HTML5，许多开发人员已经开始采用这些名称作为ID和类名的命名约定，并给样式提供有用的“钩子”(hook)

<ul id=”nav”>

<li><a href=”/home/”>Home</a></li>

<li><a href=”/about/”> About</a></li>

<li><a href=”/contact/”> Contact us </a></li>

</ul>

## ID与Class的适用环境

ID适用环境：

标识页面上的特定元素，比如站点导航

标识持久的结构性元素，例如主导航或内容区域

标识一次性元素，例如某个链接或表单元素

class适用环境：

标识内容的类型或其他相似的条目

只有在目标元素非常独特，绝不会在其他地方使用时，才会使用ID。否则就使用类。

不需要用很多类来区分各个元素。可以将一个父元素加上类名，其子元素用层叠来标识。只要你发现雷鸣中出现了重复的单词：比如news-head\news-link或section-head\section-foot，就该考虑是否是“多类症”了。

## 为元素命名法则

应该说明它们是什么，而不是用表现性元素来说明它们的外观如何。大小写是区分的，推荐全小写，中间用连字符分隔。

## 多Div症

过度使用div常常称为“多div症”div实际上代表部分，它可以将文档分割成几个有意义的区域，所以通过将主要内容区域包围在div中并非配Content类，就可以再文档中添加结构和意义。但没必要用Div包裹的元素就无需包裹，比如主导航列表，可以直接用ul来包裹。当然，有时候为了以自己希望的布局显示页面，无法避免添加额外的无语义的div或span,这样那也不必过分为此担心，CSS3有望提供更强大的文档控制能力。

CSS3有望提供更强大的文档控制能力。

## 微格式

XFN微格式

## CSS与HTML的版本

CSS2.1

HTML4与XHTML1.1用哪个？HTML4更受欢迎，原因如下：

XHTML属性必须包含引号，所有元素必须是封闭的，即使页面只包含一个错误，文本浏览器也不会显示页面。

下一代，迎接HTML5

## 文档类型、DOCTYPE切换和浏览器模式

* DOCTYPE声明指定了使用哪个DTD，由此知道要使用HTML的哪个版本，DOCTYPE通常-但不总是-包含指定的DTD文件的URL，比如HTML5就不需要URL。
* DOCTYPE有两种风格：Strict和Transitional，HTML4和XHTML1都有。
* HTML元素与合法的 Doctype：http://www.w3school.com.cn/tags/html\_ref\_dtd.asp
* DOCTYPE用途：

1. 有效性验证
2. 浏览器模式-DOCTYPE切换

* 浏览器模式有两种：混杂模式和标准模式。包含过渡DTD而没有URI的DOCTYPE会导致页面以混杂模式呈现，DOCTYPE不存在或形式不正确会导致HTML和XHTML文档以混杂模式呈现。其他不管是strict还是transitional，只要格式完整，都会以标准模式来呈现。
* 根据DOCTYPE是否存在选择呈现模式，被称为DOCTYPE切换或侦测。浏览器用其来区分遗留文档和符合标准的文档，无论是否编写了有效的CSS，如果选择了错误的DOCTYPE，那么页面将以混杂模式呈现，其行为就可能会有错误或不可预测。因此，一定要在站点的每个页面上包含形式完整的DOCTYPE声明，并且在使用HTML时选择严格的DTD。
* 不同浏览器如何根据Doctype声明来选择呈现方法：http://hsivonen.iki.fi/doctype/

# 第二章 选择器

## 常用选择器

* 类型（元素）选择器
* 后代选择器：以空格隔开就代表后代
* 同级选择器：如：<a class=”link”>在ExtJS常需要用到.中间不要用空格即可,即a.link{}
* ID选择器
* 类选择器

结合使用类型/ID/类选择器+后代选择器就可以成功找到许多元素

如果你发现自己在文档中添加了许多不必要的类，那么这可能是文档结构不合理的一个警告信号。这时应该分析这些元素之间的差异。你常常会发现唯一的差异是他们在页面上出现的位置。不要给这些元素指定不同的泪，而应将一个类或ID应用于它们的祖先，然后使用后代选择器定位它们。

* 伪类

a:link

a:visited

a:hover

a:focus

a:active

:link,:visited称为链接伪类，只能应用于锚元素。

:hover,:active,:focus称为动态伪类，理论上可以应用于任何元素。IE7不支持:active,:focus

通过把伪类连接在一起，可以创建更复杂的行为，例：a:visited:hover{},在已访问过的链接上的悬停效果

## 通用选择器

\*

在于其他选择器结合使用时，通用选择器可以用来对某个元素的所有后代应用样式，或者跳过一级后代

## 高级选择器

### 子选择器和相邻同胞选择器

子选择器只选择元素的直接后代

#nav>li

IE7和更高版本都支持子选择器，但IE7中有一个小bug,如果父元素和子元素之间有HTML注释，就会有问题。

在IE6和更低版本中，可以使用通用选扎起模拟子选择器的效果。

为此，先在所有后代上应用你希望子元素具有的样式。然后，使用通用选择器覆盖子元素的后代上的样式。所以，要实现与前面的子选择器示例相同的效果，可以这样做：

#nav li{}//子元素的样式

#nav li \*{}//子元素后代的样式

相邻同胞选择器：h2+p{}

<h2></h2>

<p></p>

<p></p>

对第一个p有效

与子选择器一样，如果在目标元素之间有注释，这在IE7中也会有问题

### 属性选择器

* p[title]——含有title这个属性的p标签
* p[title=”follow”]——含有title这个属性，且值为follow的p标签
* p[title=”follow”]:hover, p[title=”follow”]:focus——含有title这个属性，且值为follow的hover和focus状态的样式
* 注意：IE6不支持属性选择器，可选用替代方案：

<p title=”follow”>(IE7以上)=><p id=”follow”>（IE6）

[title=”follow”]{}（IE7以上）=>#follow{}（IE6）

一些属性可以有多个值，值之间用空格分隔，如：

<a href=”” rel=”friend co-worker colleague”>

属性选择器允许根据属性值之一寻找元素

a[rel~=”co-worker”]{…………}

~表示包含某一个值，此外还有不包含，不等于某属性等选择方式。

### 层叠和特殊性

* 层叠重要度

层叠采用以下重要度排序：

标有！important的用户样式

标有！important的作者样式

作者样式

用户样式

浏览器/用户代理应用的样式

* 特殊性

计算特殊性的方法：每种选择器都分配一个数字值，然后将规则的每个选择器的值加在一起，计算出规则的特殊性。

选择器的特殊性分为4个成分等级：a、b、c和d

* 如果样式是行内样式（即style属性），那么a=1
* b等于ID选择器的总数
* c等于类、伪类和属性选择器的数量
* d等于类型选择器（即标签）和伪元素选择器的数量

然后根据abcd的值合成abcd这个数字，越大的特殊性越高。

如果你遇到了似乎没有起作用的CSS规则，很可能是出现了特殊性冲突，请在你的选择器中添加他的一个父元素的ID，从而提高它的特殊性。如果这能够解决问题，就说明样式表中其他地方很可能有更特殊的规则，它覆盖了你的规则。

在编写CSS时特殊性非常有用，因为它可以对一般元素应用一般样式，然后在更特殊的元素上覆盖它们。对于小网站，这很好。但是在大型站点上，你会发现例外情况越来越多。例如，你可能希望新闻文章上的介绍文本是蓝色的，而主页上的介绍文本使用灰色背景。每当创建更特殊的样式时，可能需要覆盖一些一般规则。这可能需要一些额外的代码。而且，因为元素可以从许多地方获得样式，情况可能变得非常复杂。

为了避免过分混乱,我尽量保持一般性样式非常一般，特殊样式尽可能特殊，这样就不需要覆盖特殊样式了。如果发现不得不多次覆盖一般样式，那么从更一般的规则中删除需要覆盖的声明，并且将它显式地应用于需要它的每个元素，这样可能比较简单。

* 在主体标签上添加类或ID

一种有意思的特殊性用法是在主体标签上应用类或ID，这样做之后，就可以根据页面或在站点范围内覆盖样式。例如，如果希望新的页面具有特殊的布局，那么可以在主页的主题元素添加一个类名，并且使用它覆盖样式。

body.news{

…………..

}

<body class=”news”>

<p></p>

</body>

有时候，在特殊页面上需要覆盖这些样式，比如在新闻存档页面上，在这种情况下，可以在主体标签上添加ID来标识这个页面。

body.news{

…………..

}

body#archive{

……………….

}

<body id=”archive” class=”news”>

<p>…</p>

</body>

### 继承

要与层叠分清楚，继承是指子元素继承父元素的属性，有些属性很多子元素都有，写在父元素就可以统一管理了。

继承使用时要注意：如果大量元素继承各种样式，那么判断样式的来源就会变得困难。一般性规则太多了。

## 规则、组织和维护样式表

* 使用结构良好的单一CSS文件可以显著提高下载速度。
* 对代码进行注释是一种良好的习惯，防止过了一段时间后忘记自己写的代码。如果CSS文件非常长，那么寻找特定的样式会很困难，一种改进方法是在每个注释头中添加一个标志，搜索这个标志和注释头中的几个字母，就可以立即找到要寻找的文件部分。所以，在下面的实力中，搜索“@group typ”就会立即找到样式表中的版式部分。/\*@group typegraphy\*/

### 设计代码的结构

为了便于维护，最好把样式表划分为几大块，显然，常常把最一般的规则放在最前面，这包括应用于body标记的，应该由站点上所有元素继承的样式。接下来是可能需要的所有全局reset样式，然后是链接，标题和其他元素。

完成一般样式之后，开始处理更特殊的样式和辅助样式。这些是在整个站点中使用的一般类，包括表单和错误信息等方面。然后，处理布局和导航等结构性元素。

随着在样式表中的移动，我们在一层样式上构建另一层样式，处理的样式越来越特殊，处理完页面结构元素之后，我们把注意力转到与特定页面相关的组件上。最后，在文档的地步处理覆盖和例外情况。整个文档的结构像下面这样：

* + 一般性样式
    - 主体样式
    - reset样式
    - 链接样式
    - 标题
    - 其他元素
  + 辅助样式
    - 表单
    - 通知和错误
    - 一致的条目
  + 页面结构
    - 标题、页脚和导航
    - 布局
    - 其他页面结构元素
  + 页面组件
    - 各个页面
  + 覆盖

这里使用一种风格统一的大注释块分割每个部分

/\*@group general styles

-----------------------------------------------------------------------------------------------------------------\*/

/\*@group helper styles

-----------------------------------------------------------------------------------------------------------------\*/

/\*@group page structure

-----------------------------------------------------------------------------------------------------------------\*/

/\*@group page components

-----------------------------------------------------------------------------------------------------------------\*/

/\*@group overrides

-----------------------------------------------------------------------------------------------------------------\*/

因为许多CSS文件可能具有相似的结构，所以可以创建一个预先加上注释的CSS模板供所有项目使用，从而节省时间。还可以添加在所有站点上都使用的常用规则，形成某种原型CSS文件，这可以节省更多的时间。这样的话，在开始新项目时就不必总是重复以前的工作。

### 自我提示

对于复杂的大型项目，在CSS文件中添加临时的注释常常对开发有帮助，这些注释可以提醒你在启动前需要完成哪些工作，或者提醒你提供列宽度等常用值的查询表。

例如，如果设计中使用了许多种颜色，你可能常常需要到图形应用程序中查询十六进制值，然后再返回到文本编辑器。这非常浪费时间，所以一些人以为需要CSS变量。尽管这是一个很有意思的主意，但是这会让CSS更接近真正的编程语言，可能让非程序员畏惧它。因此，我常常使用一种简单的方法。我在样式表的开头添加一个小的颜色查询表，这样就可以在开发期间经常参考它，完成开发之后，我通常会删除它。

/\*Color Variables

@colordef #434343:dark gray

@colordef #f2f6e4:light green

@colordef #90b11f:dark green

@colordef #369:dark blue

\*/

为了使注释更有意义，可以使用关键字来区分重要的注释。我使用@todo来表示某些东西需要在以后进行修改、修复或复查。用@bugfix表示代码或特定浏览器遇到的问题，用@workaround表示并不完善的权宜之计：

/\*:@todo Remember to remove this rule before the site goes live\*/

/\*@workaround:I managed to fix this problem in IE by setting a mail negative margin but it’s not pretty\*/

/\*@bugfix:Rule breaks in IE 5.2 Mac\*/

实际上，这些词汇都是CSSDoc项目的一部分<http://cssdoc.net/>

### 删除注释和优化样式表

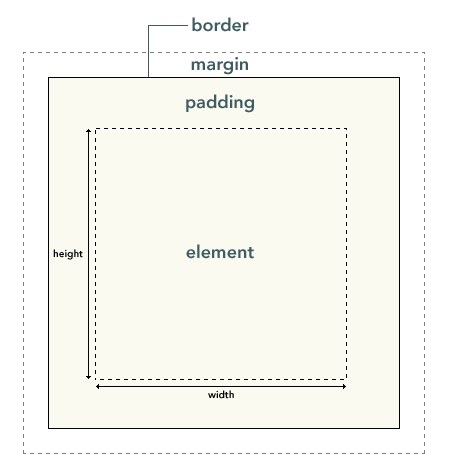
用CSS minify工具和Gzip工具

### 样式指南

建立样式指南可以帮助其他前端开发人员可能需要了解代码的元素和设计是如何工作的

# 第三章 可视化格式模型

## 盒模型



### 盒模型的组成

页面上的每个元素被看作一个矩形框。这个框由元素的内容、内边距、边框和外边距组成。如果在元素上添加背景，那么背景会应用于由内容和内边距组成的区域。因此我们常常使用内边距在内容周围创建一个隔离带，使内容不会与背景混在一起。

添加边框会在内边距的区域外边加一条线。这些线可以有多种样式：实线、虚线或点线。

在边框外边的是外边距。外边距是透明的，一般使用它控制元素之间的间隔。

外边距还可以是负值，比如需要元素向左偏移、或与左边元素叠加时可以用到。

关于盒模型相关CSS属性的参考手册，参看[附录1：padding、border、margin](#_附录1：Padding、Border、Margin)

CSS2.1还包含outline属性，与border属性不同，轮廓绘制在元素框之上，所以他们不影响元素的大小或定位。因此轮廓有助于修复bug,因为他们不影响页面的布局。IE8以及其他浏览器支持这一属性，IE7及以下不支持轮廓。

width和height指的是内容区域的宽度和高度。与内边距、边框、外边距无关

### 覆盖用户代理样式表

内边距、边框和外边距默认值为0.但是许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的margin或padding设置为0来覆盖这些浏览器样式，也就是用户代理样式。这可以分别进行，也可以使用通用选择器对所有元素进行设置：

\*{

margin：0；

padding：0；

}

由于这种技术不区分元素，所以它对option等元素有不利影响。因此使用全局reset样式把内边距和外边距显式地设置为0可能更安全。

## IE和盒模型

一般情况下，width=内容的宽度，在IE6中及之前版本的混杂模式中=内容+内边距+边框之和

CSS3的box-sizing属性可以定义要使用哪种盒模型，但是除了一些非常特殊的场合之外很少需要使用这个属性。

解决方法在第9章

## 外边距叠加

只有普通文档流中块框的垂直外边距才会发生外边距叠加。行内框、浮动框或绝对定位框之间的外边距不会叠加。

## 块框、行内框

可继续补充

|  |  |
| --- | --- |
| 默认的块框 | div、p、h1 |
| 默认的行内框 | strong、span、a |

可以使用display属性改变生成的框的类型。通过将display属性设置为block让行内元素表现的像块级元素一样。块框同样可以改变成行内框。还可以将display属性设置为none,让生成的元素根本没有框，在文档流中不占地方。

## 定位机制

CSS中有3种基本的定位机制：普通流，浮动，绝对定位。

行内框在一行中水平排列，可以使用水平内边距，边框和外边距调整它们的水平间距。但是，垂直内边距，边框和外边距不影响行内框的高度。同样，在行内框上设置显式的高度或宽度也没有影响。由一行形成的水平框称为行框，行框的高度总是足以容纳它包含的所有行内框。但是，设置行高可以增加这个框的高度。因此，修改行内框尺寸的唯一方法是修改行高或者水平边框，内边距或外边距。

CSS 2.1允许把元素的display属性设置为inline-block,但是，框的内容仍然符合块级框的行为，例如能够显示地设置宽度，高度，垂直外边距和内边距。

## 匿名块框、行内框

## 相对定位

绝对定位

固定定位

浮动

浮动的清理

# 第四章 背景图像效果

## 渐变

背景图像可以垂直平铺、水平平铺或根本不平铺。如果在页面上应用垂直渐变，需要创建一个很高但很窄的渐变图像，然后将这个图像应用于页面的主体并让它水平平铺。

|  |
| --- |
| body{  background-image:url(/img/gradient.gif);  background-repeat:repeat-x;  } |

因为这个渐变图像的高度是固定的，所以如果页面内容的长度超过了图像的高度，那么渐变就会突然终止。可以创建一个非常长的图像，逐渐变化到一个固定的颜色，但是，很难预测页面会有多长。实际上，只需再添加一个背景颜色。背景图像总是出现在背景颜色的上面，所以当图像结束时，颜色就会显示出来。如果选择的背景颜色与渐变底部的颜色相同，那么图像和北京颜色之间的转换就看不出来了。

## 钩子

平铺图像在某些情况下很有用。但是，在大多数情况下，需要在页面上添加不是平铺的图像。例如，假设希望在网页的开头显示一个大的品牌图像，那么只需将图像直接添加到页面上，在许多情况下这样做就够了。但是，如果图像不包含信息，是纯表现性的，那么可能希望将图像从其余内容中分离出来。实现的方法是在HTML中为这个图像创建一个“钩子”，然后使用CSS应用这个图像。比如：

|  |
| --- |
| <div id=”branding”>  #branding{  width:700px;  height:200px;  background-image:url(/img/branding.gif)  background-repeat:no-repeat;  } |

## 背景图像的位置

还可以设置背景图像的位置。假设要在站点的每个标题上添加一个项目符号

|  |
| --- |
| h1{  padding-left:30px;  background-image:url(/img/bullet.gif)  background-repeat:no-repeat;  background-position:left center;  } |

背景位置的意思是图像左上角到元素左上角的距离。

百分比定位并不对背景图像的左上角进行定位，而是使用图像上的一个对应点。假如是20%，是将图像上具左上角20%的点定位到父元素上距离左上角20%的位置。

## 圆角框

### 固定宽度的圆角框

这是最容易的圆角框，它们只需要两个图像：一个图像用于框的顶部，一个用于底部。

<div class=”box”>

<h2>Headline</h2>

<p>Content</p>

</div>

将顶部图像应用于标题元素，降低不图像应用于div框的底部。因为这个框样式是单色的，所以可以在div框上添加背景颜色，从而形成框的主体。

.box{

width:418px;

background:#effce7 url(/img/bottom.gif) no-repeat left bottom;

padding-bottom:1px;

}

.box h2{

background:url(/img/top.gif) no-repeat left top;

margin-top:0;

padding:20px 20px 0 20px;

}

.box p{

padding:0 20px;

}

如果不是纯色的，而是周边有边框的。需要做一个渐变的图像，在y轴上渐变。

|  |
| --- |
| <div class=”box”>  <h2>Headline</h2>  <p class=”last”>Content</p>  </div> |

|  |
| --- |
| .box{  width:418px;  background: url(/img/tiles2.gif) repeat-y;  }  .box h2{  background:url(/img/top.gif) no-repeat left top;  padding:20px 20px 0 20px;  }  .box .last{  background: url(/img/bottom.gif) no-repeat left bottom;  } |

因为没有给这个框设置高度，所以它会随着文本尺寸的增加进行垂直扩展。

### 可在一定范围内水平扩展的圆角框（4宫格）

这个方法有时候被称为滑动门技术。因为一个图像在另一个图像上滑动。将它的一部分隐藏了起来。

|  |
| --- |
| <div class=”box”>  <div class=”box-outer”>  <div class=”box-inner”>  <h2>Headline</h2>  <p>Content</p>  </div>  </div>  </div> |

这个方法需要4个图像：两个顶部图像组成顶部曲线，两个底部图像组成底部曲线和框的主体。将bottom-left.gif应用于主框div,将bottom-right.gif应用于外边的div。接下来将top-left.gif应用于内部的div,将top-right.gif应用于标题。

|  |
| --- |
| .box{  width:20em;  background:#effce7 url(/img/bottom-left.gif) no-repeat left bottom;  }  .box-outer{  background:url(/img/bottom-right.gif)) no-repeat right bottom;  padding-bottom:1em;  }  .box-inner{  background:url(/img/top-left.gif)) no-repeat left top;  }  .box h2{  background:url(/img/top-left.gif)) no-repeat right top;  } |

### ?山顶角

### CSS3

#### 多背景图像

CSS3可以在一个元素上添加多个背景图像，这样就不用为了使用4个背景图片而使用一些无意义元素。

<div class=”box”>

<h2>Headline</h2>

<p>Content</p>

</div>

.box{

background-image:url(/img/top-left.gif),url(/img/top-right.gif), url(/img/bottom-left.gif) ,url(/img/bottom-right.gif);

background-repeat:no-repeat, no-repeat, no-repeat, no-repeat

background-position:top left, top right, bottom left, bottom right

}

IE目前还不支持多个背景图像，但是这并不妨碍使用这种技术，只是IE用户看到的是直角框。

#### border-radius

.box{

border-radius:1em;

}

也可以做成圆角

当前，Firefox和Safari支持这个属性，所以我使用-moz和-webkit前缀

.box{

-moz- border-radius:1em;

-webkit -border-radius:1em;

border-radius:1em;

}

#### border-image

border-image属性自动地把图像的每个部分用于对应的边框。英雌，图像的左上部分用作左上边框，右边中间部分用作右边的边框。我希望边框的宽度为25像素，所以在CSS中设置这个宽度。如果图像不够大，他们会自动地平铺，产生一个可扩展的框。

原图：



实现这种效果的代码如下：

.box{

-webkit-border-image:url(/img/corners.gif) 25% 25% 25% 25% /25px round round;

}

## 投影

### 简单的CSS投影-负外边距

它的工作原理是：将一个大的投影图像应用于容器div的背景。然后使用负的外边距便宜这个图像，从而投影出投影。

<div class=”img-wrapper”><img src=”dunstan.jpg” width=”300” height=’300’ alt=”Dunstan Orchard”/></div>

.img-wrapper{

background:url(/img/shadow.gif) no-repeat bottom right;

clear:right;

float:left;

}

.img-wrapper img{

margin:-5px 5px 5px -5px;

}

还可以给图像加上边框和一些内编剧，从而产生类似照片边框的效果

.img-wrapper img{

background-color:#fff

border:1px solid #a9a9a9;

padding:4px;

margin:-5px 5px 5px -5px;

}

这种技术对于大多数符合标准的现代浏览器都是有效的。但是，为了在IE6中产生正确的效果，还需要添加两个简单的规则。

.img-wrapper{

background:url(/img/shadow.gif) no-repeat bottom right;

clear:right;

float:left;

position:relative;

}

.img-wrapper img{

background-color:#fff

border:1px solid #a9a9a9;

padding:4px;

display:block;

margin:-5px 5px 5px -5px;

position:relative;

}

### 相对定位

|  |
| --- |
| .img-wrapper{  background:url(/img/shadow.gif) no-repeat bottom right;  float:left;  line-height:0;  }  .img-wrapper img{  background:#fff  border:1px solid #a9a9a9;  padding:4px;  display:block;  left:-5px;  top:-5px;  } |

### CSS3

box-shadow

img{

box-shadow:3px 3px 6px #666;

}

这也是一个试验性的CSS3属性。所以目前需要使用Safari和Firefox扩展。但是这个属性可能很快就会得到广泛支持了。

img{

-webkit-box-shadow:3px 3px 6px #666;

-moz-box-shadow:3px 3px 6px #666;

box-shadow:3px 3px 6px #666;

}

这个特性最让人兴奋的特点之一是它会与border-radius属性相互配合，这意味着可以通过变成在圆角框上创建投影，根本不需要使用图形软件。

## 不透明度

是当地使用不透明度可以让设计的效果更丰富，对于相互重叠的元素，还可以用它显露下面的元素。除了这个很酷的用途外，这还可以改进站点的易用性。

### CSS不透明度

大多数现代浏览器支持CSS不透明度已经有相当一段时间了，但让我吃惊的是设计者不经常使用它。IE的老版本不支持它，但是专门针对IE的代码很容易解决这个问题。

.alert{

background-color:#000；

border-radius:2em;

opacity:0.8;

filter:alpha(opacity=80);// for IE

}

CSS不透明度的主要问题是会看到底层的内容，如果使用非常高的不透明度和高对比度的文本，这可能不是问题。但是，如果不透明度低，框的内容就会难以辨认。RGBa就是为了解决这个问题而设计的。

### RGBa

RGB代表红绿蓝，a代表alpha透明度。前面一个示例中使用RGBa的方法如下：

.alert{

background-color:rgba(0,0,0,0.8);

border-radius:2em;

}

最后一个值为1表示100%不透明，值为0表示完全透明。

### PNG透明度

PNG文件格式最大的优点之一是它支持alpha透明度。这可以使设计具备真正的创意。IE6不支持PNG透明度，而IE7和IE8支持。

在IE6中支持PNG透明度的方法是使用专有的AlphaImageLoader过滤器，为此，需要在CSS中包含以下代码行。

filter:progid:DXImageTransform.Microsoft.AlphaImageLoader

(src=’/img/my-image.png’,sizingMethod=’crop’);

但是使用这样代码会导致CSS无效，所以最好把它放在IE6专用的样式表中。

.img-wrapper div{

filter:progid:DXImageTransform.Microsoft.AlphaImageLoader

(src=’/img/my-image.png’,sizingMethod=’crop’);

background:none;

}

第一个规则使用专有的过滤器加载PNG并应用alpha透明度。原来的背景图像仍然会显示，所以第二个规则隐藏原来的背景图像。

另一种方法是使用IE PNG Fix技术。

img,div{

behavior:url(iepngfix.htc)

}

[www.twinhelix.com/css/iepngfix](http://www.twinhelix.com/css/iepngfix)

以上专为IE6写的CSS最好放在IE6专用样式表中

IE6有条件注释

希望只让IE6看到这个IE6专用的样式表，所以可以在页面顶部添加以下代码：

<!--[if ie 6]>

<link rel=”stylesheet” type=”text/css” href=”ie6.css”/>

<![endif]-->

### CSS视差效果

背景图像不仅可以创建圆角框和投影，还可以实现许多有趣的效果。访问Silverback易用性测试应用程序就可以体会到这一点，如果访问[www.silverbackapp.com](http://www.silverbackapp.com/)并重新调整浏览器窗口的大小，就会看到一种奇怪的效果。背景图像会议稍微不同的速度移动，让人觉得这个页面有深度。这种现象称为时差滚动。要想实现这种效果，首先需要创建几个不同的背景图像：

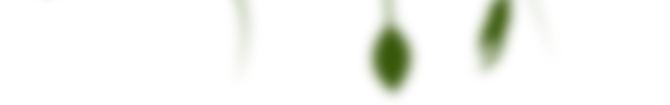
bg-rear.gif



bg-mid.png



bg-front.png



主背景将应用在body元素上。但是如果我们不使用CSS3的多背景图像特性，就需要添加两个新元素来应用背景。

<body>

<div class="midground">

<div class="foreground">

<div class="content">

<h1>Your content will go here!</h1>

</div>

</div>

</div>

</body>

body的CSS

|  |
| --- |
| body {  background-image: url(img/bg-rear.gif); //背景图像  background-repeat: repeat-x; //水平平铺  background-color: #d3ff99; //背景色，浅绿色  background-position: 20% 0; //把图像相对于窗口水平偏移20%,left=20%  } |

其他的CSS，背景图像的位置是关键，可以调整这些位置，得到自己需要的效果。

|  |
| --- |
| .midground {  background-image: url(img/bg-mid.png);  background-repeat: repeat-x;  background-color: transparent;  background-position: 40% 0;  }  .foreground{  background-image: url(img/bg-front.png);  background-repeat: repeat-x;  background-color: transparent;  background-position: 150% 0;  } |

## 图像替换（图像按钮代替HTML按钮）

HTML文本有很多优点。搜索引擎可以读取它，开发人员可以复制并黏贴它，如果在浏览器中增加文本字号，就可以让它放大。因此，尽可能使用HTML文本而不是文本的图像是一种好想法。

遗憾的是，页面设计人员能选择的字体有限，另外，尽管可以使用CSS在某个范围内控制版式，但是某些效果对于文本是不可能实现的。因此，在这些情况，只能使用文本的图像。

CSS作者发明了图像替换的概念，你可以像平常一样将文本添加到文档中，然后使用CSS隐藏文本并在他的位置上显示一个背景图像。这样的话，搜索殷勤仍然可以搜索HTML文本，如果禁用CSS，文本仍然会显示。三种流行的图像替换技术：

### FIR

原则：将文本设为隐藏

缺点：许多流行的屏幕阅读器会忽略display为none或visibility为hidden的元素，造成严重的可访问性问题。

<h2>

<span>Hello World</span>

</h2>

h2{

background:url(hello\_world.gif) no-repeat;

width:150px;

height:35px;

}

span{

display:none;

}

### Phark

原则：把字的缩进设得超出浏览器范围

缺点：在关闭图像但是打开CSS的情况下是无效的，所以会看不到文本。

<h2>

Hello World

</h2>

h2{

text-indent:-5000px;

background:url(/img/hello\_world.gif)

width:150px;

height:35px;

}

### sIFR

图像替换试图解决的主要问题之一是在大多数计算机上缺少可用的字体，为了避免将文本换成文本的图像，Mike Davidson和Shaun Inman发明了一种更新颖的方法。

原则：用Flash文件替换文本

Shaun Inman公开了他的Flash图像替换方法，并且将它命名为IFRame（Inman Flash替换）关于这个方法的细节（包括源码）可以在[www.shauninman.co,/plete/2004/04/ifr-revisited-and-revised](http://www.shauninman.co,/plete/2004/04/ifr-revisited-and-revised)上找到。

Mike Davidson对这个方法进行了扩展，创建了sIFR(可伸缩Inman Flash替换)方法。

# 第五章 对链接应用样式

## 伪类选择器

|  |  |
| --- | --- |
| :link | 没有被访问过的链接 |
| :visited | 被访问过的链接 |
| :hover | 鼠标悬停的链接 |
| :active | 被激活的链接 |
| :focus | 链接聚焦时的样式 |

1. 为了尽可能提高页面的可访问性，在定义鼠标悬停状态时，最好在连接上添加:focus,在通过键盘移动到链接上时，这让链接显示的样式与鼠标悬停时相同。

a:hover,a:focus{color:red;}

1. 其他元素也可以使用:hover,:active或:focus伪类选择器。例如，可以在表格行上添加:hover伪类，在提交按钮上添加:active伪类，在输入框上添加:focus伪类。

IE7以及更低版本不支持除链接外在其他元素上使用伪类选择器。

1. 大多数人最初使用这些选择器做的第一件事就是去掉链接的下划线。然后在悬停、激活，聚焦链接时加上下划线：

a:link,a:visited{text-decoration:none;}

a:hover,a:focus,a:active {text-decoration:underline;}

1. 上个示例中，选择器的次序非常重要，如果次序反过来，鼠标悬停和激活样式就不起作用。

这是由于层叠造成的，在第一章中提到的，当两个规则具有相同的特殊性时，后定义的规则优先，所以在这个示例中，:link和:visited样式将覆盖:hover和:active样式，为了确保不发生这种情况，最好按照以下次序应用链接样式：

:link,:visited,:hover,:focus,:active

## 简单的链接修饰

从易用性和可访问性的角度来说，通过颜色之外的某些方式让链接区别于其他内容是很重要的。这是因为许多有视觉障碍的人很难区分对比不强烈的颜色，尤其是在文本比较小的情况下。因此，默认情况下，链接是有下划线的。

但是，设计人员往往不喜欢链接的下划线，因为下划线让页面看上去比较乱。如果决定去掉链接的下划线，那么可以让链接显示为粗体。这样的话，页面看起来没那么乱，而链接仍然醒目。

a{font-weight:bold;}

也可以使用边框创建不太影响美观的下划线，比如点线，当鼠标悬停在链接上或激活链接时，这条线变成实线，从而向用户提供视觉反馈

a:link,a:visited{text-decoration:none:border-bottom:1px dotted #000}

a:hover,a:focus,a:active{border-bottom-style:solid;}

通过使用背景图像创建链接下划线，可以产生非常有意思的效果。

a:link,a:visited{color:#666;text-decoration:none;background:url(/img/underline1.gif) repeat-x left bottom;}

并非所有浏览器都支持背景图像动画，但是不支持这个特性的浏览器常常会显示动画的第一帧，这能确保效果在老式浏览器中可以平稳退化。

## 已访问链接的样式

不同的已访问链接样式可以帮助用户，让他们知道哪些页面或站点他们已经访问过了，避免不必要的“回溯”操作。

通过在每个已访问链接的旁边添加一个复选框，就可以创建一种非常简单的已访问链接的样式：a:visited{

background:url(/img/check.gif) no-repeat right middle;

padding-right:20px;

}

## 为链接目标设置样式

链接还可以用来链接到页面的特定部分，实现方法是在href的末尾加一个#字符，然后加上要链接的元素ID。这非常适合指向很长的评论页面中的某一个评论。例如希望连接到这个页面上的第三个评论：

<a href=[http://example.com/story.html#comment3](http://example.com/story.html" \l "comment3)>

a great comment by Simon

</a>

如果页面内容非常多，常常很难看出链接把你转到了哪个元素。为了解决这个问题，CSS3允许使用:target伪类为目标元素设置样式。

.comment:target{

background-color:yellow;

}

如果希望更清楚一些，可以给这个元素设置动画背景图像，图像从黄色逐渐褪色为白色，从而模拟37 Signals等公司常用的黄色褪色技术。

.comment:target{

background-image:url(/img/fade.gif);

}

IE目前还不支持。

## 突出显示不同类型的链接

在许多站点上，很难看出链接是指向本站点上的另一个页面，还是指向另一个站点。我们曾经都有这样的经历：单击一个链接，期望浏览器转到当前站点上的另一个页面，却被带到了别处。为了解决这个问题，许多站点在新窗口中打开外部链接。但是这不是好办法，因为它使用户失去了控制能力，而且这些多余的窗口可能会弄乱用户的桌面。如果没有通知用户出现了新窗口，这还会给屏幕阅读器用户造成问题。

较好的解决方案是让外部链接看起来不一样，让用户选择是离开当前站点还是在新窗口中打开这个链接。为此，可以在外部链接旁边加一个小图标。wiki等站点就是这么做的，而且对于离站链接的图标已经出现了一种约定：一个框架一个箭头。在页面上包含外部链接最容易的方法是在所有外部链接上加一个类，然后将图标作为背景图像应用。

.external{

background:url(/img/externalLink.gif) no-repeat right top;

padding-right:10px;

}

有办法让CSS判断链接是否是外部链接吗？实际上，确实有办法，这就是使用属性选择器。CSS3扩展了它的功能，提供了子字符串匹配。顾名思义，这些选择器允许通过对属性值的一部分和指定的文本进行匹配来寻找元素。这种技术的工作方式是使用[attr^=val]属性选择器寻找以文本http:开头的所有链接：

a[href^=”http:”]{

background:url(/img/externalLink.gif) no-repeat right top;

padding-right:10px;

}

但是他也会选中使用绝对URL而不是相对URL的内部链接。为了避免这个问题，需要重新设置指向自己站点的所有链接，删除他们的外部链接图标。方法是匹配指向自己域名的链接，删除外部链接图标，重新设置右内边距。

a[href^=”http://www.yoursite.com”], a[href^=”http://yoursite.com”]{

background:none;

padding-right:0;

}

还可以扩展这种技术，对邮件链接也进行突出显示。

a[href^=”mailto:”]{

background:url(/img/externalLink.gif) no-repeat right top;

padding-right:10px;

}

甚至可以突出显示非标准协议，比如用小的aim图标突出显示aim即时消息协议

a[href^=”aim:”]{

background:url(/img/im.gif) no-repeat right top;

padding-right:10px;

}

<a href=”aim:goim?screenname=andybudd”>instant message</a>

## 突出显示可下载的文档和提要

这要使用[attr$=val]属性选择器，它寻找以特定值结尾的属性：

a[href$=”.pdf”]{

background:url(/img/pdfLink.gif) no-repeat right top;

padding-right:10px;

}

IE6和更低的版本不支持属性选择器。好在，可以通过在每个元素中添加类，使用JS和DOM实现类似的效果，<http://simonwillison.net/2003/Mar/25/getElementsBySelector/>

可以使用Simon Willison编写的getElementsBySelector函数

## 创建类似按钮的链接

## 纯CSS工具提示

# 附录1：Padding、Border、Margin

## 内边距padding

padding 属性接受长度值或百分比值，但不允许使用负值。

如果您希望所有 h1 元素的各边都有 10 像素的内边距，只需要这样：

h1 {padding: 10px;}

您还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

h1 {padding: 10px 0.25em 2ex 20%;}

也通过使用下面四个单独的属性，分别设置上、右、下、左内边距：

padding-top

padding-right

padding-bottom

padding-left

h1 {

padding-top: 10px;

padding-right: 0.25em;

padding-bottom: 2ex;

padding-left: 20%;

}

内边距的百分比数值

前面提到过，可以为元素的内边距设置百分数值。百分数值是相对于其父元素的 width 计算的，这一点与外边距一样。所以，如果父元素的 width 改变，它们也会改变。

下面这条规则把段落的内边距设置为父元素 width 的 10%：

p {padding: 10%;}

例如：如果一个段落的父元素是 div 元素，那么它的内边距要根据 div 的 width 计算。

<div style="width: 200px;">

<p>This paragragh is contained within a DIV that has a width of 200 pixels.</p>

</div>

注意：上下内边距与左右内边距一致；即上下内边距的百分数会相对于父元素宽度设置，而不是相对于高度。

## 边框border

在 HTML 中，我们使用表格来创建文本周围的边框，但是通过使用 CSS 边框属性，我们可以创建出效果出色的边框，并且可以应用于任何元素。

元素外边距内就是元素的的边框 (border)。元素的边框就是围绕元素内容和内边据的一条或多条线。

每个边框有 3 个方面：宽度、样式，以及颜色。在下面的篇幅，我们会为您详细讲解这三个方面。

边框与背景

CSS 规范指出，边框绘制在“元素的背景之上”。这很重要，因为有些边框是“间断的”（例如，点线边框或虚线框），元素的背景应当出现在边框的可见部分之间。

CSS2 指出背景只延伸到内边距，而不是边框。后来 CSS2.1 进行了更正：元素的背景是内容、内边距和边框区的背景。大多数浏览器都遵循 CSS2.1 定义，不过一些较老的浏览器可能会有不同的表现。

边框的样式

样式是边框最重要的一个方面，这不是因为样式控制着边框的显示（当然，样式确实控制着边框的显示），而是因为如果没有样式，将根本没有边框。

CSS 的 border-style 属性定义了 10 个不同的非 inherit 样式，包括 none。

例如，您可以为把一幅图片的边框定义为 outset，使之看上去像是“凸起按钮”：

a:link img {border-style: outset;}

定义多种样式

您可以为一个边框定义多个样式，例如：

p.aside {border-style: solid dotted dashed double;}

上面这条规则为类名为 aside 的段落定义了四种边框样式：实线上边框、点线右边框、虚线下边框和一个双线左边框。

我们又看到了这里的值采用了 top-right-bottom-left 的顺序，讨论用多个值设置不同内边距时也见过这个顺序。

定义单边样式

如果您希望为元素框的某一个边设置边框样式，而不是设置所有 4 个边的边框样式，可以使用下面的单边边框样式属性：

border-top-style

border-right-style

border-bottom-style

border-left-style

因此这两种方法是等价的：

p {border-style: solid solid solid none;}

p {border-style: solid; border-left-style: none;}

注意：如果要使用第二种方法，必须把单边属性放在简写属性之后。因为如果把单边属性放在 border-style 之前，简写属性的值就会覆盖单边值 none。

边框的宽度

您可以通过 border-width 属性为边框指定宽度。

为边框指定宽度有两种方法：可以指定长度值，比如 2px 或 0.1em；或者使用 3 个关键字之一，它们分别是 thin 、medium（默认值） 和 thick。

注释：CSS 没有定义 3 个关键字的具体宽度，所以一个用户代理可能把 thin 、medium 和 thick 分别设置为等于 5px、3px 和 2px，而另一个用户代理则分别设置为 3px、2px 和 1px。

所以，我们可以这样设置边框的宽度：

p {border-style: solid; border-width: 5px;}

或者：

p {border-style: solid; border-width: thick;}

定义单边宽度

您可以按照 top-right-bottom-left 的顺序设置元素的各边边框：

p {border-style: solid; border-width: 15px 5px 15px 5px;}

上面的例子也可以简写为（这样写法称为值复制）：

p {border-style: solid; border-width: 15px 5px;}

您也可以通过下列属性分别设置边框各边的宽度：

border-top-width

border-right-width

border-bottom-width

border-left-width

因此，下面的规则与上面的例子是等价的：

p {

border-style: solid;

border-top-width: 15px;

border-right-width: 5px;

border-bottom-width: 15px;

border-left-width: 5px;

}

没有边框

在前面的例子中，您已经看到，如果希望显示某种边框，就必须设置边框样式，比如 solid 或 outset。

那么如果把 border-style 设置为 none 会出现什么情况：

p {border-style: none; border-width: 50px;}

尽管边框的宽度是 50px，但是边框样式设置为 none。在这种情况下，不仅边框的样式没有了，其宽度也会变成 0。边框消失了，为什么呢？

这是因为如果边框样式为 none，即边框根本不存在，那么边框就不可能有宽度，因此边框宽度自动设置为 0，而不论您原先定义的是什么？

记住这一点非常重要。事实上，忘记声明边框样式是一个常犯的错误。根据以下规则，所有 h1 元素都不会有任何边框，更不用说 20 像素宽了：

h1 {border-width: 20px;}

由于 border-style 的默认值是 none，如果没有声明样式，就相当于 border-style: none。因此，如果您希望边框出现，就必须声明一个边框样式。

边框的颜色

设置边框颜色非常简单。CSS 使用一个简单的 border-color 属性，它一次可以接受最多 4 个颜色值。

可以使用任何类型的颜色值，例如可以是命名颜色，也可以是十六进制和 RGB 值：

p {

border-style: solid;

border-color: blue rgb(25%,35%,45%) #909090 red;

}

如果颜色值小于 4 个，值复制就会起作用。例如下面的规则声明了段落的上下边框是蓝色，左右边框是红色：

p {

border-style: solid;

border-color: blue red;

}

注释：默认的边框颜色是元素本身的前景色。如果没有为边框声明颜色，它将与元素的文本颜色相同。另一方面，如果元素没有任何文本，假设它是一个表格，其中只包含图像，那么该表的边框颜色就是其父元素的文本颜色（因为 color 可以继承）。这个父元素很可能是 body、div 或另一个 table。

定义单边颜色

还有一些单边边框颜色属性。它们的原理与单边样式和宽度属性相同：

border-top-color

border-right-color

border-bottom-color

border-left-color

要为 h1 元素指定实线黑色边框，而右边框为实线红色，可以这样指定：

h1 {

border-style: solid;

border-color: black;

border-right-color: red;

}

透明边框

我们刚才讲过，如果边框没有样式，就没有宽度。不过有些情况下您可能希望创建一个不可见的边框。

CSS2 引入了边框颜色值 transparent。这个值用于创建有宽度的不可见边框。请看下面的例子：

<a href="#">AAA</a>

<a href="#">BBB</a>

<a href="#">CCC</a>

我们为上面的链接定义了如下样式：

a:link, a:visited {

border-style: solid;

border-width: 5px;

border-color: transparent;

}

a:hover {border-color: gray;}

如需查看以上样式的效果，请点击：TIY。

从某种意义上说，利用 transparent，使用边框就像是额外的内边距一样；此外还有一个好处，就是能在你需要的时候使其可见。这种透明边框相当于内边距，因为元素的背景会延伸到边框区域（如果有可见背景的话）。

重要事项：在 IE7 之前，IE/WIN 没有提供对 transparent 的支持。在以前的版本，IE 会根据元素的 color 值来设置边框颜色。

## 外边距margin

设置外边距的最简单的方法就是使用 margin 属性。

margin 属性接受任何长度单位，可以是像素、英寸、毫米或 em。

margin 可以设置为 auto。更常见的做法是为外边距设置长度值。下面的声明在 h1 元素的各个边上设置了 1/4 英寸宽的空白：

h1 {margin : 0.25in;}

下面的例子为 h1 元素的四个边分别定义了不同的外边距，所使用的长度单位是像素 (px)：

h1 {margin : 10px 0px 15px 5px;}

与内边距的设置相同，这些值的顺序是从上外边距 (top) 开始围着元素顺时针旋转的：

margin: top right bottom left

另外，还可以为 margin 设置一个百分比数值：

p {margin : 10%;}

百分数是相对于父元素的 width 计算的。上面这个例子为 p 元素设置的外边距是其父元素的 width 的 10%。

margin 的默认值是 0，所以如果没有为 margin 声明一个值，就不会出现外边距。但是，在实际中，浏览器对许多元素已经提供了预定的样式，外边距也不例外。例如，在支持 CSS 的浏览器中，外边距会在每个段落元素的上面和下面生成“空行”。因此，如果没有为 p 元素声明外边距，浏览器可能会自己应用一个外边距。当然，只要你特别作了声明，就会覆盖默认样式。

值复制

还记得吗？我们曾经在前两节中提到过值复制。下面我们为您讲解如何使用值复制。

有时，我们会输入一些重复的值：

p {margin: 0.5em 1em 0.5em 1em;}

通过值复制，您可以不必重复地键入这对数字。上面的规则与下面的规则是等价的：

p {margin: 0.5em 1em;}

这两个值可以取代前面 4 个值。这是如何做到的呢？CSS 定义了一些规则，允许为外边距指定少于 4 个值。规则如下：

如果缺少左外边距的值，则使用右外边距的值。

如果缺少下外边距的值，则使用上外边距的值。

如果缺少右外边距的值，则使用上外边距的值。

下图提供了更直观的方法来了解这一点：

换句话说，如果为外边距指定了 3 个值，则第 4 个值（即左外边距）会从第 2 个值（右外边距）复制得到。如果给定了两个值，第 4 个值会从第 2 个值复制得到，第 3 个值（下外边距）会从第 1 个值（上外边距）复制得到。最后一个情况，如果只给定一个值，那么其他 3 个外边距都由这个值（上外边距）复制得到。

利用这个简单的机制，您只需指定必要的值，而不必全部都应用 4 个值，例如：

h1 {margin: 0.25em 1em 0.5em;} /\* 等价于 0.25em 1em 0.5em 1em \*/

h2 {margin: 0.5em 1em;} /\* 等价于 0.5em 1em 0.5em 1em \*/

p {margin: 1px;} /\* 等价于 1px 1px 1px 1px \*/

这种办法有一个小缺点，您最后肯定会遇到这个问题。假设希望把 p 元素的上外边距和左外边距设置为 20 像素，下外边距和右外边距设置为 30 像素。在这种情况下，必须写作：

p {margin: 20px 30px 30px 20px;}

这样才能得到您想要的结果。遗憾的是，在这种情况下，所需值的个数没有办法更少了。

再来看另外一个例子。如果希望除了左外边距以外所有其他外边距都是 auto（左外边距是 20px）：

p {margin: auto auto auto 20px;}

同样的，这样才能得到你想要的效果。问题在于，键入这些 auto 有些麻烦。如果您只是希望控制元素单边上的外边距，请使用单边外边距属性。

单边外边距属性

您可以使用单边外边距属性为元素单边上的外边距设置值。假设您希望把 p 元素的左外边距设置为 20px。不必使用 margin（需要键入很多 auto），而是可以采用以下方法：

p {margin-left: 20px;}

您可以使用下列任何一个属性来只设置相应上的外边距，而不会直接影响所有其他外边距：

margin-top

margin-right

margin-bottom

margin-left

一个规则中可以使用多个这种单边属性，例如：

h2 {

margin-top: 20px;

margin-right: 30px;

margin-bottom: 30px;

margin-left: 20px;

}

当然，对于这种情况，使用 margin 可能更容易一些：

p {margin: 20px 30px 30px 20px;}

不论使用单边属性还是使用 margin，得到的结果都一样。一般来说，如果希望为多个边设置外边距，使用 margin 会更容易一些。不过，从文档显示的角度看，实际上使用哪种方法都不重要，所以应该选择对自己来说更容易的一种方法。

提示和注释

提示：Netscape 和 IE 对 body 标签定义的默认边距（margin）值是 8px。而 Opera 不是这样。相反地，Opera 将内部填充（padding）的默认值定义为 8px，因此如果希望对整个网站的边缘部分进行调整，并将之正确显示于 Opera 中，那么必须对 body 的 padding 进行自定义。

## 轮廓outline

**定义和用法**

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

如果不设置其中的某个值，也不会出问题，比如 outline:solid #ff0000; 也是允许的。

|  |  |
| --- | --- |
| **默认值：** | invert none medium |
| **继承性：** | no |
| **版本：** | CSS2 |
| **JavaScript 语法：** | *object*.style.outline="#0000FF dotted thin" |

**实例**

设置 4 个边框的样式：

p

{

outline:#00FF00 dotted thick;

}

**浏览器支持**

除了 Internet Explorer，所有浏览器都支持 outline 属性。

**可能的值**

|  |  |
| --- | --- |
| **值** | **描述** |
| *outline-color* | 规定边框的颜色。参阅：[outline-color](http://www.w3cschool.cn/pr_outline-color.html" \o "CSS outline-color 属性) 中可能的值。 |
| *outline-style* | 规定边框的样式。参阅：[outline-style](http://www.w3cschool.cn/pr_outline-style.html" \o "CSS outline-style 属性) 中可能的值。 |
| *outline-width* | 规定边框的宽度。参阅：[outline-width](http://www.w3cschool.cn/pr_outline-width.html" \o "CSS outline-width 属性) 中可能的值。 |
| inherit | 规定应该从父元素继承 outline 属性的设置。 |

**outline-width**

**定义和用法**

outline-width 属性设置元素整个轮廓的宽度，只有当轮廓样式不是 none 时，这个宽度才会起作用。如果样式为 none，宽度实际上会重置为 0。不允许设置负长度值。

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。outline 属性设置元素周围的轮廓线。

**注释：**请始终在 outline-width 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

**注释：**轮廓线不会占据空间，也不一定是矩形。

|  |  |
| --- | --- |
| **默认值：** | medium |
| **继承性：** | no |
| **版本：** | CSS2 |
| **JavaScript 语法：** | *object*.style.outlineWidth="thin" |

**实例**

设置点状轮廓的颜色：

p

{

outline-style:dotted;

outline-width:5px;

}

[TIY](http://www.w3cschool.cn/taf74.html?f=csse_outline-width)

**浏览器支持**

所有主流浏览器都支持 outline-width 属性。

只有在规定了 !DOCTYPE 时，Internet Explorer 才支持 outline-width 属性。

**可能的值**

|  |  |
| --- | --- |
| **值** | **描述** |
| thin | 规定细轮廓。 |
| medium | 默认。规定中等的轮廓。 |
| thick | 规定粗的轮廓。 |
| *length* | 允许您规定轮廓粗细的值。 |
| inherit | 规定应该从父元素继承轮廓宽度的设置。 |

**outline-color**

**定义和用法**

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。outline 属性可设置元素周围的轮廓线。

**注释：**请始终在 outline-color 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

**注释：**轮廓线不会占据空间，也不一定是矩形。

outline-color 属性设置一个元素整个轮廓中可见部分的颜色。要记住，轮廓的样式不能是 none，否则轮廓不会出现。

|  |  |
| --- | --- |
| **默认值：** | invert |
| **继承性：** | no |
| **版本：** | CSS2 |
| **JavaScript 语法：** | *object*.style.outlineColor="#0000FF" |

**实例**

设置点状轮廓的颜色：

p

{

outline-style:dotted;

outline-color:#00ff00;

}

[TIY](http://www.w3cschool.cn/t9377.html?f=csse_outline-color)

**浏览器支持**

所有主流浏览器都支持 outline-color 属性。

只有在规定了 !DOCTYPE 时，Internet Explorer 才支持 outline-color 属性。

**可能的值**

|  |  |
| --- | --- |
| **值** | **描述** |
| *color\_name* | 规定颜色值为颜色名称的轮廓颜色（比如 red）。 |
| *hex\_number* | 规定颜色值为十六进制值的轮廓颜色（比如 #ff0000）。 |
| *rgb\_number* | 规定颜色值为 rgb 代码的轮廓颜色（比如 rgb(255,0,0)）。 |
| invert | 默认。执行颜色反转（逆向的颜色）。可使轮廓在不同的背景颜色中都是可见。 |
| inherit | 规定应该从父元素继承轮廓颜色的设置。 |

**outline-style**

**定义和用法**

outline-style 属性用于设置元素的整个轮廓的样式。样式不能是 none，否则轮廓不会出现。

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。outline 属性设置元素周围的轮廓线。

**注释：**请始终在 outline-color 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

**注释：**轮廓线不会占据空间，也不一定是矩形。

|  |  |
| --- | --- |
| **默认值：** | none |
| **继承性：** | no |
| **版本：** | CSS2 |
| **JavaScript 语法：** | *object*.style.outlineStyle="dotted" |

**实例**

设置轮廓的样式：

p

{

outline-style:dotted;

}

[TIY](http://www.w3cschool.cn/t9377.html?f=csse_outline-color)

**浏览器支持**

所有主流浏览器都支持 outline-style 属性。

只有在规定了 !DOCTYPE 时，Internet Explorer 才支持 outline-style 属性。

**可能的值**

|  |  |
| --- | --- |
| **值** | **描述** |
| none | 默认。定义无轮廓。 |
| dotted | 定义点状的轮廓。 |
| dashed | 定义虚线轮廓。 |
| solid | 定义实线轮廓。 |
| double | 定义双线轮廓。双线的宽度等同于 outline-width 的值。 |
| groove | 定义 3D 凹槽轮廓。此效果取决于 outline-color 值。 |
| ridge | 定义 3D 凸槽轮廓。此效果取决于 outline-color 值。 |
| inset | 定义 3D 凹边轮廓。此效果取决于 outline-color 值。 |
| outset | 定义 3D 凸边轮廓。此效果取决于 outline-color 值。 |
| inherit | 规定应该从父元素继承轮廓样式的设置。 |

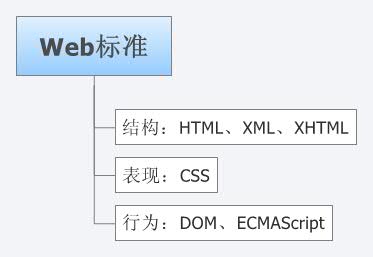
其他补充笔记

工作流程：1.先设计出样图 2.切图 3.写HTML+CSS，做出HTML版的网页 4.写JS脚本

Web标准-W3C和其他标准化组织指定的，包含HTML、XHTML、Javascript、以及CSS。

表现层技术-Web标准、Flash、WPF

RIA-Web标准（Ajax）、Flash、WPF



## CSS布局与Table布局的区别

Table布局使用<table>等标记，CSS布局使用<div>

## 向Web标准过渡

1. 从HTML转向XHTML

XHTML的语法结构与标记更加严谨，提供了更合理的架构。而HTML的设计形式已经不能满足表现与内容分离的网站架构原则。XHTML是XML的过渡型语言，XML才是完全面向结构的设计语言。

XHTML的使用标准有3种

traditional:松散过渡型的XHTML应用，允许用户使用部分HTML标签来编写XHTML文档。推荐使用 这种方式。

strict：严格型的应用方式，XHTML不能使用任何样式表现的标签及属性

frameset：针对框架网页的应用方式，如使用框架式网页则应当使用此方式。

2.使用CSS

* 合理的CSS文件结构：针对大型网站的设计，不妨使用分离的CSS文件来组织样式，比如将字体样式专门使用font.css文件来编写，表单设计则放到form.css文件中，通过合理地组织这些文件，带来后期维护的便利性。
* CSS继承的优势：首先使用本身标签的样式，接着使用父级标签的样式，可重用父级标签的样式。
* 设计跨平台的代码：CSS有浏览器版本问题，所以要减少生僻属性的使用。如想兼容旧版本的浏览器，可留有一定的CSS Hack代码。