

[

CSS 教程

这是第一篇

[CSS 简介 »](#)

通过使用 CSS 来我们可以大大提升网页开发的工作效率！

在我们的 CSS 教程中，您会学到如何使用 CSS 同时控制多重网页的样式和布局。

各章节实例

本 CSS 教程包含了数百个CSS在线实例

通过本站的在线编辑器，你可以在线编辑CSS,并且可以在线查看修改后的效果。

CSS 实例

```
body {
  background-color:#d0e4fe;
}
h1 {
  color:orange;
  text-align:center;
}
p {
  font-family:"Times New Roman";
  font-size:20px;
}
```

[尝试一下 »](#)

点击 "尝试一下" 按钮查看css是如何工作的。

CSS 实例

css 在线150个实例，通过本站编辑器，你可以学习在线查看修改后css的运行效果。

尝试一下！

CSS 参考手册

在自强学堂中你可以找到更完整的CSS属性、选择器的语法，浏览器支持等资料.

[CSS 属性](#)

[CSS 选择器参考手册](#)

[CSS 声音参考手册](#)

[CSS 单位](#)

[CSS 颜色参考手册](#)

]

[

CSS 简介

[«CSS 教程](#)
[CSS 语法»](#)

你需要具备的知识

在继续学习之前，你需要对下面的知识有基本的了解：

- HTML / XHTML

如果你希望首先学习这些项目，请在 [首页](#) 访问相关教程。.

什么是 CSS?

- CSS 指层叠样式表 (Cascading Style Sheets)
- 样式定义如何显示 HTML 元素
- 样式通常存储在样式表中
- 把样式添加到 HTML 4.0 中，是为了解决内容与表现分离的问题
- 外部样式表可以极大提高工作效率
- 外部样式表通常存储在 CSS 文件中
- 多个样式定义可层叠为一

CSS 实例

一个HTML文档可以显示不同的样式: [查看CSS是如何工作的](#)

样式解决了一个很大的问题

HTML 标签原本被设计为用于定义文档内容，如下实例：

```
<h1>这是一个标题</h1>
```

```
<p>这是一个段落.</p>
```

样式表定义如何显示 HTML 元素，就像 HTML 3.2 的字体标签和颜色属性所起的作用那样。样式通常保存在外部的 .css 文件中。通过仅仅编辑一个简单的 CSS 文档，外部样式表使你有能力同时改变站点中所有页面的布局和外观。

为了解决这个问题，万维网联盟（W3C），这个非营利的标准化联盟，肩负起了 HTML 标准化的使命，并在 HTML 4.0 之外创造出样式（Style）。

当代浏览器都支持 CSS。

CSS 样式表极大地提高了工作效率

样式表定义如何显示 HTML 元素

样式表定义如何显示 HTML 元素，就像 HTML 3.2 的字体标签和颜色属性所起的作用那样。样式通常保存在外部的 .css 文件中。通过仅仅编辑一个简单的 CSS 文档，外部样式表使你有能力同时改变站点中所有页面的布局和外观。

]

[

CSS 语法

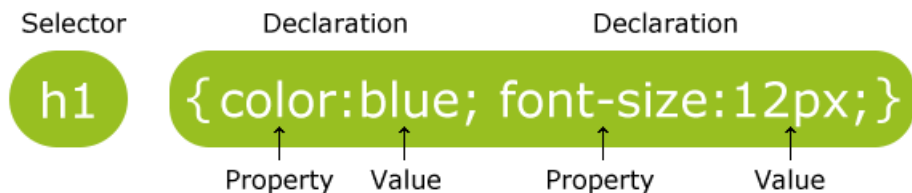
[«CSS 简介](#)
[CSS Id 和 Class选择器 »](#)

实例

- 查看 [实例 1](#)
- 查看 [实例 2](#)

CSS 实例

CSS 规则由两个主要的部分构成：选择器，以及一条或多条声明：



选择器通常是您需要改变样式的 HTML 元素。

每条声明由一个属性和一个值组成。

属性（property）是您希望设置的样式属性（style attribute）。每个属性有一个值。属性和值被冒号分开。

CSS 实例

CSS声明总是以分号(;)结束，声明组以大括号({})括起来:

```
p {color:red;text-align:center;}
```

为了让CSS可读性更强，你可以每行只描述一个属性：

实例

```
p
{
color:red;
text-align:center;
}
```

[尝试一下 »](#)

CSS 注释

注释是用来解释你的代码，并且可以随意编辑它，浏览器会忽略它。

CSS注释以 "/*" 开始, 以 "*/" 介绍, 实例如下:

```
/*This is a comment*/
p
```

```
{
text-align:center;
/*This is another comment*/
color:black;
font-family:arial;
}
]
```

[

CSS Id 和 Class选择器

[«CSS 语法](#)
[CSS 创建 »](#)

id 和 class 选择器

如果你要在HTML元素中设置CSS样式，你需要在元素中设置"id" 和 "class"选择器。

id 选择器

id 选择器可以为标有特定 id 的 HTML 元素指定特定的样式。

HTML元素以id属性来设置id选择器,CSS 中 id 选择器以 "#" 来定义。

以下的样式规则应用于元素属性 id="para1":

实例

```
#para1
{
text-align:center;
color:red;
}
```

[尝试一下 »](#)

💡 ID属性不要以数字开头，数字开头的ID在 Mozilla/Firefox 浏览器中不起作用。

class 选择器

class 选择器用于描述一组元素的样式，class 选择器有别于id选择器，class可以在多个元素中使用。

class 选择器在HTML中以class属性表示, 在 CSS 中，类选择器以一个点"."号显示：

在以下的例子中，所有拥有 center 类的 HTML 元素均为居中。

实例

```
.center {text-align:center;}
```

[尝试一下 »](#)

你也可以指定特定的HTML元素使用class。

在以下实例中, 所有的 p 元素使用 class="center" 让该元素的文本居中：

实例

```
p.center {text-align:center;}
```

[尝试一下 »](#)

💡 类名的第一个字符不能使用数字！它无法在 Mozilla 或 Firefox 中起作用。

[

CSS 创建

[«CSS Id 和 Class选择器](#)
[CSS Backgrounds\(背景\)»](#)

当读到一个样式表时，浏览器会根据它来格式化 HTML 文档。

如何插入样式表

插入样式表的方法有三种：

- 外部样式表
 - 内部样式表
 - 内联样式
-

外部样式表

当样式需要应用于很多页面时，外部样式表将是理想的选择。在使用外部样式表的情况下，你可以通过改变一个文件来改变整个站点的外观。每个页面使用 `<link>` 标签链接到样式表。`<link>` 标签在（文档的）头部：

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

浏览器会从文件 `mystyle.css` 中读到样式声明，并根据它来格式文档。

外部样式表可以在任何文本编辑器中进行编辑。文件不能包含任何的 `html` 标签。样式表应该以 `.css` 扩展名进行保存。下面是一个样式表文件的例子：

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("/static/images/back40.gif");}
```

💡 不要在属性值与单位之间留有空格。假如你使用 `"margin-left: 20 px"` 而不是 `"margin-left: 20px"`，它仅在 IE 6 中有效，但是在 Mozilla/Firefox 或 Netscape 中却无法正常工作。

内部样式表

当单个文档需要特殊的样式时，就应该使用内部样式表。你可以使用 `<style>` 标签在文档头部定义内部样式表，就像这样：

```
<head>
<style>
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

内联样式

由于要将表现和内容混杂在一起，内联样式会损失掉样式表的许多优势。请慎用这种方法，例如当样式仅需要在一个元素上应用一次时。

要使用内联样式，你需要在相关的标签内使用样式（style）属性。Style 属性可以包含任何 CSS 属性。本例展示如何改变段落的颜色和左外边距：

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

多重样式

如果某些属性在不同的样式表中被同样的选择器定义，那么属性值将从更具体的样式表中被继承过来。

例如，外部样式表拥有针对 h3 选择器的三个属性：

```
h3
{
color:red;
text-align:left;
font-size:8pt;
}
```

而内部样式表拥有针对 h3 选择器的两个属性：

```
h3
{
text-align:right;
font-size:20pt;
}
```

假如拥有内部样式表的这个页面同时与外部样式表链接，那么 h3 得到的样式是：

```
color:red;
text-align:right;
font-size:20pt;
```

即颜色属性将被继承于外部样式表，而文字排列（text-alignment）和字体尺寸（font-size）会被内部样式表中的规则取代。

多重样式将层叠为一个

样式表允许以多种方式规定样式信息。样式可以规定在单个的 HTML 元素中，在 HTML 页的头元素中，或在一个外部的 CSS 文件中。甚至可以在同一个 HTML 文档内部引用多个外部样式表。


层叠次序

当同一个 HTML 元素被不止一个样式定义时，会使用哪个样式呢？

一般而言，所有的样式会根据下面的规则层叠于一个新的虚拟样式表中，其中数字 4 拥有最高的优先权。

1. 浏览器缺省设置
2. 外部样式表
3. 内部样式表（位于 <head> 标签内部）
4. 内联样式（在 HTML 元素内部）

因此，内联样式（在 HTML 元素内部）拥有最高的优先权，这意味着它将优先于以下的样式声明： 标签中的样式声明，外部样式表中的样式声明，或者浏览器中的样式声明（缺省值）。

 **提示:**如果你使用了外部文件的样式在 <head>中也定义了该样式，则内部样式表会取代外部文件的样式。

]

[

CSS Backgrounds(背景)

[«CSS 创建
CSS Text\(文本\)»](#)

CSS 背景属性用于定义HTML元素的背景。

CSS 属性定义背影效果:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

背景颜色

background-color 属性定义了元素的背景颜色.

页面的背景颜色使用在body的选择器中:

实例

```
body {background-color:#b0c4de;}
```

[尝试一下 »](#)

CSS中, 颜色值通常以以下方式定义:

- 十六进制 - 如: "#ff0000"
- RGB - 如: "rgb(255,0,0)"
- 颜色名称 - 如: "red"

以下实例中, h1, p, 和 div 元素拥有不同的背景颜色:

实例

```
h1 {background-color:#6495ed;}  
p {background-color:#e0fff;}  
div {background-color:#b0c4de;}
```

[尝试一下 »](#)

背景图像

background-image 属性描述了元素的背景图像.

默认情况下, 背景图像进行平铺重复显示, 以覆盖整个元素实体.

页面背景图片设置实例:

实例

```
body {background-image:url('paper.gif');}
```

[尝试一下 »](#)

下面是一个例子是一个糟糕的文字和背景图像组合。文本可读性差:

实例

```
body {background-image:url('bgdesert.jpg');}
```

[尝试一下 »](#)

背景图像 - 水平或垂直平铺

默认情况下 background-image 属性会在页面的水平或者垂直方向平铺。

一些图像如果在水平方向与垂直方向平铺，这样看起来很不协调，如下所示:

实例

```
body
{
background-image:url('gradient2.png');
}
```

[尝试一下 »](#)

如果图像只在水平方向平铺 (repeat-x), 页面背景会更好些:

实例

```
body
{
background-image:url('gradient2.png');
background-repeat:repeat-x;
}
```

[尝试一下 »](#)

背景图像- 设置定位与不平铺

💡 让背景图像不影响文本的排版

如果你不想让图像平铺，你可以使用 background-repeat 属性:

实例

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
}
```

[尝试一下 »](#)

以上实例中，背景图像与文本显示在同一个位置，为了让页面排版更加合理，不允许文本的阅读，我们可以改变图像的位置。

可以利用 background-position 属性改变图像在背景中的位置:

实例

```
body
{
```

```
background-image:url('img_tree.png');
background-repeat:no-repeat;
background-position:right top;
}
```

[尝试一下 »](#)

背景- 简写属性

在以上实例中我们可以看到页面的背景颜色通过了很多的属性来控制。

为了简化这些属性的代码，我们可以将这些属性合并在一个属性中。

背景颜色的简写属性为 "background"：

实例

```
body {background:#ffffff url('img_tree.png') no-repeat right top;}
```

[尝试一下 »](#)

当使用简写属性时，属性值得顺序为：：

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

以上属性无需全部使用，你可以按照页面的实际需要使用。

这个实例使用了先前介绍的CSS，你可以查看相应实例：[CSS 实例](#)



更多实例

[如何设置固定的背景图像](#)

本例演示如何设置固定的背景图像。图像不会随着页面的其他部分滚动。

CSS 背景属性

Property	描述
background	简写属性，作用是将背景属性设置在一个声明中。
background-attachment	背景图像是否固定或者随着页面的其余部分滚动。
background-color	设置元素的背景颜色。
background-image	把图像设置为背景。
background-position	设置背景图像的起始位置。
background-repeat	设置背景图像是否及如何重复。

]

[

CSS Text(文本)

[«CSS Backgrounds\(背景\)](#)

[CSS Fonts\(字体\)»](#)

text formatting

This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from the ["尝试一下"](#) link.

Text Color

颜色属性被用来设置文字的颜色。

颜色是通过CSS最经常的指定：

- 十六进制值 - 如"#FF0000"
- 一个RGB值 - "RGB (255,0,0) "
- 颜色的名称 - 如"红"

参阅 [CSS 颜色值](#) 查看完整的颜色值。

一个网页的背景颜色是指在主体内的选择：

实例

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

[尝试一下»](#)

💡 对于W3C标准的CSS：如果你定义了颜色属性，你还必须定义背景色属性。

文本的对齐方式

文本排列属性是用来设置文本的水平对齐方式。

文本可居中或对齐到左或右,两端对齐.

当text-align设置为"justify", 每一行被展开为宽度相等, 左, 右外边距是对齐（如杂志和报纸）。

实例

```
h1 {text-align:center;}
p.date {text-align:right;}
p.main {text-align:justify;}
```

[尝试一下»](#)

文本修饰

text-decoration 属性用来设置或删除文本的装饰。

从设计的角度看 text-decoration属性主要是用来删除链接的下划线：

实例

```
a {text-decoration:none;}
```

[尝试一下 »](#)

也可以这样装饰文字：

实例

```
h1 {text-decoration:overline;}  
h2 {text-decoration:line-through;}  
h3 {text-decoration:underline;}
```

[尝试一下 »](#)

💡 我们不建议强调指出不是链接的文本，因为这常常混淆用户。

文本转换

文本转换属性是用来指定在一个文本中的大写和小写字母。

可用于所有字句变成大写或小写字母，或每个单词的首字母大写。

实例

```
p.uppercase {text-transform:uppercase;}  
p.lowercase {text-transform:lowercase;}  
p.capitalize {text-transform:capitalize;}
```

[尝试一下 »](#)

文本缩进

文本缩进属性是用来指定文本的第一行的缩进。

实例

```
p {text-indent:50px;}
```

[尝试一下 »](#)



更多实例

[指定字符之间的空间](#)

这个例子演示了如何增加或减少字符之间的空间。

[指定行与行之间的空间](#)

这个例子演示了如何指定在一个段落中行之间的空间

[设置元素的文本方向](#)

这个例子演示了如何改变元素的文本方向。

[增加单词之间的空白空间](#)

这个例子演示了如何增加一个段落中的单词之间的空白空间。

[在元素内禁用文字环绕](#)

这个例子演示了如何禁用一个元素内的文字环绕。

[垂直对齐图像](#)

这个例子演示了如何设置文本的垂直对齐图像。

[添加文本阴影](#)

这个例子演示了如何设置文本阴影。

所有CSS文本属性。

属性	描述
color	设置文本颜色
direction	设置文本方向。
letter-spacing	设置字符间距
line-height	设置行高
text-align	对齐元素中的文本
text-decoration	向文本添加修饰
text-indent	缩进元素中文本的首行
text-shadow	设置文本阴影
text-transform	控制元素中的字母
unicode-bidi	
vertical-align	设置元素的垂直对齐
white-space	设置元素中空白的处理方式
word-spacing	设置字间距

]

[

CSS Fonts(字体)

[«CSS Text\(文本\)](#)
[CSS 链接\(link\)»](#)

CSS字体属性定义字体，加粗，大小，文字样式。

sserif和sans-serif字体之间的区别



💡 在计算机屏幕上，sans-serif字体被认为是比serif字体容易阅读

CSS字型

在CSS中，有两种类型的字体系列名称：

- 通用字体系列 - 拥有相似外观的字体系统组合（如 "Serif" 或 "Monospace"）
- 特定字体系列 - 一个特定的字体系列（如 "Times" 或 "Courier"）

Generic family	字体系列	说明
Serif	Times New Roman	Serif字体中字符在行的末端拥有额外的装饰
	Georgia	
Sans-serif	Arial	"Sans"是指无 - 这些字体在末端没有额外的装饰
	Verdana	
Monospace	Courier New	所有的等宽字符具有相同的宽度
	Lucida Console	

字体系列

font-family 属性设置文本的字体系列。

font-family 属性应该设置几个字体名称作为一种"后备"机制，如果浏览器不支持第一种字体，他将尝试下一种字体。

注意: 如果字体系列的名称超过一个字，它必须用引号，如Font Family: "宋体"。

多个字体系列是用一个逗号分隔指明：

实例

```
p{font-family:"Times New Roman", Times, serif;}
```

[尝试一下 »](#)

对于较常用的字体组合，看看我们的 [Web安全字体组合](#)。

字体样式

主要是用于指定斜体文字的字体样式属性。

这个属性有三个值：

- 正常 - 正常显示文本
- 斜体 - 以斜体字显示的文字
- 倾斜的文字 - 文字向一边倾斜（和斜体非常类似，但不太支持）

实例

```
p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}
```

[尝试一下 »](#)

字体大小

font-size 属性设置文本的大小。

能否管理文字的大小，在网页设计中是非常重要的。但是，你 cannot 通过调整字体大小使段落看上去像标题，或者使标题看上去像段落。

请务必使用正确的HTML标签，就<h1> - <h6>表示标题和<p>表示段落：

字体大小的值可以是绝对或相对的大小。

绝对大小：

- 设置一个指定大小的文本
- 不允许用户在所有浏览器中改变文本大小
- 确定了输出的物理尺寸时绝对大小很有用

相对大小：

- 相对于周围的元素来设置大小
- 允许用户在浏览器中改变文字大小

💡 如果你不指定一个字体的大小，默认大小和普通文本段落一样，是16像素（16px=1em）。

设置字体大小像素

设置文字的大小与像素，让您完全控制文字大小：

实例

```
h1 {font-size:40px;}
h2 {font-size:30px;}
p {font-size:14px;}
```

[尝试一下 »](#)

上面的例子可以在 Internet Explorer 9, Firefox, Chrome, Opera, 和 Safari 调整文本大小。

注意：以上实例不能在IE9以前的版本运行。

虽然可以通过浏览器的缩放工具调整文本大小，但是，这种调整是整个页面，而不仅仅是文本

用em来设置字体大小

为了避免Internet Explorer 中无法调整文本的问题，许多开发者使用 em 单位代替像素。

em的尺寸单位由W3C建议。

1em和当前字体大小相等。在浏览器中默认的文字大小是16px。

因此，1em的默认大小是16px。可以通过下面这个公式将像素转换为em: $px/16=em$

实例

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */
h2 {font-size:1.875em;} /* 30px/16=1.875em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

[尝试一下 »](#)

在上面的例子，em的文字大小是与前面的例子中像素一样。不过，如果使用 em 单位，则可以在所有浏览器中调整文本大小。

不幸的是，仍然是IE浏览器的问题。调整文本的大小时，会比正常的尺寸更大或更小。

使用百分比和EM组合

在所有浏览器的解决方案中，设置 <body>元素的默认字体大小的是百分比：

实例

```
body {font-size:100%;}
h1 {font-size:2.5em;}
h2 {font-size:1.875em;}
p {font-size:0.875em;}
```

[尝试一下 »](#)

我们的代码非常有效。在所有浏览器中，可以显示相同的文本大小，并允许所有浏览器缩放文本的大小。



更多实例

[设置字体加粗](#)

这个例子演示了如何设置字体的加粗。

[可以设置字体的转变](#)

这个例子演示了如何设置字体的转变。

[在一个声明中的所有字体属性](#)

本例演示如何使用简写属性将字体属性设置在一个声明之内。

所有CSS字体属性

Property	描述
font	在一个声明中设置所有的字体属性
font-family	指定文本的字体系列
font-size	指定文本的字体大小
font-style	指定文本的字体样式

[font-variant](#) 以小型大写字体或者正常字体显示文本。

[font-weight](#) 指定字体的粗细。

]

[

CSS 链接(link)

[«CSS Fonts\(字体\)](#)
[CSS 列表样式\(ul\)»](#)

不同的链接可以有不同的样式。

链接样式

链接的样式，可以用任何CSS属性（如颜色，字体，背景等）。

特别的链接，可以有不同的样式，这取决于他们是什么状态。

这四个链接状态是：

- a:link - 正常，未访问过的链接
- a:visited - 用户已访问过的链接
- a:hover - 当用户鼠标放在链接上时
- a:active - 链接被点击的那一刻

实例

```
a:link {color:#FF0000;} /* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
```

[尝试一下 »](#)

当设置为若干链路状态的样式，也有一些顺序规则：

- a:hover 必须跟在 a:link 和 a:visited后面
- a:active 必须跟在 a:hover后面

常见的链接样式

根据上述链接的颜色变化的例子，看它是在什么状态。

让我们通过一些其他常见的方式转到链接样式：

文本修饰

text-decoration 属性主要用于删除链接中的下划线：

实例

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

[尝试一下 »](#)

背景颜色

背景颜色属性指定链接背景色：

实例

```
a:link {background-color:#B2FF99;}  
a:visited {background-color:#FFFF85;}  
a:hover {background-color:#FF704D;}  
a:active {background-color:#FF704D;}
```

[尝试一下 »](#)



More Examples

[添加不同样式的超链接](#)

这个例子演示了如何为超链接添加其他样式。

[高级 - 创建链接框](#)

这个例子演示了一个更高级的例子，我们结合若干CSS属性显示为方框。

]

[

CSS 列表样式(ul)

[«CSS 链接\(link\)](#)
[CSS Table\(表格\)»](#)

CSS列表属性作用如下：

- 设置不同的列表项标记为有序列表
- 设置不同的列表项标记为无序列表
- 设置列表项标记为图像

列表

在HTML中，有两种类型的列表：

- 无序列表 - 列表项标记用特殊图形（如小黑点、小方框等）
- 有序列表 - 列表项的标记有数字或字母

使用CSS，可以列出进一步的样式，并可用图像作列表项标记。

不同的列表项标记

list-style-type属性指定列表项标记的类型是：

实例

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}

ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

[尝试一下 »](#)

一些值是无序列表，以及有些是有序列表。

作为列表项标记的图像

要指定列表项标记的图像，使用列表样式图像属性：

实例

```
ul
{
list-style-image: url('sqpurple.gif');
}
```

[尝试一下 »](#)

上面的例子在所有浏览器中显示并不相同，IE和Opera显示图像标记比火狐，Chrome和Safari更高一点点。

如果你想在所有的浏览器放置同样的形象标志，就应使用浏览器兼容性解决方案，过程如下

浏览器兼容性解决方案

同样在所有的浏览器，下面的例子会显示的图像标记：

实例

```
ul
{
list-style-type: none;
padding: 0px;
margin: 0px;
}
ul li
{
background-image: url(sqpurple.gif);
background-repeat: no-repeat;
background-position: 0px 5px;
padding-left: 14px;
}
```

[尝试一下 »](#)

例子解释：

- ul:
 - 设置列表样式类型为没有删除列表项标记
 - 设置填充和边距0px（浏览器兼容性）
- ul中所有li:
 - 设置图像的URL，并设置它只显示一次（无重复）
 - 您需要的定位图像位置（左0px和上下5px）
 - 用padding-left属性吧文本置于列表中

列表 - 缩写属性

在单个属性中可以指定所有的列表属性。这就是所谓的缩写属性。

为列表使用缩写属性，列表样式属性设置如下：

实例

```
ul
{
list-style: square url("sqpurple.gif");
}
```

[尝试一下 »](#)

如果使用缩写属性值的顺序是：

- list-style-type
- list-style-position (有关说明， 请参见下面的CSS属性表)
- list-style-image

如果上述值丢失一个， 其余仍在指定的顺序， 就没关系。



更多实例

[所有不同的列表项标记](#)

这个例子演示了所有不同的CSS列表项标记。

所有的CSS列表属性

属性	描述
list-style	简写属性。用于把所有用于列表的属性设置于一个声明中
list-style-image	将图象设置为列表项标志。
list-style-position	设置列表中列表项标志的位置。
list-style-type	设置列表项标志的类型。

]

[

CSS Table(表格)

[«CSS 列表样式\(ul\)](#)
[CSS 框模型 »](#)

使用CSS可以大大提高HTML表格的外观。

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy
North/South	Simon Crowther	UK
Paris spécialités	Marie Bertrand	France
The Big Cheese	Liz Nixon	USA
Vaffeljernet	Palle Ibsen	Denmark

表格边框

指定CSS表格边框，使用border属性。

下面的例子指定了一个表格的Th和TD元素的黑色边框：

实例

```
table, th, td
{
border: 1px solid black;
}
```

[尝试一下 »](#)

请注意，在上面的例子中的表格有双边框。这是因为表和th/ td元素有独立的边界。

为了显示一个表的单个边框，使用 border-collapse属性。

折叠边框

border-collapse 属性设置表格的边框是否被折叠成一个单一的边框或隔开：

实例

```
table
{
border-collapse:collapse;
}
table,th, td
{
border: 1px solid black;
```



```
}
```

[尝试一下 »](#)

表格宽度和高度

Width和height属性定义表格的宽度和高度。

下面的例子是设置100%的宽度，50像素的th元素的高度的表格：

实例

```
table
{
width:100%;
}
th
{
height:50px;
}
```

[尝试一下 »](#)

表格文字对齐

表格中的文本对齐和垂直对齐属性。

text-align属性设置水平对齐方式，像左，右，或中心：

实例

```
td
{
text-align:right;
}
```

[尝试一下 »](#)

垂直对齐属性设置垂直对齐，比如顶部，底部或中间：

实例

```
td
{
height:50px;
vertical-align:bottom;
}
```

[尝试一下 »](#)

表格填充

如果在表的内容中控制空格之间的边框，应使用td和th元素的填充属性：

实例

```
td
{
padding:15px;
}
```

[尝试一下 »](#)

表格颜色

下面的例子指定边框的颜色，和th元素的文本和背景颜色：

实例

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```

[尝试一下 »](#)



更多实例

[制作一个个性表格](#)

这个例子演示了如何创建一个个性的表格。

[设置表格标题的位置](#)

这个例子演示了如何定位表格标题。

]

[

CSS 框模型

[«CSS Table\(表格\)](#)
[CSS Border\(边框\)»](#)

The CSS Box Model

所有HTML元素可以看作盒子，在CSS中，"box model"这一术语是用来设计和布局时使用。

CSS盒模型本质上是一个盒子，封装周围的HTML元素，它包括：边距，边框，填充，和实际内容。

盒模型允许我们在其它元素和周围元素边框之间的空间放置元素。

下面的图片说明了box model:



不同部分的说明：

- **Margin** - 清除边框区域。Margin没有背景颜色，它是完全透明
- **Border** - 边框周围的填充和内容。边框是受到盒子的背景颜色影响
- **Padding** - 清除内容周围的区域。会受到框中填充的背景颜色影响
- **Content** - 盒子的内容，显示文本和图像

为了在所有浏览器中的元素的宽度和高度设置正确的话，你需要知道的盒模型是如何工作的。

元素的宽度和高度

💡**重要:** 当您指定一个CSS元素的宽度和高度属性时，你只是设置内容区域的宽度和高度。要知道，完全大小的元素，你还必须添加填充，边框和边距。.

下面的例子中的元素的总宽度为300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

让我们自己算算：

250px (宽)
+ 20px (left + 右填充)
+ 10px (left + 右边框)
+ 20px (left + 右边距)

= 300px

试想一下，你只有250像素的空间。让我们设置总宽度为250像素的元素：

实例

```
width:220px;  
padding:10px;  
border:5px solid gray;  
margin:0px;
```

[尝试一下»](#)

最终元素的总宽度计算公式是这样的：

总元素的宽度=宽度+左填充+右填充+左边框+右边框+左边距+右边距

元素的总高度最终计算公式是这样的：

总元素的高度=高度+顶部填充+底部填充+上边框+下边框+上边距+下边距

浏览器的兼容性问题

一旦为页面设置了恰当的 DTD，大多数浏览器都会按照上面的图示来呈现内容。然而 IE 5 和 6 的呈现却是不正确的。根据 W3C 的规范，元素内容占据的空间是由 width 属性设置的，而内容周围的 padding 和 border 值是另外计算的。不幸的是，IE5.X 和 6 在怪异模式中使用自己的非标准模型。这些浏览器的 width 属性不是内容的宽度，而是内容、内边距和边框的宽度的总和。

虽然有方法解决这个问题。但是目前最好的解决方案是回避这个问题。也就是，不要给元素添加具有指定宽度的内边距，而是尝试将内边距或外边距添加到元素的父元素和子元素。

IE8 及更早IE版本不支持 填充的宽度和边框的宽度属性设。

解决IE8及更早版本不兼容问题可以在HTML页面声明 <!DOCTYPE html>即可。

]

[

CSS Border(边框)

[«CSS 框模型](#)
[CSS 轮廓 \(outline\) 属性»](#)

CSS 边框属性

CSS边框属性允许你指定一个元素边框的样式和颜色。

边框样式

边框样式属性指定要显示什么样的边界。

💡**border-style**属性用来定义边框的样式

border-style 值：

none: 默认无边框

dotted: dotted:定义一个点线框

dashed: 定义一个虚线框

solid: 定义实线边界

double: 定义两个边界。 两个边界的宽度和border-width的值相同

groove: 定义3D沟槽边界。效果取决于边界的颜色值

ridge: 定义3D脊边界。效果取决于边界的颜色值

inset:定义一个3D的嵌入边框。效果取决于边界的颜色值

outset: 定义一个3D突出边框。 效果取决于边界的颜色值

尝试一下: [设置边框样式](#)

边框宽度

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。

注意：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。

实例

```
p.one
{
```

```
border-style:solid;
border-width:5px;
}
p.two
{
border-style:solid;
border-width:medium;
}
```

[尝试一下 »](#)

边框颜色

border-color属性用于设置边框的颜色。可以设置的颜色：

- name - 指定颜色的名称, 如 "red"
- RGB - 指定 RGB 值, 如 "rgb(255,0,0)"
- Hex - 指定16进制值, 如 "#ff0000"

您还可以设置边框的颜色为"transparent"。

注意： border-color单独使用是不起作用的，必须得先使用border-style来设置边框样式。

实例

```
p.one
{
border-style:solid;
border-color:red;
}
p.two
{
border-style:solid;
border-color:#98bf21;
}
```

[尝试一下 »](#)

边框-单独设置各边

在CSS中，可以指定不同的侧面不同的边框：

实例

```
p
{
border-top-style:dotted;
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
}
```

[尝试一下 »](#)

上面的例子也可以设置一个单一属性：

实例

`border-style:dotted solid;`

[尝试一下 »](#)

`border-style`属性可以有1-4个值：

- **`border-style:dotted solid double dashed;`**
 - 上边框是 `dotted`
 - 右边框是 `solid`
 - 底边框是 `double`
 - 左边框是 `dashed`
- **`border-style:dotted solid double;`**
 - 上边框是 `dotted`
 - 右边框是 `solid`
 - 底边框是 `double`
- **`border-style:dotted solid;`**
 - 上、底边框是 `dotted`
 - 右、左边框是 `solid`
- **`border-style:dotted;`**
 - 四面边框是 `dotted`

上面的例子用了`border-style`。然而，它也可以和`border-width`、`border-color`一起使用。

边框-简写属性

上面的例子用了很多属性来设置边框。

T你也可以在一个属性中设置边框。

你可以在"border"属性中设置：

- `border-width`
- `border-style` (required)
- `border-color`

实例

`border:5px solid red;`

[尝试一下 »](#)



更多实例

[所有边框属性在一个声明之中](#)

本例演示用简写属性来将所有四个边框属性设置于同一声明中。

[设置下边框的样式](#)

本例演示如何设置下边框的样式。

[设置左边框的宽度](#)

本例演示如何设置左边框的宽度。

[设置四个边框的颜色](#)

本例演示如何设置四个边框的颜色。可以设置一到四个颜色。

[设置右边框的颜色](#)

本例演示如何设置右边框的颜色。

CSS 边框属性

属性	描述
border	简写属性，用于把针对四个边的属性设置在一个声明中。
border-style	用于设置元素所有边框的样式，或者单独地为各边设置边框样式。
border-width	简写属性，用于为元素的所有边框设置宽度，或者单独地为各边边框设置宽度。
border-color	简写属性，设置元素的所有边框中可见部分的颜色，或为 4 个边分别设置颜色。
border-bottom	简写属性，用于把下边框的所有属性设置到一个声明中。
border-bottom-color	设置元素的下边框的颜色。
border-bottom-style	设置元素的下边框的样式。
border-bottom-width	设置元素的下边框的宽度。
border-left	简写属性，用于把左边框的所有属性设置到一个声明中。
border-left-color	设置元素的左边框的颜色。
border-left-style	设置元素的左边框的样式。
border-left-width	设置元素的左边框的宽度。
border-right	简写属性，用于把右边框的所有属性设置到一个声明中。
border-right-color	设置元素的右边框的颜色。
border-right-style	设置元素的右边框的样式。
border-right-width	设置元素的右边框的宽度。
border-top	简写属性，用于把上边框的所有属性设置到一个声明中。
border-top-color	设置元素的上边框的颜色。
border-top-style	设置元素的上边框的样式。
border-top-width	设置元素的上边框的宽度。

]

CSS 轮廓（outline）属性

[«CSS Border\(边框\)](#)
[CSS Margin\(外边距\) »](#)

轮廓（outline）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。

轮廓（outline）属性指定了样式，颜色和外边框的宽度。

轮廓（outline）实例

[在元素周围画线](#)

本例演示使用outline属性在元素周围画一条线。.

[设置轮廓的样式](#)

本例演示如何设置轮廓的样式。

[设置轮廓的颜色](#)

本例演示如何设置轮廓的颜色。

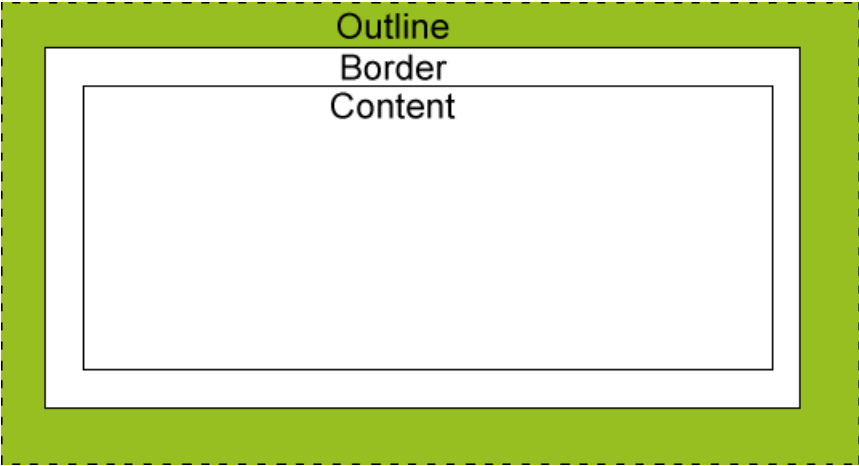
[设置轮廓的宽度](#)

本例演示如何设置轮廓的宽度。

CSS 轮廓（outline）

轮廓（outline）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。

CSS outline 属性规定元素轮廓的样式、颜色和宽度。



所有CSS 轮廓（outline）属性

"CSS" 列中的数字表示哪个CSS版本定义了该属性(CSS1 或者CSS2)。

属性	说明	值	CSS
outline	在一个声明中设置所有的外边框属性	<i>outline-color</i> <i>outline-style</i> <i>outline-width</i> <i>inherit</i>	2
outline-color	设置外边框的颜色	<i>color-name</i> <i>hex-number</i> <i>rgb-number</i>	2

		invert inherit	
outline-style	设置外边框的样式	none dotted dashed solid double groove ridge inset outset inherit	2
outline-width	设置外边框的宽度	thin medium thick <i>length</i> inherit	2

[

CSS Margin(外边距)

[«CSS 轮廓（outline）属性](#)
[CSS Padding（填充）»](#)

CSS Margin(外边距)属性定义元素周围的空间。

Margin

54 230 587 246" data-label="Text">

margin清除周围的元素（外边框）的区域。margin没有背景颜色，是完全透明的

margin可以单独改变元素的上，下，左，右边距。也可以一次改变所有的属性。

可能的值

值	说明
auto	设置浏览器边距。 这样做的结果会依赖于浏览器
<i>length</i>	定义一个固定的margin（使用像素，pt，em等）
%	定义一个使用百分比的边距

💡 Margin可以使用负值，重叠的内容。

在CSS中，它可以指定不同的侧面不同的边距：

实例

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

[尝试一下 »](#)

Margin - 简写属性

所有边距属性的缩写属性是"margin".

实例

```
margin:100px 50px;
```

[尝试一下 »](#)

margin属性可以有一到四个值。</p>
</div>
<div data-label="List-Group">

• **margin:25px 50px 75px 100px;**

◦ 上边距为25px
◦ 右边距为50px
◦ 下边距为75px
◦ 左边距为100px

• **margin:25px 50px 75px;**

◦ 上边距为25px

</div>

- 左右边距为50px
 - 下边距为75px
 - **margin:25px 50px;**
 - 上下边距为25px
 - 左右边距为50px
 - **margin:25px;**
 - 所有的4个边距都是25px
-



更多实例

[文本的上边距设置使用厘米值](#)

这个例子演示了如何设置一个使用厘米值的文本的顶部margin。

[Set 使用百分比值设置文本的下边距](#)

这个例子演示了如何设置使用百分比值的下边距，相对于包含的元素的宽度。

所有的CSS边距属性

属性	描述
margin	简写属性。在一个声明中设置所有外边距属性。
margin-bottom	设置元素的下外边距。
margin-left	设置元素的左外边距。
margin-right	设置元素的右外边距。
margin-top	设置元素的上外边距。

]

[

CSS Padding（填充）

[«CSS Margin\(外边距\)](#)

[CSS 分组和嵌套 »](#)

CSS Padding（填充）属性定义元素边框与元素内容之间的空间。

Padding（填充）

当元素的 Padding（填充）（内边距）被清除时，所"释放"的区域将会受到元素背景颜色的填充。

单独使用填充属性可以改变上下左右的填充。缩写填充属性也可以使用，一旦改变一切都改变。

可能的值

值	说明
---	----

<i>length</i>	定义一个固定的填充(像素, pt, em,等)
---------------	-------------------------

%	使用百分比值定义一个填充
---	--------------

填充- 单边内边距属性

在CSS中，它可以指定不同的侧面不同的填充：

实例

```
padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;
```

[尝试一下 »](#)

填充 - 简写属性

为了缩短代码，它可以在一个属性中指定的所有填充属性。

这就是所谓的缩写属性。所有的填充属性的缩写属性是"padding":

实例

```
padding:25px 50px;
```

[尝试一下 »](#)

Padding属性，可以有一到四个值。

- 上填充为25px
- 右填充为50px
- 下填充为75px
- 左填充为100px

```
padding:25px 50px 75px;
```

- 上填充为25px
- 左右填充为50px
- 下填充为75px

padding:25px 50px;

- 上下填充为25px
- 左右填充为50px

padding:25px;

- 所有的填充都是25px



更多实例

[在一个声明中的所有填充属性](#)

这个例子演示了使用缩写属性设置在一个声明中的所有填充属性，可以有一到四个值。

[设置左部填充](#)

这个例子演示了如何设置元素左填充。

[设置右部填充](#)

这个例子演示了如何设置元素右填充。

[设置上部填充](#)

这个例子演示了如何设置元素上填充。

[设置下部填充](#)

这个例子演示了如何设置元素下填充。

所有的CSS填充属性

属性	说明
padding	使用缩写属性设置在一个声明中的所有填充属性
padding-bottom	设置元素的底部填充
padding-left	设置元素的左部填充
padding-right	设置元素的右部填充
padding-top	设置元素的顶部填充

]

[

CSS 分组和嵌套

[«CSS Padding（填充）](#)
[CSS 尺寸 \(Dimension\) »](#)

Grouping Selectors

在样式表中有很多具有相同样式的元素。

```
h1
{
color:green;
}
h2
{
color:green;
}
p
{
color:green;
}
```

为了尽量减少代码，你可以使用分组选择器。

每个选择器用逗号分隔。

在下面的例子中，我们对以上代码使用分组选择器：

实例

```
h1,h2,p
{
color:green;
}
```

[尝试一下 »](#)

嵌套选择器

它可能适用于选择器内部的选择器的样式。

在下面的例子，为所有p元素指定一个样式，为所有元素指定一个class="marked"的样式，并仅用于class="标记"，类内的p元素指定第三个样式：

实例

```
p
{
color:blue;
text-align:center;
}
.marked
{
background-color:red;
}
```

```
.marked p
{
color:white;
}
```

[尝试一下»](#)

]

[

CSS 尺寸 (Dimension)

[«CSS 分组和嵌套
CSS Display\(显示\) 与 Visibility \(可见性\) »](#)

CSS 尺寸 (Dimension) 属性允许你控制元素的高度和宽度。同样，它允许你增加行间距。



更多实例

[设置元素的高度](#)

这个例子演示了如何设置不同元素的高度。

[使用百分比设置图像的高度](#)

这个例子演示了如何使用百分比值设置元素的高度。

[使用像素值来设置元素的宽度](#)

本例演示如何使用像素值来设置元素的宽度。

[设置元素的最大高度](#)

此示例演示如何设置元素的最大高度。

[使用百分比来设置元素的最大宽度](#)

本例演示如何使用百分比值来设置元素的最大宽度。

[设置元素的最低高度](#)

此示例演示如何设置元素的最小高度。

[使用像素值设置元素的最小宽度](#)

这个例子演示了如何使用像素值设置元素的最小宽度。

所有CSS 尺寸 (Dimension)属性

"CSS" 列中的数字表示哪个CSS版本定义了该属性 (CSS1 or CSS2)。

属性	描述
height	设置元素的高度。
line-height	设置行高。
max-height	设置元素的最大高度。
max-width	设置元素的最大宽度。
min-height	设置元素的最小高度。
min-width	设置元素的最小宽度。
width	设置元素的宽度。

]

[

CSS Display(显示) 与 Visibility（可见性）

[«CSS 尺寸 \(Dimension\)](#)
[CSS Positioning\(定位\)»](#)

如何设置一个元素是否显示，visibility属性指定一个元素应可见或隐藏。



隐藏元素 - display:none或visibility:hidden

隐藏一个元素可以通过display属性设置为"none"或"hidden"属性的可见性。但是，请注意这两种方法产生不同的结果
visibility:hidden可以让您隐藏某个元素，但它仍然需要像以前一样的空间。该元素将被隐藏，但仍然会影响布局。

实例

```
h1.hidden {visibility:hidden;}
```

[尝试一下 »](#)

display:没有隐藏的元素，它不会占用任何空间。元素将被隐藏，但页面会显示：

实例

```
h1.hidden {display:none;}
```

[尝试一下 »](#)

CSS Display - 块和内联元素

块元素是一个元素，占用了全部宽度，在前后都是换行符。

块元素的例子：

- <h1>
- <p>
- <div>

内联元素只需要必要的宽度，不强制换行。

内联元素的例子：

-
- <a>

如何改变一个元素显示

可以更改内联元素和块元素，反之亦然，可以使页面看起来是以一种特定的方式组合，并仍然遵循web标准。

下面的示例显示为内联元素的列表项：

实例

```
li {display:inline;}
```

[尝试一下 »](#)

下面的示例是把span元素作为块元素

实例

```
span {display:block;}
```

[尝试一下 »](#)

注意： 变更元素的显示类型看该元素是如何显示，它是什么样的元素。例如：一个内联元素设置为display: block是不允许有它内部的嵌套块元素。.



更多实例

[如何显示元素的内联元素。](#)

这个例子演示了如何显示一个元素的内联元素。

[如何显示元素的块元素。](#)

这个例子演示了如何显示一个元素的块元素。

[如何使用一个表的collapse属性](#)

这个例子演示里如何使用表的collapse属性

]

[

CSS Positioning(定位)

[«CSS Display\(显示\)与 Visibility \(可见性\)](#)
[CSS Float\(浮动\)»](#)

决定显示在前面的元素！

定位有时很棘手！

元素可以重叠！

Positioning(定位)

CSS定位属性允许你为一个元素定位。它也可以将一个元素放在另一个元素后面，并指定一个元素的内容太大时，应该发生什么。

元素可以使用的顶部，底部，左侧和右侧属性定位。然而，这些属性无法工作，除非是先设定position属性。他们也有不同的工作方式，这取决于定位方法。

有四种不同的定位方法。

Static 定位

HTML元素的默认值，即没有定位，元素出现在正常的流中。

静态定位的元素不会受到top, bottom, left, right影响。

Fixed 定位

元素的位置相对于浏览器窗口是固定位置。

即使窗口是滚动的它也不会移动：

实例

```
p.pos_fixed
{
position:fixed;
top:30px;
right:5px;
}
```

[尝试一下 »](#)

注意： Fixed 定位在 IE7 和 IE8 下需要描述 !DOCTYPE 才能支持。

Fixed定位使元素的位置与文档流无关，因此不占据空间。

Fixed定位的元素和其他元素重叠。

Relative 定位

相对定位元素的定位是相对其正常位置。

实例

```
h2.pos_left
{
position: relative;
left: -20px;
}
h2.pos_right
{
position: relative;
left: 20px;
}
```

[尝试一下 »](#)

可以移动的相对定位元素的内容和相互重叠的元素，它原本所占的空间不会改变。

实例

```
h2.pos_top
{
position: relative;
top: -50px;
}
```

[尝试一下 »](#)

相对定位元素经常被用来作为绝对定位元素的容器块。

Absolute 定位

绝对定位的元素的位置相对于最近的已定位父元素，如果元素没有已定位的父元素，那么它的位置相对于<html>:

实例

```
h2
{
position: absolute;
left: 100px;
top: 150px;
}
```

[尝试一下 »](#)

Absolutely定位使元素的位置与文档流无关，因此不占据空间。

Absolutely定位的元素和其他元素重叠。

重叠的元素

元素的定位与文档流无关，所以它们可以覆盖页面上的其它元素

z-index属性指定了一个元素的堆叠顺序（哪个元素应该放在前面，或后面）

一个元素可以有正数或负数的堆叠顺序：

实例

```
img
{
```

```
position:absolute;
left:0px;
top:0px;
z-index:-1;
}
```

[尝试一下 »](#)

具有更高堆叠顺序的元素总是在较低的堆叠顺序元素的前面。

注意： 如果两个定位元素重叠，没有指定z - index，最后定位在HTML代码中的元素将被显示在最前面。



更多实例

[裁剪元素的外形](#)

此示例演示如何设置元素的外形。该元素被剪裁成这种形状，并显示出来。

[如何使用滚动条来显示元素内溢出的内容](#)

这个例子演示了overflow属性创建一个滚动条，当一个元素的内容在指定的区域过大时如何设置以适应。

[如何设置浏览器自动溢出处理](#)

这个例子演示了如何设置浏览器来自动处理溢出。

[更改光标](#)

这个例子演示了如何改变光标。

所有的CSS定位属性

"CSS" 列中的数字表示哪个CSS(CSS1 或者CSS2)版本定义了该属性。

属性	说明	值	CSS
bottom	定义了定位元素下外边距边界与其包含块下边界之间的偏移。	auto <i>length</i> % inherit	2
clip	剪辑一个绝对定位的元素	<i>shape</i> auto inherit	2
cursor	显示光标移动到指定的类型	<i>url</i> auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text	2

left	定义了定位元素左外边距边界与其包含块左边界之间的偏移。	wait help auto <i>length</i> % inherit	2
overflow	设置当元素的内容溢出其区域时发生的事情。	auto hidden scroll visible inherit	2
position	指定元素的定位类型	absolute fixed relative static inherit	2
right	定义了定位元素右外边距边界与其包含块右边界之间的偏移。	auto <i>length</i> % inherit	2
top	定义了一个定位元素的上外边距边界与其包含块上边界之间的偏移。	auto <i>length</i> % inherit	2
z-index	设置元素的堆叠顺序	<i>number</i> auto inherit	2

]

[

CSS Float(浮动)

[«CSS Positioning\(定位\)](#)
[CSS 水平对齐\(Horizontal Align\) »](#)

什么是CSS Float(浮动)?



CSS的Float(浮动)，元素可以围绕其他元素向左或向右被推动

Float(浮动)，往往是用于图像，但它在布局时一样非常有用。

元素怎样浮动

元素的水平方向浮动意味着元素只能左右移动而不能上下移动。

一个浮动元素会尽量向左或右。通常，这意味着尽所有的可能在所有包含元素的左侧或右侧的。

浮动元素之后的元素将围绕它。

浮动元素之前的元素将不会受到影响。

如果图像是右浮动，下面的文本流将环绕在它左边：

实例

```
img
{
float:right;
}
```

[尝试一下 »](#)

彼此相邻的浮动元素

如果你把几个浮动的元素放到一起，如果有空间的话，它们将彼此相邻。

在这里，我们已经对图片廊使用float属性：

实例

```
.thumbnail
{
float:left;
width:110px;
height:90px;
margin:5px;
}
```

[尝试一下 »](#)

清除浮动 - 使用clear

元素浮动之后，周围的元素会重新排列，为了避免这种情况，使用clear属性

。clear属性指定其他元素双方都不能使用元素的浮动功能。

使用clear属性往文本中添加图片廊：

实例

```
.text_line
{
clear:both;
}
```

[尝试一下 »](#)



更多实例

[为图像添加边框和边距并浮动到段落的左侧](#)

让我们为图像添加边框和边距并浮动到段落的左侧

[标题和图片向右侧浮动](#)

让标题和图片向右侧浮动。

[让段落的第一个字母浮动到左侧](#)

改变样式，让段落的第一个字母浮动到左侧。

[创建一个没有表格的网页](#)

使用float创建一个网页页眉、页脚、左边的内容和主要内容。

所有CSS浮动属性

"CSS" 列中的数字表示不同的CSS版本 (CSS1 or CSS2)定义了该属性。

属性	描述	值	CSS
clear	指定不允许元素周围有浮动元素	left	1
		right	
		both	
		none	
		inherit	
float	指定Box是否可以浮动	left	1
		right	
		none	
		inherit	

]

[

CSS 水平对齐(Horizontal Align)

[«CSS Float\(浮动\)
CSS 伪类»](#)

在CSS中，有几个属性用于元素水平对齐。

块元素对齐

块元素是一个元素，占用了全宽，前后都是换行符。

块元素的例子：

- <h1>
- <p>
- <div>

文本对齐，请参阅 [CSS文本](#) 章节。

在这一章中，我们会告诉你块元素如何水平对齐布局。

中心对齐,使用margin属性

块元素可以把左，右页边距设置为"自动"对齐。

Note: 在IE8中使用margin:auto属性无法正常工作，除非声明 **!DOCTYPE**

margin属性可任意拆分为左，右页边距设置自动指定，结果都是出现居中元素：

实例

```
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
}
```

[尝试一下»](#)

提示: 如果宽度是100%，对齐是没有效果的。

注意: IE5中块元素有一个margin处理BUG。为了使上述例子能工作，在IE5中，需要添加一些额外的代码。 [实例](#)

使用position属性设置左，右对齐

元素对齐的方法之一是使用绝对定位：

实例

```
.right
```

```
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

[尝试一下 »](#)

注意：绝对定位与文档流无关，所以它们可以覆盖页面上的其它元素。

Crossbrowser 兼容性问题

元素的填充，始终是一个好主意。这是为了避免在不同的浏览器中的可视化差异。

IE8和早期有一个问题，当使用position属性时。如果一个容器元素（在本例中<div class="container">）指定的宽度，!DOCTYPE声明是缺失，IE8和早期版本会在右边增添17px的margin。这似乎是一个滚动的预留空间。使用position属性时始终设置在DOCTYPE声明中！

实例

```
body
{
margin:0;
padding:0;
}
.container
{
position:relative;
width:100%;
}
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

[尝试一下 »](#)

使用float属性设置左，右对齐

使用float属性是对齐元素的方法之一：

实例

```
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

[尝试一下 »](#)

Crossbrowser 兼容性问题

类似这样的元素对齐时，预先确定margin和元素的填充，始终是一个好主意。这是为了避免在不同的浏览器中的可视化差异。

IE8和早期有一个问题，当使用float属性时。如果一个容器元素（在本例中<div class="container">）指定的宽度，!DOCTYPE声明是缺失，IE8和早期版本会在右边增添17px的margin。这似乎是一个滚动的预留空间。使用float属性时始终设置在DOCTYPE声明中！

实例

```
body
{
margin:0;
padding:0;
}
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

[尝试一下»](#)

]

[

CSS 伪类

[«CSS 水平对齐\(Horizontal Align\)](#)
[CSS 伪元素»](#)

CSS伪类是用来添加一些选择器的特殊效果。

语法

伪类的语法：

```
selector:pseudo-class {property:value;}
```

CSS类也可以使用伪类：

```
selector.class:pseudo-class {property:value;}
```

anchor伪类

在支持 CSS 的浏览器中，链接的不同状态都可以以不同的方式显示

实例

```
a:link {color:#FF0000;} /* 未访问的链接 */  
a:visited {color:#00FF00;} /* 已访问的链接 */  
a:hover {color:#FF00FF;} /* 鼠标划过链接 */  
a:active {color:#0000FF;} /* 已选中的链接 */
```

[尝试一下»](#)

注意： 在CSS定义中，a:hover 必须被置于 a:link 和 a:visited 之后，才是有效的。

注意： 在 CSS 定义中，a:active 必须被置于 a:hover 之后，才是有效的。

注意： 伪类的名称不区分大小写。

伪类和CSS类

伪类可以与 CSS 类配合使用：

```
a.red:visited {color:#FF0000;}
```

```
<a class="red" href="css-syntax.html">CSS Syntax</a>
```

如果在上面的例子的链接已被访问，它会显示为红色。

CSS - :first-child伪类

您可以使用 :first-child 伪类来选择元素的第一个子元素

注意： 在IE8的之前版本必须声明<!DOCTYPE>，这样 :first-child 才能生效。

匹配第一个 <p> 元素

在下面的例子中，选择器匹配作为任何元素的第一个子元素的 <p> 元素：

实例

```
<html>
<head>
<style>
p:first-child
{
color:blue;
}
</style>
</head>

<body>
<p>I am a strong man.</p>
<p>I am a strong man.</p>
</body>
</html>
```

[尝试一下 »](#)

匹配所有<p> 元素中的第一个 <i> 元素

在下面的例子中，选择相匹配的所有<p>元素的第一 <i>元素：

实例

```
<html>
<head>
<style>
p > i:first-child
{
color:blue;
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

[尝试一下 »](#)

实例

```
<html>
<head>
<style>
p:first-child i
{
color:blue;
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
```

```
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

[尝试一下 »](#)

CSS - :lang 伪类

:lang 伪类使你有能力为不同的语言定义特殊的规则

注意：IE8必须声明[<!DOCTYPE>](#)才能支持:lang伪类。

在下面的例子中，:lang 类为属性值为 no 的q元素定义引号的类型：

实例

```
<html>
<head>
<style>
q:lang(no) {quotes: "~" "~";}
</style>
</head>

<body>
<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
</body>
</html>
```

[尝试一下 »](#)



更多实例

[为超链接添加不同样式](#)

这个例子演示了如何为超链接添加其他样式。

[使用 .focus](#)

这个例子演示了如何使用 :focus伪类。

所有CSS伪类/元素

选择器	示例	示例说明
:link	a:link	选择所有未访问链接
:visited	a:visited	选择所有访问过的链接
:active	a:active	选择正在活动链接
:hover	a:hover	把鼠标放在链接上的状态
:focus	input:focus	选择元素输入后具有焦点
:first-letter	p:first-letter	选择每个<p> 元素的第一个字母
:first-line	p:first-line	选择每个<p> 元素的第一行
:first-child	p:first-child	选择器匹配属于任意元素的第一个子元素的 < p> 元素
:before	p:before	Insert content before every <p> element
:after	p:after	在每个<p>元素之前插入内容
:lang(language)	p:lang(it)	为<p>元素的lang属性选择一个开始值

[

CSS 伪元素

[«CSS 伪类](#)
[CSS 导航栏»](#)

CSS伪元素是用来添加一些选择器的特殊效果。

语法

伪元素的语法：

```
selector:pseudo-element {property:value;}
```

CSS类也可以使用伪元素：

```
selector.class:pseudo-element {property:value;}
```

:first-line 伪元素

"first-line" 伪元素用于向文本的首行设置特殊样式。

在下面的例子中，浏览器会根据 "first-line" 伪元素中的样式对 p 元素的第一行文本进行格式化：

实例

```
p:first-line
{
color:#ff0000;
font-variant:small-caps;
}
```

[尝试一下»](#)

注意： "first-line" 伪元素只能用于块级元素。

注意： 下面的属性可应用于 "first-line" 伪元素：

- font properties
 - color properties
 - background properties
 - word-spacing
 - letter-spacing
 - text-decoration
 - vertical-align
 - text-transform
 - line-height
 - clear
-

:first-letter 伪元素

"first-letter" 伪元素用于向文本的首字母设置特殊样式：

实例

```
p:first-letter
```

```
{
color:#ff0000;
font-size:xx-large;
}
```

[尝试一下 »](#)

注意： "first-letter" 伪元素只能用于块级元素。

注意： 下面的属性可应用于 "first-letter" 伪元素：

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

伪元素和CSS类

伪元素可以结合CSS类：

```
p.article:first-letter {color:#ff0000;}
```

```
<p class="article">A paragraph in an article</p>
```

上面的例子会使所有 class 为 article 的段落的首字母变为红色。

Multiple Pseudo-elements

可以结合多个伪元素来使用。

在下面的例子中，段落的第一个字母将显示为红色，其字体大小为 xx-large。第一行中的其余文本将为蓝色，并以小型大写字母显示。

段落中的其余文本将以默认字体大小和颜色来显示：

实例

```
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
p:first-line
{
color:#0000ff;
font-variant:small-caps;
}
```

[尝试一下 »](#)

CSS - :before 伪元素

"before" 伪元素可以在元素的内容前面插入新内容。

下面的例子在每个 <h1>元素前面插入一幅图片：

实例

```
h1:before
{
content:url(smiley.gif);
}
```

[尝试一下 »](#)

CSS - :after 伪元素

"after" 伪元素可以在元素的内容之后插入新内容。

下面的例子在每个 <h1> 元素后面插入一幅图片：

实例

```
h1:after
{
content:url(smiley.gif);
}
```

[尝试一下 »](#)

所有CSS伪类/元素

选择器	示例	示例说明
:link	a:link	选择所有未访问链接
:visited	a:visited	选择所有访问过的链接
:active	a:active	选择正在活动链接
:hover	a:hover	把鼠标放在链接上的状态
:focus	input:focus	选择元素输入后具有焦点
:first-letter	p:first-letter	选择每个<p> 元素的第一个字母
:first-line	p:first-line	选择每个<p> 元素的第一行
:first-child	p:first-child	选择器匹配属于任意元素的第一个子元素的 <p> 元素
:before	p:before	Insert content before every <p> element
:after	p:after	在每个<p>元素之前插入内容
:lang(language)	p:lang(it)	为<p>元素的lang属性选择一个开始值

[

CSS 导航栏

[«CSS 伪元素](#)
[CSS 图片廊»](#)

实例：导航栏

- [Home](#)
- [News](#)
- [Articles](#)
- [Forum](#)
- [Contact](#)
- [About](#)

导航栏

熟练使用导航栏，对于任何网站都非常重要。

使用CSS你可以转换成好看的导航栏而不是枯燥的HTML菜单。

导航栏=链接列表

作为标准的HTML基础一个导航栏是必须的

。在我们的例子中我们将建立一个标准的HTML列表导航栏。

导航条基本上是一个链接列表，所以使用 `` 和 `` 元素非常有意义：

实例

```
<ul>
<li><a href="default.asp">Home</a></li>
<li><a href="news.asp">News</a></li>
<li><a href="contact.asp">Contact</a></li>
<li><a href="about.asp">About</a></li>
</ul>
```

[尝试一下»](#)

现在，让我们从列表中删除边距和填充：

实例

```
ul
{
list-style-type:none;
margin:0;
padding:0;
}
```

[尝试一下»](#)

例子解析：

- `list-style-type:none` - 移除列表前小标志。一个导航栏并不需要列表标记

- 移除浏览器的默认设置将边距和填充设置为0

上面的例子中的代码是垂直和水平导航栏使用的标准代码。

垂直导航栏

上面的代码，我们只需要 `<a>` 元素的样式，建立一个垂直的导航栏：

实例

```
a
{
display:block;
width:60px;
}
```

[尝试一下 »](#)

示例说明：

- `display:block` - 显示块元素的链接，让整体变为可点击链接区域（不只是文本），它允许我们指定宽度
- `width:60px` - 块元素默认情况下是最大宽度。我们要指定一个60像素的宽度

提示：查看 [完全样式的垂直导航栏的示例](#)。

注意：请务必指定 `<a>` 元素在垂直导航栏的的宽度。如果省略宽度，IE6可能产生意想不到的效果。

水平导航栏

有两种方法创建横向导航栏。使用**内联**或**浮动**的列表项。

这两种方法都很好，但如果你想链接到具有相同的大小，你必须使用浮动的方法。

内嵌列表项

建立一个横向导航栏的方法之一是指定

- 元素， 上述代码是标准的内嵌：

实例

```
li
{
display:inline;
}
```

[尝试一下 »](#)

实例解析：

- `display:inline;` - 默认情况下，`` 元素是块元素。在这里，我们删除换行符之前和之后每个列表项，以显示一行。

提示：查看 [完全样式的水平导航栏的示例](#)。

浮动列表项

在上面的例子中链接有不同的宽度。

对于所有的链接宽度相等，浮动 `` 元素，并指定为 `<a>` 元素的宽度：

实例

```
li
{
float:left;
}
a
{
display:block;
width:60px;
}
```

[尝试一下»](#)

实例解析：

- float:left - 使用浮动块元素的幻灯片彼此相邻
- display:block - 显示块元素的链接，让整体变为可点击链接区域（不只是文本），它允许我们指定宽度
- width:60px - 块元素默认情况下是最大宽度。我们要指定一个60像素的宽度

Tip:查看 [完全样式的横向导航栏的示例](#)..

]

[

CSS 图片廊

[«CSS 导航栏](#)
[CSS 图像透明/不透明 »](#)

以下是使用CSS创建图片廊:



图片廊

T以下是使用CSS创建图片廊:

实例

```
<html>
<head>
<style>
div.ing
{
margin:2px;
border:1px solid #0000ff;
height:auto;
width:auto;
float:left;
text-align:center;
}
div.ing img
{
display:inline;
margin:3px;
border:1px solid #ffff;
}
div.ing a:hover img
{
border:1px solid #0000ff;
}
div.desc
{
text-align:center;
font-weight:normal;
width:120px;
margin:2px;
}
</style>
</head>
<body>

<div class="ing">
<a target="_blank" href="klematis_big.htm">

</a>
<div class="desc">Add a description of the image here</div>
</div>
<div class="ing">
<a target="_blank" href="klematis2_big.htm">

</a>
```

```
<div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis3_big.htm">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis4_big.htm">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>
```

[尝试一下 »](#)

]

[

CSS 图像透明/不透明

[«CSS 图片廊](#)
[CSS 图像拼合技术»](#)

使用CSS很容易创建透明的图像。

注意：CSS Opacity属性是W3C的CSS3建议的一部分。



更多实例

[创建透明图像 - 悬停效果](#)

[创建一个具有文本的拥有背景图像的透明框](#)

实例1 - 创建一个透明图像

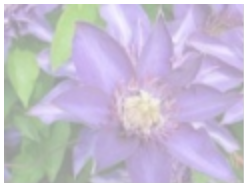
CSS3中属性的透明度是 **opacity**.

首先，我们将向您展示如何用CSS创建一个透明图像。

正常的图像



相同的图像带有透明度：



看看下面的CSS：

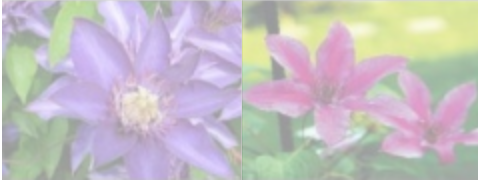
```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
```

IE9, Firefox, Chrome, Opera, 和Safari浏览器使用透明度属性可以将图像变的不透明。Opacity属性值从0.0 - 1.0。值越小，使得元素更加透明。

IE8和早期版本使用滤镜：alpha（opacity= x）。x可以采取的值是从0 - 100。较低的值，使得元素更加透明。

实例2 - 图像的透明度 - 悬停效果

将鼠标移到图像上:



CSS样式:

```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
img:hover
{
opacity:1.0;
filter:alpha(opacity=100); /* For IE8 and earlier */
}
```

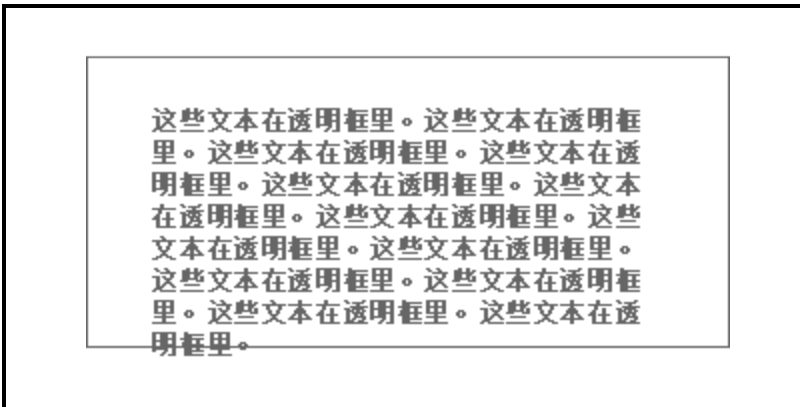
第一个CSS块是和例1中的代码类似。此外，我们还增加了当用户将鼠标悬停在其中一个图像上时发生什么。在这种情况下，当用户将鼠标悬停在图像上时，我们希望图片是清晰的。

此CSS是: **opacity=1**.

IE8和更早版本使用: **filter:alpha(opacity=100)**.

当鼠标指针远离图像时，图像将重新具有透明度。

实例3 - 透明的盒子中的文字



源代码如下:

```
<html>
<head>
<style>
div.background
{
width:500px;
height:250px;
background:url(klematis.jpg) repeat;
border:2px solid black;
}
div.transbox
```

```

{
width:400px;
height:180px;
margin:30px 50px;
background-color:#ffffff;
border:1px solid black;
opacity:0.6;
filter:alpha(opacity=60); /* For IE8 and earlier */
}
div.transbox p
{
margin:30px 40px;
font-weight:bold;
color:#000000;
}
</style>
</head>

<body>

<div class="background">
<div class="transbox">
<p>This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>
</div>

</body>
</html>

```

首先，我们创建一个固定的高度和宽度的div元素，带有一个背景图片和边框。然后我们在第一个div内部创建一个较小的div元素。这个div也有一个固定的宽度，背景颜色，边框 - 而且它是透明的。透明的div里面，我们在P元素内部添加一些文本。

]

[

CSS 图像拼合技术

[«CSS 图像透明/不透明
CSS 媒体类型»](#)

图像拼合

图像拼合就是单个图像的集合。

有许多图像的网页可能需要很长的时间来加载和生成多个服务器的请求。

使用图像拼合会降低服务器的请求数量，并节省带宽。

图像拼合 - 简单实例

与其使用三个独立的图像，不如我们使用这种单个图像（"img_navsprites.gif"）：



有了CSS，我们可以只显示我们需要的图像的一部分。

在下面的例子CSS指定显示 "img_navsprites.gif" 的图像的一部分：

实例

```
img.home
{
width:46px;
height:44px;
background.url(img_navsprites.gif) 0 0;
}
```

[尝试一下»](#)

实例解析：

- - 因为不能为空,src属性只定义了一个小的透明图像。显示的图像将是我们在CSS中指定的背景图像
- 宽度：46px;高度：44px; - 定义我们使用的那部分图像
- background.url(img_navsprites.gif) 0 0; - 定义背景图像和它的位置（左0px，顶部0px）

这是使用图像拼合最简单的方法，现在我们使用链接和悬停效果。

图像拼合 - 创建一个导航列表

我们想使用拼合图像 ("img_navsprites.gif")，以创建一个导航列表。

我们将使用一个HTML列表，因为它可以链接，同时还支持背景图像：

实例

```
#navlist{position:relative;}
#navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a{height:44px;display:block;}
```

```
#home{left:0px;width:46px;}
#home{background:url('img_navsprites.gif') 0 0;}

#prev{left:63px;width:43px;}
#prev{background:url('img_navsprites.gif') -47px 0;}

#next{left:129px;width:43px;}
#next{background:url('img_navsprites.gif') -91px 0;}
```

[尝试一下 »](#)

实例解析：

- #navlist{position:relative;} - 位置设置相对定位，让里面的绝对定位
- #navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;} - margin和padding设置为0，列表样式被删除，所有列表项是绝对定位
- #navlist li, #navlist a{height:44px;display:block;} - 所有图像的高度是44px

现在开始每个具体部分的定位和样式：

- #home{left:0px;width:46px;} - 定位到最左边的方式，以及图像的宽度是46px
- #home{background:url('img_navsprites.gif') 0 0;} - 定义背景图像和它的位置（左0px，顶部0px）
- #prev{left:63px;width:43px;} - 右侧定位63px（#home宽46px+项目之间的一些多余的空间），宽度为43px。
- #prev{background:url('img_navsprites.gif') -47px 0;} - 定义背景图像右侧47px（#home宽46px+分隔线的1px）
- #next{left:129px;width:43px;} - 右边定位129px（#prev 63px + #prev宽是43px + 剩余的空间），宽度是43px。
- #next{background:url('img_navsprites.gif') no-repeat -91px 0;} - 定义背景图像右边91px（#home 46px+1px的分割线+#prev宽43px+1px的分隔线）

图像拼合s - 悬停效果

现在，我们希望我们的导航列表中添加一个悬停效果。



:hover 选择器用于鼠标悬停在元素上的显示的效果

提示：:hover 选择器可以运用于有元素。

我们的新图像 ("img_navsprites_hover.gif") 包含三个导航图像和三幅图像：



因为这是一个单一的图像，而不是6个单独的图像文件，当用户停留在图像上不会有延迟加载。

我们添加悬停效果只添加三行代码：

实例

```
#home a:hover{background: url('img_navsprites_hover.gif') 0 -45px;}
#prev a:hover{background: url('img_navsprites_hover.gif') -47px -45px;}
#next a:hover{background: url('img_navsprites_hover.gif') -91px -45px;}
```

[尝试一下 »](#)

实例解析：

- 由于该列表项包含一个链接，我们可以使用：hover伪类
- #home a:hover{background: transparent url('img_navsprites_hover.gif') 0 -45px;} - 对于所有三个悬停图像，我们指定相同的背景位置，只是每个再向下45px

[

CSS 媒体类型

[«CSS 图像拼合技术](#)
[CSS 属性选择器 »](#)

媒体类型允许你指定文件将如何在不同媒体呈现。该文件可以以不同的方式显示在屏幕上，在纸张上，或听觉浏览器等等。

媒体类型

一些CSS属性只设计了某些媒体。例如"voice-family"属性是专为听觉用户代理。其他一些属性可用于不同的媒体类型。例如，"font-size"属性可用于屏幕和印刷媒体，但有不同的值。屏幕和纸上的文件不同，通常需要一个更大的字体， sans - serif字体比较适合在屏幕上阅读，而serif字体更容易在纸上阅读。

@media 规则

@media 规则允许在相同样式表为不同媒体设置不同的样式。

在下面的例子告诉我们浏览器屏幕上显示一个14像素的Verdana字体样式。但是如果页面打印，将是10个像素的Times字体。请注意， font-weight在屏幕上和纸上设置为粗体：

```
<html>
<head>
<style>
@media screen
{
  p.test {font-family:verdana,sans-serif;font-size:14px;}
}
@media print
{
  p.test {font-family:times,serif;font-size:10px;}
}
@media screen,print
{
  p.test {font-weight:bold;}
}
</style>
</head>

<body>
....
</body>
</html>
```

你可以自己尝试看看！如果您使用的是Mozilla / Firefox或IE5+打印此页，你会看到， "Media Types"将使用另一种比其他文本字体大小小点的字体显示。

其他媒体类型

注意：媒体类型名称不区分大小写。

媒体类型	描述
all	用于所有的媒体设备。
aural	用于语音和音频合成器。

braille 用于盲人用点字法触觉回馈设备。
embossed 用于分页的盲人用点字法打印机。
handheld 用于小的手持的设备。
print 用于打印机。
projection 用于方案展示，比如幻灯片。
screen 用于电脑显示器。
tty 用于使用固定密度字母栅格的媒体，比如电传打字机和终端。
tv 用于电视机类型的设备。
]

[

CSS 属性选择器

[«CSS 媒体类型
CSS 总结»](#)

具有特定属性的HTML元素样式

具有特定属性的HTML元素样式不仅仅是class和id。

注意：IE7和IE8需声明!DOCTYPE才支持属性选择器！IE6和更低的版本不支持属性选择器。

属性选择器

下面的例子是把包含标题（title）的所有元素变为蓝色：

实例

```
[title]
{
color:blue;
}
```

[尝试一下»](#)

属性和值选择器

下面的例子是把属性为 title="zqiangxuetang" 的标签更改样式：

实例

```
[title=zqiangxuetang]
{
border:5px solid green;
}
```

[尝试一下»](#)

属性和值的选择器 - 多值

下面是包含指定值的title属性的元素样式的例子，使用（~）分隔属性和值：

实例

```
[title~hello] { color:blue; }
```

[尝试一下»](#)

下面是包含指定值的long属性的元素样式的例子，使用（|）分隔属性和值：

实例

```
[lang=en] { color:blue; }
```

[尝试一下 »](#)

表单样式

属性选择器样式无需使用class或id的形式:

实例

```
input[type='text']
{
width:150px;
display:block;
margin-bottom:10px;
background-color:yellow;
}
input[type='button']
{
width:120px;
margin-left:35px;
display:block;
}
```

[尝试一下 »](#)

]

[

CSS 总结

[«CSS 属性选择器
CSS 实例»](#)

CSS 总结

本教程已向你讲解了如何创建样式表来同时控制多重页面的样式和布局。

你已经学会如何使用 CSS 来添加背景、格式化文本、以及格式化边框，并定义元素的填充和边距。

同时，你也学会了如何定位元素、控制元素的可见性和尺寸、设置元素的形状、将一个元素置于另一个之后，以及向某些选择器添加特殊的效果，比如链接。

如果需要更多关于 CSS 的信息，请参阅我们的 [CSS 实例](#), [CSS 参考手册](#), and [CSS3 教程](#).

你已经学习了CSS,下一步学习什么呢?

下一步应该学习 JavaScript 。

JavaScript

JavaScript 是最流行的语言之一。

JavaScript 是属于 web 的语言，它适用于 PC、笔记本电脑、平板电脑和移动电话。

JavaScript可以使您的网站更具活力。

许多 HTML 开发者都不是程序员，但是 JavaScript 却拥有非常简单的语法。几乎每个人都有能力将小的 JavaScript 片段添加到网页中。如果您希望

如果您希望学习更多关于 JavaScript 的知识，请马上访问我们的[JavaScript 教程](#).

]

[

CSS 实例

[«CSS 总结](#)
[CSS 组合选择符 »](#)

CSS背景

[设置页面的背景颜色](#)

[设置不同元素的背景颜色](#)

[设置一个图像作为页面的背景](#)

[错误的的背景图片](#)

[如何在水平方向重复背景图像](#)

[如何定位背景图像](#)

[一个固定的背景图片（这个图片不会随页面的其余部分滚动）](#)

[在一个声明的所有背景属性](#)

[高级的背景例子](#)

[背景属性的解释](#)

CSS文本

[设置不同元素的文本颜色](#)

[文本对齐](#)

[移除链接下划线](#)

[装饰文字](#)

[控制文本中的字母](#)

[缩进文本](#)

[指定了字符之间的空间](#)

[指定了行与行之间的空间](#)

[设置元素的文本方向](#)

[增加单词之间的空格](#)

[在一个元素内禁用文字换行](#)

[内部文字图像的垂直对齐](#)

[Text属性的解释](#)

CSS的字体

[设置文本的字体](#)

[设置字体大小](#)

[用px设置的字体的大小](#)

[用em设置的字体的大小](#)

[用百分比和em设置字体的大小](#)

[设置字体样式](#)

[设置字体的异体](#)

[设置字体的粗细](#)

[在一个声明的所有字体属性](#)

[Font属性的解释](#)

CSS链接

[为访问/未访问链接添加不同的颜色](#)

[在链接上使用文本装饰](#)

[指定链接的背景颜色](#)

[超链接添加其他样式](#)

[高级 - 创建链接框](#)

[链接属性的解释](#)

CSS列表

[列表中所有不同的列表项标记](#)

[设置作为列表项标记的图像](#)

[使用Crossbrowser解决方案设置一个列表项标记的图像](#)

[在一个声明中的所有列表属性](#)

[列表属性的解释](#)

CSS表格

[指定一个表的Th, TD元素和黑色边框](#)

[使用border-collapse](#)

[指定表格的宽度和高度](#)

[设置内容的水平对齐方式（文本对齐）](#)

[设置内容的垂直对齐（垂直对齐）](#)

[指定TH和TD元素的填充](#)

[指定表格边框的颜色](#)

[设置表格标题的位置](#)

[创建一个奇特的表](#)

[表格属性的解释](#)

CSS盒模型

[指定元素的总宽度为250像素](#)

[使用Crossbrowser解决方案指定元素的总宽度为250像素的](#)

[盒模型的解释](#)

CSS边框

[设置四个边框的宽度](#)

[设置上边框的宽度](#)

[设置底部边框的宽度](#)

[设置左边框的宽度](#)

[设置右边框的宽度](#)

[设置四个边框的样式](#)

[设置上边框的样式](#)

[设置下边框的样式](#)

[设置左边框的样式](#)

[设置右边框的样式](#)

[设置四个边框的颜色](#)

[设置上边框的颜色](#)

[设置下边框的颜色](#)

[设置左边框的颜色](#)

[设置右边框的颜色](#)

[在一个声明中的所有边框属性](#)

[每边设置不同的边框](#)

[在一个声明中的所有顶部边框属性](#)

[在一个声明中的所有下边框属性](#)

[在一个声明中的所有左边框属性](#)

[在一个声明中的所有右边框属性](#)

[边框属性的解释](#)

CSS轮廓

[围绕一个元素（outline），绘制一条线](#)

[设置轮廓的样式](#)

[设置轮廓颜色](#)

[设置轮廓的宽度](#)

[轮廓属性的解释](#)

CSS边距

[指定一个元素的边距](#)

[边距缩写属性](#)

[文本顶部边距设置的值使用厘米](#)

[使用百分比值设置文本的底部边缘](#)

[使用厘米值设置文本的左边距](#)

[Margin属性的解释](#)

CSS填充

[设置元素的左部填充](#)

[设置元素的右部填充](#)

[设置元素的顶部填充](#)

[设置元素的底部填充](#)

[在一个声明中的所有填充属性](#)

[padding属性的解释](#)

CSS分組和嵌套

[组选择器](#)

[嵌套选择器](#)

[分組和嵌套解释](#)

CSS尺寸

[使用像素值设置图像的高度](#)

[使用百分比设置图像的高度](#)

[使用像素值来设置元素的宽度](#)

[使用百分比来设置元素的宽度](#)

[设置元素的最大高度](#)

[使用像素值设置元素的最大宽度](#)

[使用百分比来设置元素的最大宽度](#)

[设置元素的最低高度](#)

[使用像素值来设置元素的最小宽度](#)

[使用百分比来设置元素的最小宽度](#)

[尺寸属性的解释](#)

CSS显示

[如何隐藏一个元素\(visibility:hidden\)](#)

[如何不显示元素\(display:none\)](#)

[如何显示一个元素的内联元素](#)

[如何显示一个元素的块元素](#)

[H如何使用表格的collapse属性](#)

[Display属性的解释](#)

CSS定位

[元素相对浏览器窗口的位置](#)

[元素的相对定位](#)

[元素的绝对定位](#)

[重叠的元素](#)

[设置元素的形状](#)

[如何使用滚动条来显示元素内溢出的内容](#)

[如何设置浏览器自动溢出处理](#)

[使用像素值设置图像的顶部](#)

[使用像素值设置图像的底部](#)

[使用像素值设置图像的左边](#)

[使用像素值设置图像的右边](#)

[更改光标](#)

[定位属性的解释](#)

CSS浮动

[简单的使用float属性](#)

[为图像添加边框和边距并浮动到段落的左侧](#)

[标题和图片向右侧浮动](#)

[让段落的第一个字母浮动到左侧](#)

[使用float属性创建一个图片廊](#)

[开启float - clear属性](#)

[创建一个水平菜单](#)

[创建一个没有表格的网页](#)

[Float属性的解释](#)

CSS对齐元素

[使用margin的中间调整](#)

[左/右位置对齐](#)

[使用Crossbrowser解决方案，设置左/右位置对齐](#)

[左/右对齐，浮动](#)

[使用Crossbrowser解决方案，设置左/右位置对齐，浮动](#)

[对齐属性解释](#)

CSS生成的内容

[把括号内的URL用content属性插入到每个链接后面](#)

[章节和分节的编号是"第1节"，"1.1"，"1.2"等](#)

[quotes 属性指定了引号](#)

CSS伪类

[添加不同颜色的超链接](#)

[给超链接添加其他样式](#)

[使用：焦点](#)

[:first-child - 匹配了第一个p元素](#)

[:first-child - 匹配了第一个p元素中的l元素](#)

[:first-child - 匹配了第一个p元素中的所有l元素](#)

[使用:lang](#)

[伪类的解释](#)

CSS伪元素

[把文本的第一个字母设为特殊的字母](#)

[把第一行文字设置为特殊](#)

[把第一行文字的的第一个字母设置为特殊](#)

[使用：在一个元素之前插入一些内容](#)

[使用：在一个元素之后插入一些内容](#)

[伪元素的解释](#)

CSS导航栏

[垂直导航栏的全样式](#)

[水平导航栏的全样式](#)

[导航栏的解释](#)

CSS图片廊

[图片廊](#)

[图片廊解释](#)

CSS图像的不透明度

[创建透明图像 - 鼠标悬停效果](#)

[创建一个背景图像与文本的透明框](#)

[图像的不透明度解释](#)

CSS图像拼合

[图像拼合](#)

[图像拼合-导航列表](#)

[悬停效果与图像拼合](#)

[图像拼合解释](#)

CSS属性选择器

[选择具有title属性的元素](#)

[选择标题=一个特定值的元素](#)

[选择标题=一个特定值的元素\(使用 \(~\) 分隔属性和值\)](#)

[选择标题=一个特定值的元素\(使用 \(|\) 分隔属性和值\)](#)

[属性选择器解释](#)

]

[

CSS3 教程

这是第一篇

[CSS3 简介 »](#)

CSS 用于控制网页的样式和布局。

CSS3 是最新的 CSS 标准。

本教程向您讲解 CSS3 中的新特性。

[开始学习 CSS3!](#)



CSS3 实例



```
div
{
transform: rotate(30deg);
}
```

[尝试一下 »](#)

点击 "尝试一下" 按钮查看在线实例。

CSS3 参考手册

在 自强学堂 中, 我们提供完整的 CSS3 参考手册, 包括所有属性和选择器的语法、实例、浏览器支持信息。

[CSS 属性参考手册](#)

[CSS3 浏览器支持情况](#)

[CSS 选择器参考手册](#)

[CSS 颜色参考手册](#)

]

[

CSS3 简介

[«CSS3 教程](#)
[CSS3 边框 \(Borders\) »](#)

对CSS3已完全向后兼容，所以你不必改变现有的设计。浏览器将永远支持CSS2。

CSS3 模块

CSS3被拆分为"模块"。旧规范已拆分成小块，还增加了新的。

一些最重要CSS3模块如下：

- 选择器
- 盒模型
- 背景和边框
- 文字特效
- 2D/3D转换
- 动画
- 多列布局
- 用户界面

CSS3 建议

W3C的CSS3规范仍在开发。

但是，许多新的CSS3属性已在现代浏览器使用。

]

[

CSS3 边框（Borders）

[«CSS3 简介](#)
[CSS3 背景 »](#)

CSS3 Borders

用CSS3，你可以创建圆角边框，添加阴影框，并作为边界的形象而不使用设计程序，如Photoshop。

在本章中，您将了解以下的边框属性：

- border-radius
- box-shadow
- border-image

浏览器支持

属性 浏览器支持

border-radius

box-shadow

border-image

Internet Explorer 9+ 支持 border-radius 和 box-shadow.

Firefox, Chrome, 和 Safari 支持所有最新的 border 属性.

注意： 前缀是-webkit- 的Safari支持阴影边框。

前缀是-o-的Opera支持边框图像。

CSS3 圆角

在CSS2中添加圆角棘手。我们不得不在每个角落使用不同的图像。

在CSS3中，很容易创建圆角。

在CSS3中border-radius属性被用于创建圆角：

这是圆角边框！

实例



在div中添加圆角元素：

```
div
{
border:2px solid;
border-radius:25px;
}
```

[尝试一下 »](#)

CSS3盒阴影

CSS3中的box-shadow属性被用来添加阴影:

实例



在div中添加box-shadow属性

```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

[尝试一下 »](#)

CSS3边界图片

有了CSS3的border-image属性，你可以使用图像创建一个边框：

border-image属性允许你指定一个图片作为边框！ 用于创建上文边框的原始图像：

在div中使用图片创建边框：



实例



在div中使用图片创建边框

```
div
{
border-image:url(border.png) 30 30 round;
-webkit-border-image:url(border.png) 30 30 round; /* Safari 5 and older */
-o-border-image:url(border.png) 30 30 round; /* Opera */
}
```

[尝试一下 »](#)

新边框属性

属性	说明	CSS
border-image	设置所有边框图像的速记属性。	3
border-radius	一个用于设置所有四个边框- *-半径属性的速记属性	3
box-shadow	附加一个或多个下拉框的阴影	3

]

[

CSS3 背景

[«CSS3 边框 \(Borders\)](#)
[CSS3 渐变 »](#)

CSS3 背景

CSS3中包含几个新的背景属性，提供更大背景元素控制。

在本章您将了解以下背景属性：

- background-size
- background-origin

您还将学习如何使用多重背景图像。

浏览器支持

属性	浏览器支持
background-size	
background-origin	

Internet Explorer 9+, Firefox, Chrome, Safari, 和 Opera 支持最新的背景属性。

CSS3 background-size 属性

background-size指定背景图像的大小。CSS3以前，背景图像大小由图像的实际大小决定。

CSS3中可以指定背景图片，让我们重新在不同的环境中指定背景图片的大小。您可以指定像素或百分比大小。

你指定的大小是相对于父元素的宽度和高度的百分比的大小。

实例 1



重置背景图像：

```
div
{
background:url(img_flwr.gif);
background-size:80px 60px;
background-repeat:no-repeat;
}
```

[尝试一下 »](#)

实例 2



伸展背景图像完全填充内容区域：

```
div
{
background:url(img_flwr.gif);
background-size:100% 100%;
}
```



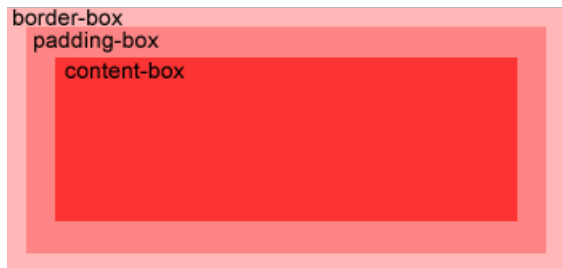
```
background-repeat:no-repeat;
}
```

[尝试一下 »](#)

CSS3的background-Origin属性

background-Origin属性指定了背景图像的位置区域。

content-box, padding-box,和 border-box区域内可以放置背景图像。



实例



在 content-box 中定位背景图片：

```
div
{
background:url(img_flwr.gif);
background-repeat:no-repeat;
background-size:100% 100%;
background-origin:content-box;
}
```

[尝试一下 »](#)

CSS3 多个背景图像

CSS3 允许你在元素

那个添加多个背景图像。

实例



在 body 元素中设置两个背景图像：

```
body
{
background-image:url(img_flwr.gif),url(img_tree.gif);
}
```

[尝试一下 »](#)

新的背景属性

顺序	描述	CSS
background-clip	规定背景的绘制区域。	3
background-origin	规定背景图片的定位区域。	3
background-size	规定背景图片的尺寸。	3

]

[

CSS3 文本效果

[«CSS3 渐变](#)
[CSS3 字体 »](#)

CSS3 文本效果

CSS3中包含几个新的文本特征。

在本章中您将了解以下文本属性：

- text-shadow
- word-wrap

浏览器支持

属性	浏览器支持
text-shadow	
word-wrap	

Internet Explorer 10, Firefox, Chrome, Safari, 和 Opera支持text-shadow 属性。

所有的主流浏览器支持自动换行（word-wrap）属性。

注意： Internet Explorer 9及更早IE版本不支持 text-shadow 属性.

CSS3的文本阴影

CSS3中，text-shadow属性适用于文本阴影。

Text shadow effect!

您指定了水平阴影，垂直阴影，模糊的距离，以及阴影的颜色：

实例



给标题添加阴影：

```
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
```

[尝试一下 »](#)

CSS3的换行

如果某个单词太长，不适合在一个区域内，它扩展到外面：

This paragraph contains a
very long word:
thisisaveryveryveryveryveryverylongword.

The long word will break
and wrap to the next line.

CSS3中，自动换行属性允许您强制文本换行 - 即使这意味着分裂它中间的一个字：

This paragraph contains a
very long word:
thisisaveryveryveryveryveryv
erylongword. The long word
will break and wrap to the
next line.

CSS代码如下：

实例



允许长文本换行：

```
p {word-wrap:break-word;}
```

[尝试一下 »](#)

New Text Properties

属性	描述	CSS
hanging-punctuation	规定标点字符是否位于线框之外。	3
punctuation-trim	规定是否对标点字符进行修剪。	3
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行。	3
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色。	3
text-justify	规定当 text-align 设置为 "justify" 时所使用的对齐方法。	3
text-outline	规定文本的轮廓。	3
text-overflow	规定当文本溢出包含元素时发生的事情。	3
text-shadow	向文本添加阴影。	3
text-wrap	规定文本的换行规则。	3
word-break	规定非中日韩文本的换行规则。	3
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行。	3

]

[

CSS3 字体

[«CSS3 文本效果
CSS3 2D 转换»](#)

With CSS3, web designers are no longer forced to use only "web-safe" fonts.

CSS3 @font-face 规则

以前CSS3的版本，网页设计师不得不使用用户计算机上已经安装的字体。

使用CSS3，网页设计师可以使用他/她喜欢的任何字体。

当你发现您要使用的字体文件时，只需简单的将字体文件包含在网站中，它会自动下载给需要的用户。

您所选择的字体在新的CSS3版本有关于@font-face规则描述。

您"自己的"的字体是在 CSS3 @font-face 规则中定义的。

浏览器支持

属性 **浏览器支持**
@font-face

Internet Explorer 9+, Firefox, Chrome, Safari, 和 Opera 支持 WOFF (Web Open Font Format) 字体。

Firefox, Chrome, Safari, 和 Opera 支持 .ttf(True Type字体)和.otf(OpenType)字体字体类型)。

Chrome, Safari 和 Opera 也支持 SVG 字体/折叠。

Internet Explorer 同样支持 EOT (Embedded OpenType) 字体。

注意： Internet Explorer 8 以及更早的版本不支持新的 @font-face 规则。

使用您需要的字体

在新的 @font-face 规则中，您必须首先定义字体的名称（比如 myFirstFont），然后指向该字体文件。



提示： URL请使用小写字母的字体，大写字母在IE中会产生意外的结果

如需为 HTML 元素使用字体，请通过 font-family 属性来引用字体的名称 (myFirstFont):

实例



```
<style>
@font-face
{
font-family: myFirstFont;
src: url(sansation_light.woff);
}
```

```
div
{
font-family:myFirstFont;
}
</style>
```

[尝试一下 »](#)

使用粗体文本

您必须添加另一个包含粗体文字的@font-face规则:

实例



```
@font-face
{
font-family: myFirstFont;
src: url(sansation_bold.woff);
font-weight:bold;
}
```

[尝试一下 »](#)

该文件"Sansation_Bold.ttf"是另一种字体文件，包含Sansation字体的粗体字。

浏览器使用这一文本的字体系列"myFirstFont"时应该呈现为粗体。

这样你就可以有许多相同的字体@font-face的规则。

CSS3 字体描述

下表列出了所有的字体描述和里面的@font-face规则定义:

描述符	值	描述
font-family	<i>name</i>	必需。规定字体的名称。
src	<i>URL</i>	必需。定义字体文件的 URL。
	<ul style="list-style-type: none">normalcondensedultra-condensedextra-condensed	
font-stretch	<ul style="list-style-type: none">semi-condensedexpandedsemi-expandedextra-expandedultra-expanded	可选。定义如何拉伸字体。默认是 "normal"。
font-style	<ul style="list-style-type: none">ormalitalicoblique	可选。定义字体的样式。默认是 "normal"。
font-weight	<ul style="list-style-type: none">normalbold100200300400500600700800900	可选。定义字体的粗细。默认是 "normal"。

unicode-range

unicode-range

可选。定义字体支持的 UNICODE 字符范围。默认是 "U+0-10FFFF"。

]

[

CSS3 2D 转换

[«CSS3 字体](#)
[CSS3 3D 转换 »](#)

CSS3 转换

CSS3转换，我们可以移动，比例化，反过来，旋转，和拉伸元素。



它是如何工作？

变换的效果，让某个元素改变形状，大小和位置。

您可以转换您使用2D或3D元素。

浏览器支持

属性 浏览器支持
transform

Internet Explorer 10, Firefox, 和 Opera支持transform属性.

Chrome 和 Safari 要求前缀 -webkit- 版本.

注意： Internet Explorer 9 要求前缀 -ms- 版本.

2D 转换

在本章您将了解2D变换方法：

- translate()
- rotate()
- scale()
- skew()
- matrix()

在下一章中您将了解3D转换。

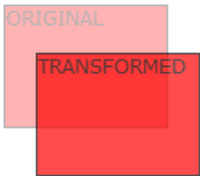
实例



```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg); /* IE 9 */
-webkit-transform: rotate(30deg); /* Safari and Chrome */
}
```


[尝试一下 »](#)

translate() 方法



translate()方法，根据左(X轴)和顶部(Y轴)位置给定的参数，从当前元素位置移动。

实例



```
div
{
transform: translate(50px,100px);
-ms-transform: translate(50px,100px); /* IE 9 */
-webkit-transform: translate(50px,100px); /* Safari and Chrome */
}
```

[尝试一下 »](#)

translate值（50px，100px）是从左边元素移动50个像素，并从顶部移动100像素。

rotate() 方法



rotate()方法，在一个给定度数顺时针旋转的元素。负值是允许的，这样是元素逆时针旋转。

实例



```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg); /* IE 9 */
-webkit-transform: rotate(30deg); /* Safari and Chrome */
}
```

[尝试一下 »](#)

rotate值（30deg）元素顺时针旋转30度。

scale() 方法



scale()方法，该元素增加或减少的大小，取决于宽度（X轴）和高度（Y轴）的参数：

实例



```
div
{
transform: scale(2,4);
-ms-transform: scale(2,4); /* IE 9 */
-webkit-transform: scale(2,4); /* Safari and Chrome */
}
```

[尝试一下 »](#)

scale（2,4）转变宽度为原来的大小的2倍，和其原始大小4倍的高度。

skew() 方法



skew()方法，该元素会根据横向（X轴）和垂直（Y轴）线参数给定角度：

实例



```
div
{
transform: skew(30deg,20deg);
-ms-transform: skew(30deg,20deg); /* IE 9 */
-webkit-transform: skew(30deg,20deg); /* Safari and Chrome */
}
```

[尝试一下 »](#)

skew(30deg,20deg)是绕X轴和Y轴周围20度30度的元素。

matrix() 方法



matrix()方法和2D变换方法合并成一个。

matrix 方法有六个参数，包含旋转，缩放，移动（平移）和倾斜功能。

实例



利用matrix()方法旋转div元素30°

```
div
{
transform: matrix(0.866,0.5,-0.5,0.866,0,0);
-ms-transform: matrix(0.866,0.5,-0.5,0.866,0,0); /* IE 9 */
}
```

```
-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Safari and Chrome */
}
```

[尝试一下 »](#)

新转换属性

以下列出了所有的转换属性:

Property	描述	CSS
transform	适用于2D或3D转换的元素	3
transform-origin	允许您更改转化元素位置	3

2D 转换方法

函数	描述
<code>matrix(<i>n,n,n,n,n,n</i>)</code>	定义 2D 转换，使用六个值的矩阵。
<code>translate(<i>x,y</i>)</code>	定义 2D 转换，沿着 X 和 Y 轴移动元素。
<code>translateX(<i>n</i>)</code>	定义 2D 转换，沿着 X 轴移动元素。
<code>translateY(<i>n</i>)</code>	定义 2D 转换，沿着 Y 轴移动元素。
<code>scale(<i>x,y</i>)</code>	定义 2D 缩放转换，改变元素的宽度和高度。
<code>scaleX(<i>n</i>)</code>	定义 2D 缩放转换，改变元素的宽度。
<code>scaleY(<i>n</i>)</code>	定义 2D 缩放转换，改变元素的高度。
<code>rotate(<i>angle</i>)</code>	定义 2D 旋转，在参数中规定角度。
<code>skew(<i>x-angle,y-angle</i>)</code>	定义 2D 倾斜转换，沿着 X 和 Y 轴。
<code>skewX(<i>angle</i>)</code>	定义 2D 倾斜转换，沿着 X 轴。
<code>skewY(<i>angle</i>)</code>	定义 2D 倾斜转换，沿着 Y 轴。

]

[

CSS3 3D 转换

[«CSS3 2D 转换](#)
[CSS3 过渡 »](#)

3D Transforms

CSS3 允许您使用 3D 转换来对元素进行格式化。

在本章中，您将学到其中的一些 3D 转换方法：

- rotateX()
- rotateY()

点击下面的元素，来查看 2D 转换与 3D 转换之间的不同之处：



浏览器支持

属性 浏览器支持
transform

Internet Explorer 10 和 Firefox 支持 3D 转换.

Chrome 和 Safari 必须添加前缀 -webkit-.

Opera 还不支持 3D 转换(支持 [2D 转换](#)).

rotateX() 方法



rotateX()方法，围绕其在一个给定度数X轴旋转的元素。

实例



```
div
{
transform: rotateX(120deg);
-webkit-transform: rotateX(120deg); /* Safari and Chrome */
}
```

[尝试一下 »](#)

rotateY() 方法



rotateY()方法，围绕其在一个给定度数Y轴旋转的元素。

实例



```
div
{
transform: rotateY(130deg);
-webkit-transform: rotateY(130deg); /* Safari and Chrome */
}
```

[尝试一下 »](#)

转换属性

下表列出了所有的转换属性：

属性	描述	CSS
transform	向元素应用 2D 或 3D 转换。	3
transform-origin	允许你改变被转换元素的位置。	3
transform-style	规定被嵌套元素如何在 3D 空间中显示。	3
perspective	规定 3D 元素的透视效果。	3
perspective-origin	规定 3D 元素的底部位置。	3
backface-visibility	定义元素在不面对屏幕时是否可见。	3

3D 转换方法

函数	描述
<code>matrix3d(<i>n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n</i>)</code>	定义 3D 转换，使用 16 个值的 4x4 矩阵。
<code>translate3d(<i>x,y,z</i>)</code>	定义 3D 转化。
<code>translateX(<i>x</i>)</code>	定义 3D 转化，仅使用用于 X 轴的值。
<code>translateY(<i>y</i>)</code>	定义 3D 转化，仅使用用于 Y 轴的值。
<code>translateZ(<i>z</i>)</code>	定义 3D 转化，仅使用用于 Z 轴的值。
<code>scale3d(<i>x,y,z</i>)</code>	定义 3D 缩放转换。
<code>scaleX(<i>x</i>)</code>	定义 3D 缩放转换，通过给定一个 X 轴的值。
<code>scaleY(<i>y</i>)</code>	定义 3D 缩放转换，通过给定一个 Y 轴的值。
<code>scaleZ(<i>z</i>)</code>	定义 3D 缩放转换，通过给定一个 Z 轴的值。
<code>rotate3d(<i>x,y,z,angle</i>)</code>	定义 3D 旋转。
<code>rotateX(<i>angle</i>)</code>	定义沿 X 轴的 3D 旋转。
<code>rotateY(<i>angle</i>)</code>	定义沿 Y 轴的 3D 旋转。
<code>rotateZ(<i>angle</i>)</code>	定义沿 Z 轴的 3D 旋转。
<code>perspective(<i>n</i>)</code>	定义 3D 转换元素的透视视图。

]

[

CSS3 过渡

[«CSS3 3D 转换](#)
[CSS3 动画 »](#)

CSS3 过渡

CSS3中，我们为了添加某种效果可以从一种样式转变到另一个的时候，无需使用Flash动画或JavaScript。用鼠标移过下面的元素：

用鼠标移过下面的元素：



浏览器支持

属性 浏览器支持
transition

Internet Explorer 10, Firefox, Opera, Chrome, 和Opera 支持transition 属性.

Safari 需要前缀 -webkit-。

注意： Internet Explorer 9 以及更早的版本，不支持 transition 属性。

注意： Chrome 25 以及更早的版本，需要前缀 -webkit-。

它是如何工作？

CSS3 过渡是元素从一种样式逐渐改变为另一种的效果。

要实现这一点，必须规定两项内容：

- 指定要添加效果的CSS属性
- 指定效果的持续时间。

实例



应用于宽度属性的过渡效果，时长为 2 秒：

```
div
{
transition: width 2s;
-webkit-transition: width 2s; /* Safari */
}
```

注意： 如果未指定的期限，transition将没有任何效果，因为默认值是0。

指定的CSS属性的值更改时效果会发生变化。一个典型CSS属性的变化是用户鼠标放在一个元素上时：

实例



规定当鼠标指针悬浮(:hover)于 <div>元素上时：

```
div:hover
{
width:300px;
}
```

[尝试一下 »](#)

注意： 当鼠标光标移动到该元素时，它逐渐改变它原有样式

多项改变

要添加多个样式的变换效果，添加的属性由逗号分隔：

实例



添加了宽度，高度和转换效果：

```
div
{
transition: width 2s, height 2s, transform 2s;
-webkit-transition: width 2s, height 2s, -webkit-transform 2s;
}
```

[尝试一下 »](#)

过渡属性

下表列出了所有的过渡属性：

属性	描述	CSS
transition	简写属性，用于在一个属性中设置四个过渡属性。	3
transition-property	规定应用过渡的 CSS 属性的名称。	3
transition-duration	定义过渡效果花费的时间。默认是 0。	3
transition-timing-function	规定过渡效果的时间曲线。默认是 "ease"。	3
transition-delay	规定过渡效果何时开始。默认是 0。	3

下面的两个例子设置所有过渡属性：

实例



在一个例子中使用所有过渡属性：

```
div
{
transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
/* Safari */
-webkit-transition-property:width;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:linear;
-webkit-transition-delay:2s;
}
```

[尝试一下 »](#)

实例



与上面的例子相同的过渡效果，但是使用了简写的 transition 属性：

```
div
{
transition: width 1s linear 2s;
/* Safari */
-webkit-transition:width 1s linear 2s;
}
```

[尝试一下 »](#)

]

[

CSS3 动画

[«CSS3 过渡](#)
[CSS3 多列 »](#)

CSS3 动画

CSS3，我们可以创建动画，它可以取代许多网页动画图像，Flash动画，和JAVAScripts。

CSS3
动画

CSS3 @keyframes 规则

要创建CSS3动画，你将不得不了解@keyframes规则。

@keyframes规则是创建动画。 @keyframes规则内指定一个CSS样式和动画将逐步从目前的样式更改为新的样式。

浏览器支持

属性 浏览器支持

@keyframes
animation

Internet Explorer 10、Firefox 以及 Opera 支持 @keyframes 规则和 animation 属性。

Chrome 和 Safari 需要前缀 -webkit-。

注意：Internet Explorer 9，以及更早的版本，不支持 @keyframe 规则或 animation 属性。

实例



```
@keyframes myfirst
{
  from {background: red;}
  to {background: yellow;}
}
```

```
@-webkit-keyframes myfirst /* Safari and Chrome */
{
  from {background: red;}
  to {background: yellow;}
}
```

CSS3 动画

当在@keyframe创建动画，把它绑定到一个选择器，否则动画不会有任何效果。

指定至少这两个CSS3的动画属性绑定向一个选择器：

- 规定动画的名称
- 规定动画的时长

实例



把 "myfirst" 动画捆绑到 div 元素，时长：5 秒：

```
div
{
animation: myfirst 5s;
-webkit-animation: myfirst 5s; /* Safari and Chrome */
}
```

[尝试一下 »](#)

注意：您必须定义动画的名称和动画的持续时间。如果省略的持续时间，动画将无法运行，因为默认值是0。

CSS3动画是什么？

动画是使元素从一种样式逐渐变化为另一种样式的效果。

您可以改变任意多的样式任意多的次数。

请用百分比来规定变化发生的时间，或用关键词 "from" 和 "to"，等同于 0% 和 100%。

0% 是动画的开始，100% 是动画的完成。

为了得到最佳的浏览器支持，您应该始终定义 0% 和 100% 选择器。

实例



当动画为 25% 及 50% 时改变背景色，然后当动画 100% 完成时再次改变：

```
@keyframes myfirst
{
0% {background: red;}
25% {background: yellow;}
50% {background: blue;}
100% {background: green;}
}

@-webkit-keyframes myfirst /* Safari and Chrome */
{
0% {background: red;}
25% {background: yellow;}
50% {background: blue;}
100% {background: green;}
}
```

[尝试一下 »](#)

实例



改变背景色和位置：

```
@keyframes myfirst
{
0% {background: red; left:0px; top:0px;}
25% {background: yellow; left:200px; top:0px;}
50% {background: blue; left:200px; top:200px;}
75% {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
```

```
@-webkit-keyframes myfirst /* Safari and Chrome */
{
0% {background: red; left:0px; top:0px;}
25% {background: yellow; left:200px; top:0px;}
50% {background: blue; left:200px; top:200px;}
75% {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
```

[尝试一下 »](#)

CSS3的动画属性

下面的表格列出了 @keyframes 规则 and 所有动画属性:

属性	描述	CSS
@keyframes	规定动画。	3
animation	所有动画属性的简写属性，除了 animation-play-state 属性。	3
animation-name	规定 @keyframes 动画的名称。	3
animation-duration	规定动画完成一个周期所花费的秒或毫秒。默认是 0。	3
animation-timing-function	规定动画的速度曲线。默认是 "ease"。	3
animation-delay	规定动画何时开始。默认是 0。	3
animation-iteration-count	规定动画被播放的次数。默认是 1。	3
animation-direction	规定动画是否在下一周期逆向地播放。默认是 "normal"。	3
animation-play-state	规定动画是否正在运行或暂停。默认是 "running"。	3

下面两个例子设置所有动画属性:

实例



运行myfirst动画，设置所有的属性:

```
div
{
animation-name: myfirst;
animation-duration: 5s;
animation-timing-function: linear;
animation-delay: 2s;
animation-iteration-count: infinite;
animation-direction: alternate;
animation-play-state: running;
/* Safari and Chrome: */
-webkit-animation-name: myfirst;
-webkit-animation-duration: 5s;
-webkit-animation-timing-function: linear;
-webkit-animation-delay: 2s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-webkit-animation-play-state: running;
}
```

[尝试一下 »](#)

实例



与上面的动画相同，但是使用了简写的动画 animation 属性:

```
div
{
animation: myfirst 5s linear 2s infinite alternate;
/* Safari and Chrome: */
-webkit-animation: myfirst 5s linear 2s infinite alternate;
}
```

[尝试一下 »](#)

]

[

CSS3 多列

[«CSS3 动画](#)
[CSS3 用户界面 »](#)

CSS3 多列

通过 CSS3，您能够创建多个列来对文本进行布局 - 就像报纸那样！

在本章中，您将学习如下多列属性：

- column-count
- column-gap
- column-rule

浏览器支持

属性 浏览器支持

column-count

column-gap

column-rule

Internet Explorer 10 和 Opera 支持多列属性。

Firefox 需要前缀 -moz-。

Chrome 和 Safari 需要前缀 -webkit-。

注意： Internet Explorer 9 以及更早的版本不支持多列属性。

CSS3创建多列

column-count属性指定元素的列数应分为：

实例



划分成三列的div元素的文本：

```
div
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;
}
```

[尝试一下 »](#)

CSS3的指定列之间的差距

column-gap属性指定的列之间的差距：

实例



指定列之间40个像素差距：

```
div
{
-moz-column-gap:40px; /* Firefox */
-webkit-column-gap:40px; /* Safari and Chrome */
column-gap:40px;
}
```

[尝试一下 »](#)

CSS3列规则

column-rule属性设置列之间的宽度，样式和颜色。

实例



指定列之间的宽度，样式和颜色的规则：

```
div
{
-moz-column-rule:3px outset #ff00ff; /* Firefox */
-webkit-column-rule:3px outset #ff00ff; /* Safari and Chrome */
column-rule:3px outset #ff00ff;
}
```

[尝试一下 »](#)

新多列属性

属性	说明	CSS
column-count	指定元素应分为的列数	3
column-fill	指定如何填充列	3
column-gap	指定列之间差距	3
column-rule	一个用于设置所有列规则的简写属性	3
column-rule-color	指定的列之间颜色规则	3
column-rule-style	指定的列之间的样式规则	3
column-rule-width	指定的列之间的宽度规则	3
column-span	指定一个元素应该横跨多少列	3
column-width	指定列的宽度	3
columns	缩写属性设置列宽和列数	3

]

CSS3 用户界面

[«CSS3 多列](#)
[CSS3 渐变 »](#)

CSS3 用户界面

在 CSS3 中, 增加了一些新的用户界面特性来调整元素尺寸, 框尺寸和外边框。

在本章中, 您将了解以下的用户界面属性:

- `resize`
- `box-sizing`
- `outline-offset`

浏览器支持

属性	浏览器支持
<code>resize</code>	
<code>box-sizing</code>	
<code>outline-offset</code>	

Firefox、Chrome 以及 Safari 支持 `resize` 属性。

Internet Explorer、Chrome、Safari 以及 Opera 支持 `box-sizing` 属性。Firefox 需要前缀 `-moz-`。

所有主流浏览器都支持 `outline-offset` 属性, 除了 Internet Explorer。

CSS3 调整尺寸(Resizing)

CSS3中, `resize`属性指定一个元素是否应该由用户去调整大小。

这个 `div` 元素由用户调整大小。(在 Firefox 4+, Chrome, 和 Safari中)

CSS代码如下:

实例



由用户指定一个

```
div
{
  resize:both;
  overflow:auto;
}
```

[尝试一下 »](#)

CSS3 方框大小调整(Box Sizing)

`box-sizing` 属性允许您以确切的方式定义适应某个区域的具体内容。

实例



规定两个并排的带边框方框：

```
div
{
box-sizing:border-box;
-moz-box-sizing:border-box; /* Firefox */
width:50%;
float:left;
}
```

[尝试一下 »](#)

CSS3 外形修饰（outline-offset）

outline-offset 属性对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。

轮廓与边框有两点不同：

- 轮廓不占用空间
- 轮廓可能是非矩形

这个 div 在边框之外 15 像素处有一个轮廓。

The CSS code is as follows:

实例



规定边框边缘之外 15 像素处的轮廓：

```
div
{
border:2px solid black;
outline:2px solid red;
outline-offset:15px;
}
```

[尝试一下 »](#)


新的用户界面特性

属性	说明	CSS
appearance	允许您使一个元素的外观像一个标准的用户界面元素	3
box-sizing	允许你以适应区域而用某种方式定义某些元素	3
icon	Provides the author the ability to style an element with an iconic equivalent	3
nav-down	指定在何处使用箭头向下导航键时进行导航	3
nav-index	指定一个元素的Tab的顺序	3
nav-left	指定在何处使用左侧的箭头导航键进行导航	3
nav-right	指定在何处使用右侧的箭头导航键进行导航	3
nav-up	指定在何处使用箭头向上导航键时进行导航	3
outline-offset	外轮廓修饰并绘制超出边框的边缘	3
resize	指定一个元素是否是由用户调整大小	3

]

CSS 参考手册

这是第一篇
[CSS 选择器 »](#)

 自强学堂 的 CSS 参考手册在所有主流浏览器中测试通过.

CSS 属性

CSS 属性组:

- [动画](#)
 - [背景](#)
 - [边框和轮廓](#)
 - [框](#)
 - [颜色](#)
 - [内容页的媒体属性](#)
 - [尺寸](#)
 - [盒子模型](#)
 - [字体](#)
 - [内容生成](#)
- [网格](#)
 - [超链接](#)
 - [线框](#)
 - [列表](#)
 - [外边距](#)
 - [字幕](#)
 - [多列](#)
 - [内边距](#)
 - [页面媒体](#)
 - [定位](#)
- [分页](#)
 - [Ruby](#)
 - [语音](#)
 - [表格](#)
 - [文本](#)
 - [2D/3D 转换](#)
 - [过渡](#)
 - [用户界面](#)

"CSS" 列指示属性是在哪个 CSS 版本中定义的 (CSS1, CSS2, 或者 CSS3).

动画属性

属性	描述	CSS
@keyframes	定义一个动画,@keyframes定义的动画名称用来被animation-name所使用。	3
animation	复合属性。检索或设置对象所应用的动画特效。	3
animation-name	检索或设置对象所应用的动画名称 ,必须与规则@keyframes配合使用，因为动画名称由@keyframes定义	3
animation-duration	检索或设置对象动画的持续时间	3
animation-timing-function	检索或设置对象动画的过渡类型	3
animation-delay	检索或设置对象动画的延迟时间	3
animation-iteration-count	检索或设置对象动画的循环次数	3
animation-direction	检索或设置对象动画在循环中是否反向运动	3
animation-play-state	检索或设置对象动画的状态	3

背景属性

属性	描述	CSS
background	复合属性。设置对象的背景特性。	1
background-attachment	设置或检索背景图像是随对象内容滚动还是固定的。必须先指定background-image属性。	1
background-color	设置或检索对象的背景颜色。	1
background-image	设置或检索对象的背景图像。	1
background-position	设置或检索对象的背景图像位置。必须先指定background-image属性。	1
background-repeat	设置或检索对象的背景图像如何铺排填充。必须先指定background-image属性。	1
background-clip	指定对象的背景图像向外裁剪的区域。	3
background-origin	S设置或检索对象的背景图像计算background-position时的参考原点(位置)。	3
background-size	检索或设置对象的背景图像的尺寸大小。	3

边框(Border) 和 轮廓(Outline) 属性

属性	描述	CSS
border	复合属性。设置对象边框的特性。	1
border-bottom	复合属性。设置对象底部边框的特性。	1
border-bottom-color	设置或检索对象的底部边框颜色。	1
border-bottom-style	设置或检索对象的底部边框样式。	1
border-bottom-width	设置或检索对象的底部边框宽度。	1
border-color	置或检索对象的边框颜色。	1
border-left	复合属性。设置对象左边边框的特性。	1
border-left-color	设置或检索对象的左边边框颜色。	1
border-left-style	设置或检索对象的左边边框颜色。	1
border-left-width	设置或检索对象的左边边框宽度。	1
border-right	复合属性。设置对象右边边框的特性。	1
border-right-color	设置或检索对象的右边边框颜色。	1
border-right-style	设置或检索对象的右边边框样式。	1
border-right-width	设置或检索对象的右边边框宽度。	1
border-style	设置或检索对象的边框样式。	1
border-top	复合属性。设置对象顶部边框的特性。	1
border-top-color	设置或检索对象的顶部边框颜色	1
border-top-style	设置或检索对象的顶部边框样式。	1
border-top-width	设置或检索对象的顶部边框宽度。	1
border-width	设置或检索对象的边框宽度。	1
outline	复合属性。设置或检索对象外的线条轮廓。	2
outline-color	设置或检索对象外的线条轮廓的颜色。	2
outline-style	设置或检索对象外的线条轮廓的样式。	2
outline-width	设置或检索对象外的线条轮廓的宽度。	2
border-bottom-left-radius	设置或检索对象的左下角圆角边框。提供2个参数，2个参数以空格分隔，每个参数允许设置1个参数值，第1个参数表示水平半径，第2个参数表示垂直半径，如第2个参数省略，则默认等于第1个参数	3
border-bottom-right-radius	设置或检索对象的右下角圆角边框。	3
border-image	设置或检索对象的边框样式使用图像来填充。	3
border-image-outset	规定边框图像超过边框的量。	3
border-image-repeat	规定图像边框是否应该被重复（repeated）、拉伸（stretched）或铺满（rounded）。	3
border-image-slice	规定图像边框的向内偏移。	3
border-image-source	规定要使用的图像，代替 border-style 属性中设置的边框样式。	3
border-image-width	规定图像边框的宽度。	3
border-radius	设置或检索对象使用圆角边框。	3
border-top-left-radius	定义左上角边框的形状。	3
border-top-right-radius	定义右下角边框的形状。	3
box-decoration-break	规定行内元素被折行	3
box-shadow	向方框添加一个或多个阴影。	3

框(Box) 属性

属性	描述	CSS
overflow-x	如果内容溢出了元素内容区域，是否对内容的左/右边缘进行裁剪。	3
overflow-y	如果内容溢出了元素内容区域，是否对内容的上/下边缘进行裁剪。	3
overflow-style	规定溢出元素的首选滚动方法。	3
rotation	围绕由 rotation-point 属性性定义的点对元素进行旋转。	3

颜色(Color) 属性

属性	描述	CSS
color-profile	允许使用源的颜色配置文件的默认以外的规范	3
opacity	设置一个元素的透明度级别	3
rendering-intent	允许超过默认颜色配置文件渲染意向的其他规范	3

内边距(Padding) 属性

属性	说明	CSS
padding	在一个声明中设置所有填充属性	1
padding-bottom	设置元素的底填充	1
padding-left	设置元素的左填充	1
padding-right	设置元素的右填充	1
padding-top	设置元素的顶部填充	1

媒体页面内容属性

属性	说明	CSS
bookmark-label	指定书签的标签	3
bookmark-level	指定了书签级别	3
bookmark-target	指定了书签链接的目标	3
float-offset	在相反的方向推动浮动元素，他们一直具有浮动	3
hyphenate-after	指定一个断字的单词断字符后的最少字符数	3
hyphenate-before	指定一个断字的单词断字符前的最少字符数	3
hyphenate-character	指定了当一个断字发生时，要显示的字符串	3
hyphenate-lines	表示连续断字的行在元素的最大数目	3
hyphenate-resource	外部资源指定一个逗号分隔的列表，可以帮助确定浏览器的断字点	3
hyphens	设置如何分割单词以改善该段的布局	3
image-resolution	指定了正确的图像分辨率	3
marks	将crop and/or cross标志添加到文档	3
string-set		3

尺寸(Dimension) 属性

属性	描述	CSS
height	设置元素的高度	1
max-height	设置元素的最大高度	2
max-width	设置元素的最大宽度	2
min-height	设置元素的最小高度	2
min-width	设置元素的最小宽度	2
width	设置元素的宽度	1

盒子模型（Flexible Box） 属性

属性	说明	CSS
box-align	指定如何对齐一个框的子元素	3
box-direction	指定在哪个方向，显示一个框的子元素	3
box-flex	指定一个框的子元素是否是灵活的或固定的大小	3
box-flex-group	指派灵活的元素到Flex组	3

box-lines	每当它在父框的空间运行时，是否指定将再上一个新的行列	3
box-ordinal-group	指定一个框的子元素的显示顺序	3
box-orient	指定一个框的子元素是否在水平或垂直方向应铺设	3
box-pack	指定横向盒在垂直框的水平位置和垂直位置	3

字体（Font） 属性

属性	说明	CSS
font	在一个声明中设置所有字体属性	1
font-family	规定文本的字体系列	1
font-size	规定文本的字体尺寸	1
font-style	规定文本的字体样式	1
font-variant	规定文本的字体样式	1
font-weight	规定字体的粗细	1
@font-face	一个规则，允许网站下载并使用其他超过"Web- safe"字体的字体	3
font-size-adjust	为元素规定 aspect 值	3
font-stretch	收缩或拉伸当前的字体系列	3

内容生成属性(Generated Content Properties)

属性	说明	CSS
content	与 :before 以及 :after 伪元素配合使用，来插入生成内容	2
counter-increment	递增或递减一个或多个计数器	2
counter-reset	创建或重置一个或多个计数器	2
quotes	设置嵌套引用的引号类型	2
crop	允许replaced元素只是作为一个对象代替整个对象的矩形区域	3
move-to	Causes an element to be removed from the flow and reinserted at a later point in the document	3
page-policy	判定基于页面的给定元素的适用于计数器的字符串值	3

网格（Grid） 属性

属性	说明	CSS
grid-columns	指定在网格中每列的宽度	3
grid-rows	指定在网格中每列的高度	3

超链接(Hyperlink) 属性

属性	说明	CSS
target	简写属性设置target-name, target-new,和target-position属性	3
target-name	指定在何处打开链接（目标位置）	3
target-new	指定是否有新的目标链接打开一个新窗口或在现有窗口打开新标签	3
target-position	指定应该放置新的目标链接的位置	3

线框(Linebox) 属性

属性	说明	CSS
alignment-adjust	允许更精确的元素的对齐方式	3
alignment-baseline	其父级指定的内联级别的元素如何对齐	3
baseline-shift	允许重新定位相对于dominant-baseline的dominant-baseline	3

dominant-baseline	指定scaled-baseline-table	3
drop-initial-after-adjust	设置下拉的主要连接点的初始对齐点	3
drop-initial-after-align	校准行内的初始行的设置就是具有首字母的框使用初级连接点	3
drop-initial-before-adjust	设置下拉的辅助连接点的初始对齐点	3
drop-initial-before-align	校准行内的初始行的设置就是具有首字母的框使用辅助连接点	3
drop-initial-size	控制局部的首字母下沉	3
drop-initial-value	激活一个下拉式的初步效果	3
inline-box-align	设置一个多行的内联块内的行具有前一个和后一个内联元素的对齐	3
line-stacking	一个速记属性设置line-stacking-strategy, line-stacking-ruby,和line-stacking-shift属性	3
line-stacking-ruby	设置包含Ruby注释元素的行对于块元素的堆叠方法	3
line-stacking-shift	设置base-shift行中块元素包含元素的堆叠方法	3
line-stacking-strategy	设置内部包含块元素的堆叠线框的堆叠方法	3
text-height	行内框的文本内容区域设置block-progression维数	3

列表(List) 属性

属性	说明	CSS
list-style	在一个声明中设置所有的列表属性	1
list-style-image	将图象设置为列表项标记	1
list-style-position	设置列表项标记的放置位置	1
list-style-type	设置列表项标记的类型	1

外边距(Margin) 属性

属性	说明	CSS
margin	在一个声明中设置所有外边距属性	1
margin-bottom	设置元素的下外边距	1
margin-left	设置元素的左外边距	1
margin-right	设置元素的右外边距	1
margin-top	设置元素的上外边距	1

字幕(Marquee) 属性

属性	说明	CSS
marquee-direction	设置内容移动的方向	3
marquee-play-count	设置内容移动多少次	3
marquee-speed	设置内容滚动的速度有多快	3
marquee-style	设置内容移动的样式	3

多列(Multi-column) 属性

属性	说明	CSS
column-count	指定元素应该分为的列数	3
column-fill	指定如何填充列	3
column-gap	指定列之间的差距	3
column-rule	对于设置所有column-rule-*属性的简写属性	3
column-rule-color	指定列之间的颜色规则	3
column-rule-style	指定列之间的样式规则	3
column-rule-width	指定列之间的宽度规则	3
column-span	指定元素应该跨越多少列	3
column-width	指定列的宽度	3
columns	缩写属性设置列宽和列数	3

页面媒体(Paged Media) 属性

属性	说明	CSS
fit	如果其宽度和高度属性都不是auto给出一个提示，如何大规模替换元素	3
fit-position	判定方框内对象的对齐方式	3
image-orientation	指定用户代理适用于图像中的向右或顺时针方向的旋转	3
page	指定一个元素应显示的页面的特定类型	3
size	指定含有BOX的页面内容的大小和方位	3

定位（Positioning） 属性

属性	说明	CSS
bottom	设置定位元素下外边距边界与其包含块下边界之间的偏移	2
clear	规定元素的哪一侧不允许其他浮动元素	1
clip	剪裁绝对定位元素	2
cursor	规定要显示的光标的类型（形状）	2
display	规定元素应该生成的框的类型	1
float	规定框是否应该浮动	1
left	设置定位元素左外边距边界与其包含块左边界之间的偏移	2
overflow	规定当内容溢出元素框时发生的事情	2
position	规定元素的定位类型	2
right	设置定位元素右外边距边界与其包含块右边界之间的偏移	2
top	设置定位元素的上外边距边界与其包含块上边界之间的偏移	2
visibility	规定元素是否可见	2
z-index	设置元素的堆叠顺序	2

分页（Print） 属性

属性	说明	CSS
orphans	设置当元素内部发生分页时必须在页面底部保留的最少行数	2
page-break-after	设置元素后的分页行为	2
page-break-before	设置元素前的分页行为	2
page-break-inside	设置元素内部的分页行为	2
widows	设置当元素内部发生分页时必须在页面顶部保留的最少行数	2

Ruby 属性

属性	说明	CSS
ruby-align	控制Ruby文本和Ruby基础内容相对彼此的文本对齐方式	3
ruby-overhang	当Ruby文本超过Ruby的基础宽，确定ruby文本是否允许局部悬置任意相邻的文本，除了自己的基础	3
ruby-position	它的base控制Ruby文本的位置	3
ruby-span	控制annotation 元素的跨越行为	3

语音（Speech） 属性

属性	说明	CSS
mark	缩写属性设置mark-before和mark-after属性	3
mark-after	允许命名的标记连接到音频流	3

mark-before phonemes	允许命名的标记连接到音频流 指定包含文本的相应元素中的一个音标发音	3 3
rest	一个缩写属性设置rest-before和rest-after属性	3
rest-after	一个元素的内容讲完之后，指定要休息一下或遵守韵律边界	3
rest-before	一个元素的内容讲完之前，指定要休息一下或遵守韵律边界	3
voice-balance	指定了左，右声道之间的平衡	3
voice-duration	指定应采取呈现所选元素的内容的长度	3
voice-pitch	指定平均说话的声音的音调（频率）	3
voice-pitch-range	指定平均间距的变化	3
voice-rate	控制语速	3
voice-stress	指示着重力度	3
voice-volume	语音合成是指波形输出幅度	3

表格（Table） 属性

属性	说明	CSS
border-collapse	规定是否合并表格边框	2
border-spacing	规定相邻单元格边框之间的距离	2
caption-side	规定表格标题的位置	2
empty-cells	规定是否显示表格中的空单元格上的边框和背景	2
table-layout	设置用于表格的布局算法	2

文本（Text） 属性

属性	说明	CSS
color	设置文本的颜色	1
direction	规定文本的方向 / 书写方向	2
letter-spacing	设置字符间距	1
line-height	设置行高	1
text-align	规定文本的水平对齐方式	1
text-decoration	规定添加到文本的装饰效果	1
text-indent	规定文本块首行的缩进	1
text-transform	控制文本的大小写	1
unicode-bidi		2
vertical-align	设置元素的垂直对齐方式	1
white-space	设置怎样给一元素控件留白	1
word-spacing	设置单词间距	1
hanging-punctuation	指定一个标点符号是否可能超出行框	3
punctuation-trim	指定一个标点符号是否要去掉	3
text-align-last	当 text-align 设置为 justify 时，最后一行的对齐方式。	3
text-justify	当 text-align 设置为 justify 时指定分散对齐的方式。	3
text-outline	设置文字的轮廓。	3
text-overflow	指定当文本溢出包含的元素，应该发生什么	3
text-shadow	为文本添加阴影	3
text-wrap	指定文本换行规则	3
word-break	指定非CJK文字的断行规则	3
word-wrap	设置浏览器是否对过长的单词进行换行。	3

2D/3D 转换属性

属性	说明	CSS
transform	适用于2D或3D转换的元素	3
transform-origin	允许您更改转化元素位置	3
transform-style	3D空间中的指定如何嵌套元素	3
perspective	指定3D元素是如何查看透视图	3
perspective-origin	指定3D元素底部位置	3
backface-visibility	定义一个元素是否应该是可见的，不对着屏幕时	3

过渡（Transition） 属性

属性	说明	CSS
transition	此属性是 transition-property、transition-duration、transition-timing-function、transition-delay 的简写形式。	3
transition-property	设置用来进行过渡的 CSS 属性。	3
transition-duration	设置过渡进行的时间长度。	3
transition-timing-function	设置过渡进行的时序函数。	3
transition-delay	指定过渡开始的时间。	3

用户外观(User-interface) 属性

属性	说明	CSS
appearance	定义元素的外观样式	3
box-sizing	允许您为了适应区域以某种方式定义某些元素	3
icon	为元素指定图标	3
nav-down	指定用户按向下键时向下导航的位置	3
nav-index	指定导航（tab）顺序。	3
nav-left	指定用户按向左键时向左导航的位置	3
nav-right	指定用户按向右键时向左导航的位置	3
nav-up	指定用户按向上键时向上导航的位置a	3
outline-offset	设置轮廓框架在 border 边缘外的偏移	3
resize	定义元素是否可以改变大小	3

]

CSS 选择器

[«CSS 参考手册](#)
[CSS 语音参考»](#)

CSS选择器用于选择你想要的元素的样式的模式。

"CSS"列表示在CSS版本的属性定义（CSS1，CSS2，或对CSS3）。

选择器	示例	示例说明	CSS
.class	.intro	选择所有class="intro"的元素	1
#id	#firstname	选择所有id="firstname"的元素	1
*	*	选择所有元素	2
element	p	选择所有<p>元素	1
element.element	div,p	选择所有<div>元素和<p>元素	1
elementelement	div p	选择<div>元素内的所有<p>元素	1
element>element	div>p	选择所有<p>元素的父级<div>元素	2
element+element	div+p	选择所有紧接着<div>元素之后的<p>元素	2
[attribute]	[target]	选择所有带有target属性元素	2
[attribute=value]	[target=blank]	选择所有使用target="blank"的元素	2
[attribute~value]	[title~flower]	选择标题属性包含单词"flower"的所有元素	2
[attribute =language]	[lang=en]	选择一个lang属性的起始值="EN"的所有元素	2
:link	a:link	选择所有未访问链接	1
:visited	a:visited	选择所有访问过的链接	1
:active	a:active	选择活动链接	1
:hover	a:hover	选择鼠标在链接上面时	1
:focus	input:focus	选择具有焦点的输入元素	2
:first-letter	p:first-letter	选择每一个<P>元素的第一个字母	1
:first-line	p:first-line	选择每一个<P>元素的第一行	1
:first-child	p:first-child	指定只有当<p>元素是其父级的第一个子级的样式。	2
:before	p:before	在每个<p>元素之前插入内容	2
:after	p:after	在每个<p>元素之后插入内容	2
:lang(language)	p:lang(it)	选择一个lang属性的起始值="it"的所有<p>元素	2
element1~element2	p~ul	选择p元素之前的每一个ul元素	3
[attribute^=value]	a[src^="https"]	选择每一个src属性的值以"https"开头的元素	3
[attribute\$=value]	a[src\$=".pdf"]	选择每一个src属性的值以".pdf"结尾的元素	3
[attribute*=value]	a[src*="44lan"]	选择每一个src属性的值包含子字符串"44lan"的元素	3
:first-of-type	p:first-of-type	选择每个p元素是其父级的第一个p元素	3
:last-of-type	p:last-of-type	选择每个p元素是其父级的最后一个p元素	3
:only-of-type	p:only-of-type	选择每个p元素是其父级的唯一p元素	3
:only-child	p:only-child	选择每个p元素是其父级的唯一子元素	3
:nth-child(n)	p:nth-child(2)	选择每个p元素是其父级的第二个子元素	3
:nth-last-child(n)	p:nth-last-child(2)	选择每个p元素的是其父级的第二个子元素，从最后一个子项计数	3
:nth-of-type(n)	p:nth-of-type(2)	选择每个p元素是其父级的第二个p元素	3
:nth-last-of-type(n)	p:nth-last-of-type(2)	选择每个p元素的是其父级的第二个p元素，从最后一个子项计数	3
:last-child	p:last-child	选择每个p元素是其父级的最后一个子级。	3
:root	:root	选择文档的根元素	3
:empty	p:empty	选择每个没有任何子级的p元素（包括文本节点）	3
:target	#news:target	选择当前活动的#news元素（包含该锚名称的点击的URL）	3

:enabled	input:enabled	选择每一个已启用的输入元素	3
:disabled	input:disabled	选择每一个禁用的输入元素	3
:checked	input:checked	选择每个选中的输入元素	3
:not(selector)	:not(p)	选择每个并非p元素的元素	3
:selection	::selection	匹配元素中被用户选中或处于高亮状态的部分	3
:out-of-range	:out-of-range	匹配值在指定区间之外的input元素	3
:in-range	:in-range	匹配值在指定区间之内的input元素	3
:read-write	:read-write	用于匹配可读及可写的元素	3
:read-only	:read-only	用于匹配设置 "readonly" (只读) 属性的元素	3
:optional	:optional	用于匹配可选的输入元素	3
:required	:required	用于匹配设置了 "required" 属性的元素	3
:valid	:valid	用于匹配输入值为合法的元素	3
:invalid	:invalid	用于匹配输入值为非法的元素	3

]

[

CSS 语音参考

[«CSS 选择器](#)
[CSS Web安全字体 »](#)

听觉样式表使用了语音合成和声音效果的结合，让用户收听信息，而不是读取信息。

有声显示可用于：

- 失明人士
- 帮助用户学习阅读
- 帮助具有阅读问题的用户
- 家庭娱乐
- 在车上

听觉呈现通常会把文档转化为纯文本，然后传给屏幕阅读器（可读出屏幕上所有字符的一种程序）。

听觉样式表的一个例子：

```
h1,h2,h3,h4
{
voice-family:male;
richness:80;
cue-before:url("beep.au")
}
```

上面的例子用语音合成器播放声音，开头有一个男性的声音说话。

CSS 语音参考手册

CSS"列表示在CSS版本的属性定义（CSS1或CSS2）。

Property	Description	Values	CSS
azimuth	设置声音应该来自哪里	<i>angle</i>	2
		left-side	
		far-left	
		left	
		center-left	
		center	
		center-right	
		right	
		far-right	
		right-side	
		behind	
		leftwards	
cue	在一个声明中设置cue属性	<i>cue-before</i>	2
		<i>cue-after</i>	
cue-after	指定要播放的声音在一个元素的内容后面	none	2
		<i>url</i>	
cue-before	指定要播放的声音在一个元素的内容前面	none	2
		<i>url</i>	
		<i>angle</i> <i>below</i>	

elevation	设置声音应该来自哪里	level above higher lower	2
pause	在一个声明中设置pause属性	<i>pause-before</i> <i>pause-after</i>	2
pause-after	在一个元素的内容之后，指定暂停	<i>time</i> %	2
pause-before	在一个元素的内容之前，指定暂停	<i>time</i> %	2
pitch	指定讲话声音	<i>frequency</i> x-low low medium high x-high	2
pitch-range	指定讲话声音的变化。（单调的声音或动态的声音？）	<i>number</i> auto none	2
play-during	指定在读一个元素的内容时要播放的声音	<i>url</i> mix repeat	2
richness	指定丰富的讲话声音。（浑厚的声音或细的声音？）	<i>number</i> normal	2
speak	指定内容是否会提供听觉方式	none spell-out	2
speak-header	此属性设置或检索表格标题是在所有的单元格之前发声，还是到一个不与之关联的单元格就结束发声。	always once	2
speak-numeral	设置或检索数字如何发音。	digits continuous	2
speak-punctuation	设置或检索标点字符如何发音	none code <i>number</i> x-slow slow medium fast x-fast faster slower	2
speech-rate	指定发言速度	<i>number</i>	2
stress	讲话声音在指定的地方"重音"	<i>specific-voice</i> <i>generic-voice</i>	2
voice-family	设置或检索当前声音类型	<i>number</i> %	2
volume	指定发言的音量	silent x-soft soft medium loud x-loud	2

[

CSS Web安全字体

[«CSS 语音参考](#)
[CSS 单位 »](#)

常用的字体组合

font-family属性是多种字体的名称，作为一个"应变"制度，以确保浏览器/操作系统之间的最大兼容性。如果浏览器不支持的第一个字体，它尝试下一个的字体。

你想要的字体类型如果浏览器找不到，它会从通用的字体类型中找到与你相似的:

实例

p{font-family:"Times New Roman", Times, serif}

[尝试一下 »](#)

下面是一些常用的字体组合，通用的字体系列。

Serif 字体

字体

Georgia, serif

"Palatino Linotype", "Book Antiqua", Palatino, serif

"Times New Roman", Times, serif

文本示例

This is a heading

This is a paragraph

This is a heading

This is a paragraph

This is a heading

This is a paragraph

sans - serif字体

字体

Arial, Helvetica, sans-serif

Arial Black, Gadget, sans-serif

"Comic Sans MS", cursive, sans-serif

Impact, Charcoal, sans-serif

"Lucida Sans Unicode", "Lucida Grande", sans-serif

文本示例

This is a heading

This is a paragraph

This is a heading

This is a paragraph

This is a heading

This is a paragraph

This is a heading

This is a paragraph

This is a heading

This is a paragraph

Tahoma, Geneva, sans-serif

"Trebuchet MS", Helvetica, sans-serif

Verdana, Geneva, sans-serif

Monospace 字体

字体

"Courier New", Courier, monospace

"Lucida Console", Monaco, monospace

]

This is a heading

This is a paragraph

This is a heading

This is a paragraph

This is a heading

This is a paragraph

文本示例

This is a heading

This is a paragraph

This is a heading

This is a paragraph

[

CSS 单位

[«CSS Web安全字体](#)
[CSS 颜色 »](#)

测量单位值

单位	简介
%	百分比
in	英寸
cm	厘米
mm	毫米
em	1EM是等于当前字体大小。 2em意味着当前字体大小2倍。例如，如果一个元素是用12磅font-family显示，然后"EM"是24磅。 在CSS中"EM"是一个非常有用单位，因为它可以自动适应读者使用的font-family
ex	一个ex是字体的x高度（x -高度通常是大约一半的字体大小）
pt	1pt和1/72英寸相同
pc	1pc=12pt
px	像素(a dot on the computer screen)

]

CSS 颜色

[«CSS 单位](#)
[CSS 合法颜色值 »](#)









颜色是由红（RED），绿（GREEN），蓝（BLUE）光线的显示结合。

颜色值

CSS中定义颜色使用十六进制（hex）表示法为红，绿，蓝的颜色值结合。可以是最低值是0（十六进制00）到最高值是255（十六进制FF）

3个双位数字的十六进制值写法，以# 符号开始。

Color Examples

Color	Color HEX	Color RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

[尝试一下 »](#)

16万个不同的颜色

红，绿，蓝值从0到255的结合，给出了总额超过16万不同的颜色（256 × 256 ×256）。

现代大多数显示器能够显示至少16384种颜色。

如果你看看下面的颜色表，你会看到从0到255不同的红灯颜色。

要看到充满色彩混合时红灯从0到255变化，单击十六进制或RGB值。

红光	16进制值	RGB
	#000000	rgb(0,0,0)
	#080000	rgb(8,0,0)
	#100000	rgb(16,0,0)
	#180000	rgb(24,0,0)
	#200000	rgb(32,0,0)
	#280000	rgb(40,0,0)
	#300000	rgb(48,0,0)
	#380000	rgb(56,0,0)
	#400000	rgb(64,0,0)
	#480000	rgb(72,0,0)
	#500000	rgb(80,0,0)
		

	#580000	rgb(88,0,0)
	#600000	rgb(96,0,0)
	#680000	rgb(104,0,0)
	#700000	rgb(112,0,0)
	#780000	rgb(120,0,0)
	#800000	rgb(128,0,0)
	#880000	rgb(136,0,0)
	#900000	rgb(144,0,0)
	#980000	rgb(152,0,0)
	#A00000	rgb(160,0,0)
	#A80000	rgb(168,0,0)
	#B00000	rgb(176,0,0)
	#B80000	rgb(184,0,0)
	#C00000	rgb(192,0,0)
	#C80000	rgb(200,0,0)
	#D00000	rgb(208,0,0)
	#D80000	rgb(216,0,0)
	#E00000	rgb(224,0,0)
	#E80000	rgb(232,0,0)
	#F00000	rgb(240,0,0)
	#F80000	rgb(248,0,0)
	#FF0000	rgb(255,0,0)

灰阶

灰阶代表了由最暗到最亮之间不同亮度的层次级别,为了使您更容易选择合适的灰色，我们已编制了灰色色调的表：

Gray Shades	HEX	RGB
	#000000	rgb(0,0,0)
	#080808	rgb(8,8,8)
	#101010	rgb(16,16,16)
	#181818	rgb(24,24,24)
	#202020	rgb(32,32,32)
	#282828	rgb(40,40,40)
	#303030	rgb(48,48,48)
	#383838	rgb(56,56,56)
	#404040	rgb(64,64,64)
	#484848	rgb(72,72,72)
	#505050	rgb(80,80,80)
	#585858	rgb(88,88,88)
	#606060	rgb(96,96,96)
	#686868	rgb(104,104,104)
	#707070	rgb(112,112,112)
	#787878	rgb(120,120,120)
	#808080	rgb(128,128,128)
	#888888	rgb(136,136,136)
	#909090	rgb(144,144,144)
	#989898	rgb(152,152,152)
	#A0A0A0	rgb(160,160,160)
	#A8A8A8	rgb(168,168,168)
	#B0B0B0	rgb(176,176,176)
	#B8B8B8	rgb(184,184,184)

	#C0C0C0	rgb(192,192,192)
	#C8C8C8	rgb(200,200,200)
	#D0D0D0	rgb(208,208,208)
	#D8D8D8	rgb(216,216,216)
	#E0E0E0	rgb(224,224,224)
	#E8E8E8	rgb(232,232,232)
	#F0F0F0	rgb(240,240,240)
	#F8F8F8	rgb(248,248,248)
	#FFFFFF	rgb(255,255,255)

网络安全色？

若干年前，当计算机支持最大256个不同的颜色，一个216"网络安全颜色"列表被建议作为Web标准，保留了40个固定的系统颜色。

这在现在看来不是很重要，因为大多数计算机可以显示数百万种不同的色彩，但选择是留给你。

跨浏览器的调色板创建，以确保所有的计算机将显示正确的颜色，运行一个256色调色板：

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF

FFCC00 FFFF00	FFCC33 FFFF33	FFCC66 FFFF66	FFCC99 FFFF99	FFCCCC FFFFCC	FFCCFF FFFFFF
------------------	------------------	------------------	------------------	------------------	------------------

]

[

CSS 合法颜色值

[«CSS 颜色](#)
[CSS 颜色名称 »](#)

CSS Colors

CSS的颜色可以通过以下方法指定：

- 十六进制颜色
- RGB颜色
- RGBA颜色
- HSL色彩
- HSLA颜色
- 预定义/跨浏览器的颜色名称

十六进制颜色

所有主要浏览器都支持十六进制颜色值。

指定一个十六进制的颜色其组成部分是：#RRGGBB，其中RR（红色），GG（绿色）和BB（蓝色）。所有值必须介于0和FF之间。

实例

```
p
{
background-color:#ff0000;
}
```

[尝试一下 »](#)

RGB颜色

RGB颜色值在所有主要浏览器都支持。

RGB颜色值指定：RGB（红，绿，蓝）。每个参数（红色，绿色和蓝色）定义颜色的亮度，可在0和255之间，或一个百分比值（从0%到100%）之间的整数。

例如RGB（0,0,255）值呈现为蓝色，因为蓝色的参数设置为最高值（255）而其他设置为0。

此外，下面的值定义相同的颜色：RGB（0,0,255），RGB（0%，0%，100%）。

实例

```
p
{
background-color:rgb(255,0,0);
}
```

[尝试一下 »](#)

RGBA颜色

RGBA颜色值被IE9, Firefox3+, Chrome, Safari和Opera10+支持。

RGBA颜色值是RGB颜色值alpha通道的延伸 - 指定对象的透明度。

RGBA颜色值指定: RGBA (红, 绿, 蓝, alpha)。Alpha参数是一个介于0.0 (完全透明) 和1.0 (完全不透明) 之间的参数。

实例

```
p
{
background-color:rgba(255,0,0,0.5);
}
```

[尝试一下 »](#)

HSL颜色

IE9, Firefox, Chrome, Safari和Opera 10+.支持HSL颜色值。

HSL代表色相, 饱和度和亮度 - 使用色彩圆柱坐标表示。

HSL颜色值指定: HSL (色调, 饱和度, 明度)。

色相是在色轮上的程度 (从0到360) -0 (或360) 是红色的, 120是绿色的, 240是蓝色的。饱和度是一个百分比值;0%意味着灰色和100%的阴影, 是全彩。亮度也是一个百分点;0%是黑色的, 100%是白色的。

实例

```
p
{
background-color:hsl(120,65%,75%);
}
```

[尝试一下 »](#)

HSLA颜色

HSLA颜色值被IE9, Firefox3+, Chrome, Safari和Opera10+.支持。

HSLA的颜色值是一个带有alpha通道的HSL颜色值的延伸 - 指定对象的透明度。

指定HSLA颜色值: HSLA (色调, 饱和度, 亮度, α), α 是Alpha参数定义的不透明度。Alpha参数是一个介于0.0 (完全透明) 和1.0 (完全不透明) 之间的参数。

实例

```
p
{
background-color:hsla(120,65%,75%,0.3);
}
```

[尝试一下 »](#)

预定义/跨浏览器的颜色名称

147是在HTML和CSS颜色规范预定义的颜色名称。你可以查看我们的[预定义颜色名称表](#)。

]

CSS 颜色名称

[«CSS 合法颜色值](#)
[CSS 颜色十六进制值 »](#)

所有浏览器都支持颜色名称

147颜色名称定义在HTML和CSS的颜色规格(17个标准色加上130多个其他)。下表列出了所有这些，连同其十六进制值。









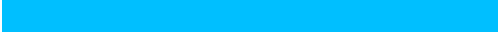

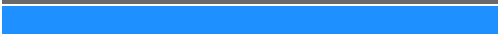





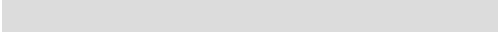
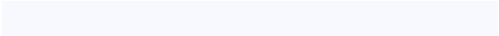












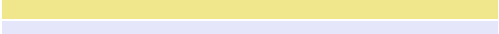
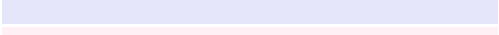





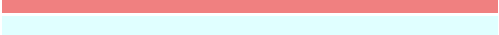
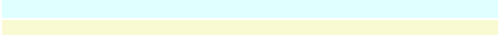











💡提示: 17种标准颜色: 浅绿色, 黑色, 蓝色, 紫红色, 灰色, 灰色, 绿色, 石灰, 栗色, 海军, 橄榄, 紫, 红, 银, 蓝绿色, 白色和黄色。







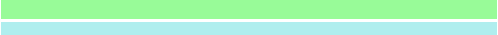





点击颜色的名称或十六进制值，伴随着不同的颜色名称或值其背景颜色会改变:

按颜色名称排序

[列表按十六进制值进行排序](#)

Color Name	HEX	Color
AliceBlue	#F0F8FF	
AntiqueWhite	#FAEBD7	
Aqua	#00FFFF	
Aquamarine	#7FFFD4	
Azure	#F0FFFF	
Beige	#F5F5DC	
Bisque	#FFE4C4	
Black	#000000	
BlanchedAlmond	#FFEBCD	
Blue	#0000FF	
BlueViolet	#8A2BE2	
Brown	#A52A2A	
BurlyWood	#DEB887	
CadetBlue	#5F9EA0	
Chartreuse	#7FFF00	
Chocolate	#D2691E	
Coral	#FF7F50	
CornflowerBlue	#6495ED	
Cornsilk	#FFF8DC	
Crimson	#DC143C	
Cyan	#00FFFF	
DarkBlue	#00008B	
DarkCyan	#008B8B	
DarkGoldenRod	#B8860B	
DarkGray	#A9A9A9	
DarkGreen	#006400	
DarkKhaki	#BDB76B	
DarkMagenta	#8B008B	
DarkOliveGreen	#556B2F	
DarkOrange	#FF8C00	
DarkOrchid	#9932CC	
DarkRed	#8B0000	

DarkSalmon	#E9967A	
DarkSeaGreen	#8FBC8F	
DarkSlateBlue	#483D8B	
DarkSlateGray	#2F4F4F	
DarkTurquoise	#00CED1	
DarkViolet	#9400D3	
DeepPink	#FF1493	
DeepSkyBlue	#00BFFF	
DimGray	#696969	
DodgerBlue	#1E90FF	
FireBrick	#B22222	
FloralWhite	#FFFAF0	
ForestGreen	#228B22	
Fuchsia	#FF00FF	
Gainsboro	#DCDCDC	
GhostWhite	#F8F8FF	
Gold	#FFD700	
GoldenRod	#DAA520	
Gray	#808080	
Green	#008000	
GreenYellow	#ADFF2F	
HoneyDew	#F0FFF0	
HotPink	#FF69B4	
IndianRed	#CD5C5C	
Indigo	#4B0082	
Ivory	#FFFFFF	
Khaki	#F0E68C	
Lavender	#E6E6FA	
LavenderBlush	#FFF0F5	
LawnGreen	#7CFC00	
LemonChiffon	#FFFACD	
LightBlue	#ADD8E6	
LightCoral	#F08080	
LightCyan	#E0FFFF	
LightGoldenRodYellow	#FAFAD2	
LightGray	#D3D3D3	
LightGreen	#90EE90	
LightPink	#FFB6C1	
LightSalmon	#FFA07A	
LightSeaGreen	#20B2AA	
LightSkyBlue	#87CEFA	
LightSlateGray	#778899	
LightSteelBlue	#B0C4DE	
LightYellow	#FFFFE0	
Lime	#00FF00	
LimeGreen	#32CD32	
Linen	#FAF0E6	
Magenta	#FF00FF	
Maroon	#800000	
MediumAquaMarine	#66CDAA	

MediumBlue	#0000CD	
MediumOrchid	#BA55D3	
MediumPurple	#9370DB	
MediumSeaGreen	#3CB371	
MediumSlateBlue	#7B68EE	
MediumSpringGreen	#00FA9A	
MediumTurquoise	#48D1CC	
MediumVioletRed	#C71585	
MidnightBlue	#191970	
MintCream	#F5FFFA	
MistyRose	#FFE4E1	
Moccasin	#FFE4B5	
NavajoWhite	#FFDEAD	
Navy	#000080	
OldLace	#FDF5E6	
Olive	#808000	
OliveDrab	#6B8E23	
Orange	#FFA500	
OrangeRed	#FF4500	
Orchid	#DA70D6	
PaleGoldenRod	#EEE8AA	
PaleGreen	#98FB98	
PaleTurquoise	#AFEEEE	
PaleVioletRed	#DB7093	
PapayaWhip	#FFEFD5	
PeachPuff	#FFDAB9	
Peru	#CD853F	
Pink	#FFC0CB	
Plum	#DDA0DD	
PowderBlue	#B0E0E6	
Purple	#800080	
Red	#FF0000	
RosyBrown	#BC8F8F	
RoyalBlue	#4169E1	
SaddleBrown	#8B4513	
Salmon	#FA8072	
SandyBrown	#F4A460	
SeaGreen	#2E8B57	
SeaShell	#FFF5EE	
Sienna	#A0522D	
Silver	#C0C0C0	
SkyBlue	#87CEEB	
SlateBlue	#6A5ACD	
SlateGray	#708090	
Snow	#FFFAFA	
SpringGreen	#00FF7F	
SteelBlue	#4682B4	
Tan	#D2B48C	
Teal	#008080	
Thistle	#D8BFD8	
		

<u>Tomato</u>	<u>#FF6347</u>	
<u>Turquoise</u>	<u>#40E0D0</u>	
<u>Violet</u>	<u>#EE82EE</u>	
<u>Wheat</u>	<u>#F5DEB3</u>	
<u>White</u>	<u>#FFFFFF</u>	
<u>WhiteSmoke</u>	<u>#F5F5F5</u>	
<u>Yellow</u>	<u>#FFFF00</u>	
<u>YellowGreen</u>	<u>#9ACD32</u>	

]

[

CSS 颜色十六进制值

«CSS 颜色名称
CSS 浏览器支持 »

按十六进制值排序

列表按颜色名称进行排序

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	
DarkGreen	#006400	
Green	#008000	
Teal	#008080	
DarkCyan	#008B8B	
DeepSkyBlue	#00BFFF	
DarkTurquoise	#00CED1	
MediumSpringGreen	#00FA9A	
Lime	#00FF00	
SpringGreen	#00FF7F	
Aqua	#00FFFF	
Cyan	#00FFFF	
MidnightBlue	#191970	
DodgerBlue	#1E90FF	
LightSeaGreen	#20B2AA	
ForestGreen	#228B22	
SeaGreen	#2E8B57	
DarkSlateGray	#2F4F4F	
LimeGreen	#32CD32	
MediumSeaGreen	#3CB371	
Turquoise	#40E0D0	
RoyalBlue	#4169E1	
SteelBlue	#4682B4	
DarkSlateBlue	#483D8B	
MediumTurquoise	#48D1CC	
Indigo	#4B0082	
DarkOliveGreen	#556B2F	
CadetBlue	#5F9EA0	
CornflowerBlue	#6495ED	
MediumAquaMarine	#66CDAA	
DimGray	#696969	
SlateBlue	#6A5ACD	
OliveDrab	#6B8E23	
SlateGray	#708090	
LightSlateGray	#778899	
MediumSlateBlue	#7B68EE	

LawnGreen	#7CFC00	
Chartreuse	#7FFF00	
Aquamarine	#7FFFD4	
Maroon	#800000	
Purple	#800080	
Olive	#808000	
Gray	#808080	
SkyBlue	#87CEEB	
LightSkyBlue	#87CEFA	
BlueViolet	#8A2BE2	
DarkRed	#8B0000	
DarkMagenta	#8B008B	
SaddleBrown	#8B4513	
DarkSeaGreen	#8FBC8F	
LightGreen	#90EE90	
MediumPurple	#9370DB	
DarkViolet	#9400D3	
PaleGreen	#98FB98	
DarkOrchid	#9932CC	
YellowGreen	#9ACD32	
Sienna	#A0522D	
Brown	#A52A2A	
DarkGray	#A9A9A9	
LightBlue	#ADD8E6	
GreenYellow	#ADFF2F	
PaleTurquoise	#AFEEEE	
LightSteelBlue	#B0C4DE	
PowderBlue	#B0E0E6	
FireBrick	#B22222	
DarkGoldenRod	#B8860B	
MediumOrchid	#BA55D3	
RosyBrown	#BC8F8F	
DarkKhaki	#BDB76B	
Silver	#C0C0C0	
MediumVioletRed	#C71585	
IndianRed	#CD5C5C	
Peru	#CD853F	
Chocolate	#D2691E	
Tan	#D2B48C	
LightGray	#D3D3D3	
Thistle	#D8BFD8	
Orchid	#DA70D6	
GoldenRod	#DAA520	
PaleVioletRed	#DB7093	
Crimson	#DC143C	
Gainsboro	#DCDCDC	
Plum	#DDA0DD	
BurlyWood	#DEB887	
LightCyan	#E0FFFF	
Lavender	#E6E6FA	
DarkSalmon	#E9967A	

Violet	#EE82EE	
PaleGoldenRod	#EEE8AA	
LightCoral	#F08080	
Khaki	#F0E68C	
AliceBlue	#F0F8FF	
HoneyDew	#F0FFF0	
Azure	#F0FFFF	
SandyBrown	#F4A460	
Wheat	#F5DEB3	
Beige	#F5F5DC	
WhiteSmoke	#F5F5F5	
MintCream	#F5FFFA	
GhostWhite	#F8F8FF	
Salmon	#FA8072	
AntiqueWhite	#FAEBD7	
Linen	#FAF0E6	
LightGoldenRodYellow	#FAFAD2	
OldLace	#FDF5E6	
Red	#FF0000	
Fuchsia	#FF00FF	
Magenta	#FF00FF	
DeepPink	#FF1493	
OrangeRed	#FF4500	
Tomato	#FF6347	
HotPink	#FF69B4	
Coral	#FF7F50	
DarkOrange	#FF8C00	
LightSalmon	#FFA07A	
Orange	#FFA500	
LightPink	#FFB6C1	
Pink	#FFC0CB	
Gold	#FFD700	
PeachPuff	#FFDAB9	
NavajoWhite	#FFDEAD	
Moccasin	#FFE4B5	
Bisque	#FFE4C4	
MistyRose	#FFE4E1	
BlanchedAlmond	#FFEBCD	
PapayaWhip	#FFEFD5	
LavenderBlush	#FFF0F5	
SeaShell	#FFF5EE	
Cornsilk	#FFF8DC	
LemonChiffon	#FFFACD	
FloralWhite	#FFFAF0	
Snow	#FFFAFA	
Yellow	#FFFF00	
LightYellow	#FFFFE0	
Ivory	#FFFFF0	
White	#FFFFFF	