**什么是 CSS?**

* CSS 指层叠样式表 (**C**ascading **S**tyle **S**heets)
* 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

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。

<style>

h2{color: red} 对元素定义样式

</style>

## 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;  
}

# 选择器

## CSS 元素选择器

最常见的 CSS 选择器是元素选择器。换句话说，文档的元素就是最基本的选择器。

如果设置 HTML 的样式，选择器通常将是某个 HTML 元素，比如 p、h1、em、a，甚至可以是 html 本身：

html {color:black;}

h1 {color:blue;}

h2 {color:silver;}

### 选择器分组

假设希望 h2 元素和段落都有灰色。为达到这个目的，最容易的做法是使用以下声明：

h2, p {color:gray;}

将 h2 和 p 选择器放在规则左边，然后用逗号分隔，就定义了一个规则。其右边的样式（color:gray;）将应用到这两个选择器所引用的元素。逗号告诉浏览器，规则中包含两个不同的选择器。如果没有这个逗号，那么规则的含义将完全不同。参见后代选择器。

### 通配符选择器

CSS2 引入了一种新的简单选择器 - 通配选择器（universal selector），显示为一个星号（\*）。该选择器可以与任何元素匹配，就像是一个通配符。

例如，下面的规则可以使文档中的每个元素都为红色：

\* {color:red;} 相当于 html{color:red;}

## 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;}

我们先声明一个类选择器：

<style>

.blue-text {

color: blue;

}

</style>

上面的代码在 <style> 标记中声明了一个叫做 blue-text 的类样式。

然后在h2元素上应用我们声明的类选择器：

<h2 class="blue-text">CatPhotoApp</h2>

注意：在CSS中，类选择器应该添加.为前缀。

而在HTML中，class属性不能添加.为前缀。

创建一个名为larger-image的类选择器，把HTML元素的宽度设定为500像素，我们使用：

<style>

.larger-image {

width: 500px;

}

</style>

多个class到一个元素，只需要在多个class之间用空格分开即可。例如：

<img class="class1 class2">

边框颜色为红色、边框宽度为5像素(px)、边框样式为固体(solid)，代码如下:

<style>  
  .thin-red-border {  
    border-color: red;  
    border-width: 5px;  
    border-style: solid;

border-radius: 10px; border-radius: 50%使用百分比来指定border-radius边框半径的值。50是圆形  
  }  
</style>

有时你想为你的网站添加一个a元素，但此时你还不知道要将它们链接到哪儿，此时可以使用固定链接。

把你的a元素的href属性的值替换为一个#，别名hash(哈希)符号，将其变为一个固定链接。

Href=”#”

<a href="#"><img src="/images/relaxing-cat.jpg"></a>点击图片超链接

body {background-color:yellow;}

h1 {font-size:36pt;}

h2 {color:blue;}

p {margin-left:150px;}

链接 点击前后 覆盖 激活

a:link {color:green;}

a:visited {color:yellow;}

a:hover {color:black;}

a:active {color:blue;}

|  |  |  |
| --- | --- | --- |
| **选择符** | **选择的内容** | **示例** |
| 元素 选择符（有时也称作标签或类型选择符） | 所有指定的 HTML 元素 | p 选择 <p> |
| ID 选择符 | 指定 ID 的元素（在 HTML 页面中，每个 ID 只被允许指定一次） | #my-id 选择 <p id="my-id"> 或 <a id="my-id"> |
| Class 选择符 | 指定 Class 的元素（多个 Class 可以同时出现在一个页面中） | .my-class 选择 <p class="my-class">和 <a class="my-class"> |
| 属性 选择符 | 页面上拥有指定属性的元素 | img[src] 选择 <img src="myimage.png"> 而不是 <img> |
| 伪类 选择符 | 指定的元素，但是需要在特殊的状态，比如悬停 | a:hover 选择 <a>, 但是只在鼠标悬停在链接上时 |

### 结合元素选择器

类选择器可以结合元素选择器来使用。

例如，您可能希望只有段落显示为红色文本：

p.important {color:red;}

选择器现在会匹配 class 属性包含 important 的所有 p 元素，但是其他任何类型的元素都不匹配，不论是否有此 class 属性。选择器 p.important 解释为：“其 class 属性值为 important 的所有段落”。 因为 h1 元素不是段落，这个规则的选择器与之不匹配，因此 h1 元素不会变成红色文本。

如果你确实希望为 h1 元素指定不同的样式，可以使用选择器 h1.important：

p.important {color:red;}

h1.important {color:blue;}

通过把两个类选择器链接在一起，仅可以选择同时包含这些类名的元素（类名的顺序不限）。

如果一个多类选择器包含类名列表中没有的一个类名，匹配就会失败。请看下面的规则：

.important.urgent {background:silver;}

不出所料，这个选择器将只匹配 class 属性中包含词 important 和 urgent 的 p 元素。因此，如果一个 p 元素的 class 属性中只有词 important 和 warning，将不能匹配。不过，它能匹配以下元素：

<p class="important urgent warning">

This paragraph is a very important and urgent warning.

</p>

## 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;  
}

* ID属性不要以数字开头

## 类选择器还是 ID 选择器？

在类选择器这一章中我们曾讲解过，可以为任意多个元素指定类。前一章中类名 important 被应用到 p 和 h1 元素，而且它还可以应用到更多元素。

### 区别 1：只能在文档中使用一次

与类不同，在一个 HTML 文档中，ID 选择器会使用一次，而且仅一次。

### 区别 2：不能使用 ID 词列表

不同于类选择器，ID 选择器不能结合使用，因为 ID 属性不允许有以空格分隔的词列表。

### 区别 3：ID 能包含更多含义

类似于类，可以独立于元素来选择 ID。有些情况下，您知道文档中会出现某个特定 ID 值，但是并不知道它会出现在哪个元素上，所以您想声明独立的 ID 选择器。例如，您可能知道在一个给定的文档中会有一个 ID 值为 mostImportant 的元素。您不知道这个最重要的东西是一个段落、一个短语、一个列表项还是一个小节标题。您只知道每个文档都会有这么一个最重要的内容，它可能在任何元素中，而且只能出现一个。在这种情况下，可以编写如下规则：

#mostImportant {color:red; background:yellow;}

这个规则会与以下各个元素匹配（这些元素不能在同一个文档中同时出现，因为它们都有相同的 ID 值）：

<h1 id="mostImportant">This is important!</h1>

<em id="mostImportant">This is important!</em>

<ul id="mostImportant">This is important!</ul>

# CSS 属性选择器详解

**CSS 2 引入了属性选择器。**

**属性选择器可以根据元素的属性及属性值来选择元素。**

## 简单属性选择

如果希望选择有某个属性的元素，而不论属性值是什么，可以使用简单属性选择器。

\*[……] {…:…;}

### 例子 1

所有包含某属性

如果您希望把包含标题（title）的所有元素变为红色，可以写作：

\*[title] {color:red;}

[亲自试一试](mk:@MSITStore:F:\\qqfile\\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_att)

### 例子 2

某类元素包含某属性

与上面类似，可以只对有 href 属性的锚（a 元素）应用样式：

a[href] {color:red;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_att_2)

### 例子 3

还可以根据多个属性进行选择，只需将属性选择器链接在一起即可。

例如，为了将同时有 href 和 title 属性的 HTML 超链接的文本设置为红色，可以这样写：

a[href][title] {color:red;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_att_3)

### 例子 4

可以采用一些创造性的方法使用这个特性。

例如，可以对所有带有 alt 属性的图像应用样式，从而突出显示这些有效的图像：

img[alt] {border: 5px solid red;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_att_4)

提示：上面这个特例更适合用来诊断而不是设计，即用来确定图像是否确实有效。

### 例子 5：为 XML 文档使用属性选择器

属性选择器在 XML 文档中相当有用，因为 XML 语言主张要针对元素和属性的用途指定元素名和属性名。

假设我们为描述太阳系行星设计了一个 XML 文档。如果我们想选择有 moons 属性的所有 planet 元素，使之显示为红色，以便能更关注有 moons 的行星，就可以这样写：

planet[moons] {color:red;}

这会让以下标记片段中的第二个和第三个元素的文本显示为红色，但第一个元素的文本不是红色：

<planet>Venus</planet>

<planet moons="1">Earth</planet>

<planet moons="2">Mars</planet>

[查看效果](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/css/planetml_attselector_att.xml)

## 根据具体属性值选择

除了选择拥有某些属性的元素，还可以进一步缩小选择范围，只选择有特定属性值的元素。

### 例子 1

例如，假设希望将指向 Web 服务器上某个指定文档的超链接变成红色，可以这样写：

a[href="http://www.w3school.com.cn/about\_us.asp"] {color: red;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_value_1)

### 例子 2

与简单属性选择器类似，可以把多个属性-值选择器链接在一起来选择一个文档。

a[href="http://www.w3school.com.cn/"][title="W3School"] {color: red;}

这会把以下标记中的第一个超链接的文本变为红色，但是第二个或第三个链接不受影响：

<a href="http://www.w3school.com.cn/" title="W3School">W3School</a>

<a href="http://www.w3school.com.cn/css/" title="CSS">CSS</a>

<a href="http://www.w3school.com.cn/html/" title="HTML">HTML</a>

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_value_2)

### 例子 3

同样地，XML 语言也可以利用这种方法来设置样式。

下面我们再回到行星那个例子中。假设只希望选择 moons 属性值为 1 的那些 planet 元素：

planet[moons="1"] {color: red;}

上面的代码会把以下标记中的第二个元素变成红色，但第一个和第三个元素不受影响：

<planet>Venus</planet>

<planet moons="1">Earth</planet>

<planet moons="2">Mars</planet>

[查看效果](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/css/planetml_attselector_val.xml)

### 属性与属性值必须完全匹配

请注意，这种格式要求必须与属性值完全匹配。

如果属性值包含用空格分隔的值列表，匹配就可能出问题。

请考虑一下的标记片段：

<p class="important warning">This paragraph is a very important warning.</p>

如果写成 p[class="important"]，那么这个规则不能匹配示例标记。

要根据具体属性值来选择该元素，必须这样写：

p[class="important warning"] {color: red;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_value_3)

## 根据部分属性值选择

如果需要根据属性值中的词列表的某个词进行选择，则需要使用波浪号（~）。

假设您想选择 class 属性中包含 important 的元素，可以用下面这个选择器做到这一点：

p[class~="important"] {color: red;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_valuelist_space_1)

如果忽略了波浪号，则说明需要完成完全值匹配。

### 部分值属性选择器与点号类名记法的区别

该选择器等价于我们在类选择器中讨论过的点号类名记法。

也就是说，p.important 和 p[class="important"] 应用到 HTML 文档时是等价的。

那么，为什么还要有 "~=" 属性选择器呢？因为它能用于任何属性，而不只是 class。

例如，可以有一个包含大量图像的文档，其中只有一部分是图片。对此，可以使用一个基于 title 文档的部分属性选择器，只选择这些图片：

img[title~="Figure"] {border: 1px solid gray;}

这个规则会选择 title 文本包含 "Figure" 的所有图像。没有 title 属性或者 title 属性中不包含 "Figure" 的图像都不会匹配。

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_valuelist_space_2)

### 子串匹配属性选择器

下面为您介绍一个更高级的选择器模块，它是 CSS2 完成之后发布的，其中包含了更多的部分值属性选择器。按照规范的说法，应该称之为“子串匹配属性选择器”。

很多现代浏览器都支持这些选择器，包括 IE7。

下表是对这些选择器的简单总结：

|  |  |
| --- | --- |
| 类型 | 描述 |
| [abc^="def"] | 选择 abc 属性值以 "def" 开头的所有元素 |
| [abc$="def"] | 选择 abc 属性值以 "def" 结尾的所有元素 |
| [abc\*="def"] | 选择 abc 属性值中包含子串 "def" 的所有元素 |

可以想到，这些选择有很多用途。

举例来说，如果希望对指向 W3School 的所有链接应用样式，不必为所有这些链接指定 class，再根据这个类编写样式，而只需编写以下规则：

a[href\*="w3school.com.cn"] {color: red;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_substring_matching)

提示：任何属性都可以使用这些选择器。

## 特定属性选择类型

最后为您介绍特定属性选择器。请看下面的例子：

\*[lang|="en"] {color: red;}

上面这个规则会选择 lang 属性等于 en 或以 en- 开头的所有元素。因此，以下示例标记中的前三个元素将被选中，而不会选择后两个元素：

<p lang="en">Hello!</p>

<p lang="en-us">Greetings!</p>

<p lang="en-au">G'day!</p>

<p lang="fr">Bonjour!</p>

<p lang="cy-en">Jrooana!</p>

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_valuelist_hyphen_1)

一般来说，[att|="val"] 可以用于任何属性及其值。

假设一个 HTML 文档中有一系列图片，其中每个图片的文件名都形如 *figure-1.jpg* 和 *figure-2.jpg*。就可以使用以下选择器匹配所有这些图像：

img[src|="figure"] {border: 1px solid gray;}

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_attribute_valuelist_hyphen_2)

当然，这种属性选择器最常见的用途还是匹配语言值。

# CSS 后代选择器

**后代选择器（descendant selector）又称为包含选择器。**

**后代选择器可以选择作为某元素后代的元素。**

## 根据上下文选择元素

我们可以定义后代选择器来创建一些规则，使这些规则在某些文档结构中起作用，而在另外一些结构中不起作用。

举例来说，如果您希望只对 h1 元素中的 em 元素应用样式，可以这样写：

h1 em {color:red;} 空格，子类

上面这个规则会把作为 h1 元素后代的 em 元素的文本变为 红色。其他 em 文本（如段落或块引用中的 em）则不会被这个规则选中：

<h1>This is a <em>important</em> heading</h1>

<p>This is a <em>important</em> paragraph.</p>

[亲自试一试](mk:@MSITStore:F:\qqfile\W3School离线手册(2016.07.15).chm::/www.w3school.com.cn/tiy/t.asp@f=csse_selector_descendant_1)

当然，您也可以在 h1 中找到的每个 em 元素上放一个 class 属性，但是显然，后代选择器的效率更高。

## 语法解释

在后代选择器中，规则左边的选择器一端包括两个或多个用空格分隔的选择器。选择器之间的空格是一种结合符（combinator）。每个空格结合符可以解释为“... 在 ... 找到”、“... 作为 ... 的一部分”、“... 作为 ... 的后代”，但是要求必须从右向左读选择器。

因此，h1 em 选择器可以解释为 “作为 h1 元素后代的任何 em 元素”。如果要从左向右读选择器，可以换成以下说法：“包含 em 的所有 h1 会把以下样式应用到该 em”。

## 具体应用

后代选择器的功能极其强大。有了它，可以使 HTML 中不可能实现的任务成为可能。

假设有一个文档，其中有一个边栏，还有一个主区。边栏的背景为蓝色，主区的背景为白色，这两个区都包含链接列表。不能把所有链接都设置为蓝色，因为这样一来边栏中的蓝色链接都无法看到。

解决方法是使用后代选择器。在这种情况下，可以为包含边栏的 div 指定值为 sidebar 的 class 属性，并把主区的 class 属性值设置为 maincontent。然后编写以下样式：

div.sidebar {background:blue;}

div.maincontent {background:white;}

div.sidebar a:link {color:white;}

div.maincontent a:link {color:blue;}

有关后代选择器有一个易被忽视的方面，即两个元素之间的层次间隔可以是无限的。

例如，如果写作 ul em，这个语法就会选择从 ul 元素继承的所有 em 元素，而不论 em 的嵌套层次多深。

因此，ul em 将会选择以下标记中的所有 em 元素：

<ul>

<li>List item 1

<ol>

<li>List item 1-1</li>

<li>List item 1-2</li>

<li>List item 1-3

<ol>

<li>List item 1-3-1</li>

<li>List item <em>1-3-2</em></li>

<li>List item 1-3-3</li>

</ol>

</li>

<li>List item 1-4</li>

</ol>

</li>

<li>List item 2</li>

<li>List item 3</li>

</ul>

# CSS 子元素选择器

**与后代选择器相比，子元素选择器（Child selectors）只能选择作为某元素子元素的元素。**

## 选择子元素

如果您不希望选择任意的后代元素，而是希望缩小范围，只选择某个元素的子元素，请使用子元素选择器（Child selector）。

例如，如果您希望选择只作为 h1 元素子元素的 strong 元素，可以这样写：

h1 > strong {color:red;}

这个规则会把第一个 h1 下面的两个 strong 元素变为红色，但是第二个 h1 中的 strong 不受影响：

所写的规则只针对 h1的下一级strong，不包括下下一级strong

<h1>This is <strong>very</strong> <strong>very</strong> important.</h1>

<h1>This is <em>really <strong>very</strong></em> important.</h1>

## 结合后代选择器和子选择器

请看下面这个选择器：

table.company td > p

上面的选择器会选择作为 td 元素子元素的所有 p 元素，这个 td 元素本身从 table 元素继承，该 table 元素有一个包含 company 的 class 属性。

# CSS 相邻兄弟选择器

**相邻兄弟选择器（Adjacent sibling selector）可选择紧接在另一元素后的元素，且二者有相同父元素。**

## 选择相邻兄弟

如果需要选择紧接在另一个元素后的元素，而且二者有相同的父元素，可以使用相邻兄弟选择器（Adjacent sibling selector）。

例如，如果要增加紧接在 h1 元素后出现的段落的上边距，可以这样写：

h1 + p {margin-top:50px;}

括号里面的设定应用于 h1 后第一个p之后。

# CSS 伪类 (Pseudo-classes)

**CSS 伪类用于向某些选择器添加特殊的效果。**

## 语法

伪类的语法：

selector : pseudo-class {property: value}

CSS 类也可与伪类搭配使用。

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

## 锚伪类

在支持 CSS 的浏览器中，链接的不同状态都可以不同的方式显示，这些状态包括：活动状态，已被访问状态，未被访问状态，和鼠标悬停状态。

a:link {color: #FF0000} /\* 未访问的链接 \*/

a:visited {color: #00FF00} /\* 已访问的链接 \*/

a:hover {color: #FF00FF} /\* 鼠标移动到链接上 \*/

a:active {color: #0000FF} /\* 选定的链接 \*/

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

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

**提示：**伪类名称对大小写不敏感。

## 伪类与 CSS 类

伪类可以与 CSS 类配合使用：

a.red : visited {color: #FF0000}

<a class="red" href="css\_syntax.asp">CSS Syntax</a>

假如上面的例子中的链接被访问过，那么它将显示为红色。

## CSS2 - :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。这个特定伪类很容易遭到误解，所以有必要举例来说明。考虑以下标记：

<div>

<p>These are the necessary steps:</p>

<ul>

<li>Intert Key</li>

<li>Turn key <strong>clockwise</strong></li>

<li>Push accelerator</li>

</ul>

<p>Do <em>not</em> push the brake at the same time as the accelerator.</p>

</div>

在上面的例子中，作为第一个元素的元素包括第一个 p、第一个 li 和 strong 和 em 元素。

给定以下规则：

p:first-child {font-weight: bold;}

li:first-child {text-transform:uppercase;}

第一个规则将作为某元素第一个子元素的所有 p 元素设置为粗体。第二个规则将作为某个元素（在 HTML 中，这肯定是 ol 或 ul 元素）第一个子元素的所有 li 元素变成大写。

**提示：**最常见的错误是认为 p:first-child 之类的选择器会选择 p 元素的第一个子元素。

**注释：**必须声明 [<!DOCTYPE>](http://www.w3school.com.cn/tags/tag_doctype.asp)，这样 :first-child 才能在 IE 中生效。

为了使您更透彻地理解 :first-child 伪类，我们另外提供了 3 个例子：

### 例子 1 - 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：

<html>

<head>

<style type="text/css">

p:first-child {

color: red;

}

</style>

</head>

<body>

<p>some text</p>

<p>some text</p>

</body>

</html>

[TIY](http://www.w3school.com.cn/tiy/t.asp?f=csse_first-child_1)

### 例子 2 - 匹配所有 <p> 元素中的第一个 <i> 元素

在下面的例子中，选择器匹配所有 <p> 元素中的第一个 <i> 元素：

<html>

<head>

<style type="text/css">

p > i:first-child {

font-weight:bold;

}

</style>

</head>

<body>

<p>some <i>text</i>. some <i>text</i>.</p>

<p>some <i>text</i>. some <i>text</i>.</p>

</body>

</html>

[TIY](http://www.w3school.com.cn/tiy/t.asp?f=csse_first-child_2)

### 例子 3 - 匹配所有作为第一个子元素的 <p> 元素中的所有 <i> 元素

在下面的例子中，选择器匹配所有作为元素的第一个子元素的 <p> 元素中的所有 <i> 元素：

<html>

<head>

<style type="text/css">

p:first-child i { //这里冒号和后面不能有空格

color:blue;

}

</style>

</head>

<body>

<p>some <i>text</i>. some <i>text</i>.</p>

<p>some <i>text</i>. some <i>text</i>.</p>

</body>

</html>

[TIY](http://www.w3school.com.cn/tiy/t.asp?f=csse_first-child_3)

## CSS2 - :lang 伪类

:lang 伪类使你有能力为不同的语言定义特殊的规则。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：

<html>

<head>

<style type="text/css">

q:lang(no)

{

quotes: "~" "~"

}

</style>

</head>

<body>

<p>文字<q lang="no">段落中的引用的文字</q>文字</p>

</body></html>

## 伪类

**W3C**："W3C" 列指示出该属性在哪个 CSS 版本中定义（CSS1 还是 CSS2）。

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [:active](http://www.w3school.com.cn/cssref/pr_pseudo_active.asp) | 向被激活的元素添加样式。 | 1 |
| [:focus](http://www.w3school.com.cn/cssref/pr_pseudo_focus.asp) | 向拥有键盘输入焦点的元素添加样式。 | 2 |
| [:hover](http://www.w3school.com.cn/cssref/pr_pseudo_hover.asp) | 当鼠标悬浮在元素上方时，向元素添加样式。 | 1 |
| [:link](http://www.w3school.com.cn/cssref/pr_pseudo_link.asp) | 向未被访问的链接添加样式。 | 1 |
| [:visited](http://www.w3school.com.cn/cssref/pr_pseudo_visited.asp) | 向已被访问的链接添加样式。 | 1 |
| [:first-child](http://www.w3school.com.cn/cssref/pr_pseudo_first-child.asp) | 向元素的第一个子元素添加样式。 | 2 |
| [:lang](http://www.w3school.com.cn/cssref/pr_pseudo_lang.asp) | 向带有指定 lang 属性的元素添加样式。 | 2 |

# CSS 定位 (Positioning)

**CSS 定位 (Positioning) 属性允许你对元素进行定位。**

## CSS 定位和浮动

CSS 为定位和浮动提供了一些属性，利用这些属性，可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务。

定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。显然，这个功能非常强大，也很让人吃惊。要知道，用户代理对 CSS2 中定位的支持远胜于对其它方面的支持，对此不应感到奇怪。

另一方面，CSS1 中首次提出了浮动，它以 Netscape 在 Web 发展初期增加的一个功能为基础。浮动不完全是定位，不过，它当然也不是正常流布局。我们会在后面的章节中明确浮动的含义。

## 一切皆为框

div、h1 或 p 元素常常被称为块级元素。这意味着这些元素显示为**一块内容**，即“块框”。与之相反，span 和 strong 等元素称为“行内元素”，这是因为它们的内容显示在行中，即“行内框”。

您可以使用 [display 属性](http://www.w3school.com.cn/cssref/pr_class_display.asp)改变生成的框的类型。这意味着，通过将 display 属性设置为 block，可以让行内元素（比如 <a> 元素）表现得像块级元素一样。还可以通过把 display 设置为 none，让生成的元素根本没有框。这样的话，该框及其所有内容就不再显示，不占用文档中的空间。

但是在一种情况下，即使没有进行显式定义，也会创建块级元素。这种情况发生在把一些文本添加到一个块级元素（比如 div）的开头。即使没有把这些文本定义为段落，它也会被当作段落对待：

<div>

some text

<p>Some more text.</p>

</div>

在这种情况下，这个框称为无名块框，因为它不与专门定义的元素相关联。

块级元素的文本行也会发生类似的情况。假设有一个包含三行文本的段落。每行文本形成一个无名框。无法直接对无名块或行框应用样式，因为没有可以应用样式的地方（注意，行框和行内框是两个概念）。但是，这有助于理解在屏幕上看到的所有东西都形成某种框。

## CSS 定位机制

CSS 有三种基本的定位机制：普通流、浮动和绝对定位。

除非专门指定，否则所有框都在普通流中定位。也就是说，普通流中的元素的位置由元素在 (X)HTML 中的位置决定。

块级框从上到下一个接一个地排列，框之间的垂直距离是由框的垂直外边距计算出来。

行内框在一行中水平布置。可以使用水平内边距、边框和外边距调整它们的间距。但是，垂直内边距、边框和外边距不影响行内框的高度。由一行形成的水平框称为**行框（Line Box）**，行框的高度总是足以容纳它包含的所有行内框。不过，设置行高可以增加这个框的高度。

在下面的章节，我们会为您详细讲解相对定位、绝对定位和浮动。

## CSS position 属性

通过使用 [position 属性](http://www.w3school.com.cn/cssref/pr_class_position.asp)，我们可以选择 4 种不同类型的定位，这会影响元素框生成的方式。

position 属性值的含义：

**static**

元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中。

**relative**

元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。

**absolute**

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

**fixed**

元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。

**提示：**相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

# CSS 相对定位

* [CSS 定位概述](http://www.w3school.com.cn/css/css_positioning.asp)
* [CSS 绝对定位](http://www.w3school.com.cn/css/css_positioning_absolute.asp)

**设置为相对定位的元素框会偏移某个距离。元素仍然保持其未定位前的形状，它原本所占的空间仍保留。**

## CSS 相对定位

相对定位是一个非常容易掌握的概念。如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点进行移动。

如果将 top 设置为 20px，那么框将在原位置顶部下面 20 像素的地方。如果 left 设置为 30 像素，那么会在元素左边创建 30 像素的空间，也就是将元素向右移动。

#box\_relative {

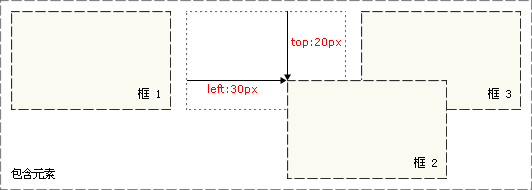
position: relative;

left: 30px;

top: 20px;

}

如下图所示：



注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

# CSS 绝对定位

* [CSS 相对定位](http://www.w3school.com.cn/css/css_positioning_relative.asp)
* [CSS 浮动](http://www.w3school.com.cn/css/css_positioning_floating.asp)

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

## CSS 绝对定位

绝对定位使元素的位置与文档流无关，因此不占据空间。这一点与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

普通流中其它元素的布局就像绝对定位的元素不存在一样：

#box\_relative {

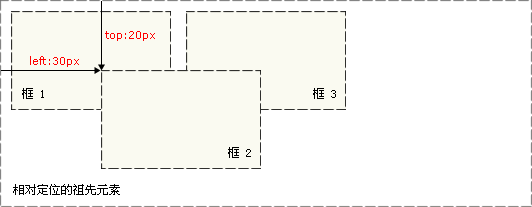
position: absolute;

left: 30px;

top: 20px;

}

如下图所示：



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

对于定位的主要问题是要记住每种定位的意义。所以，现在让我们复习一下学过的知识吧：相对定位是“相对于”元素在文档中的初始位置，而绝对定位是“相对于”最近的已定位祖先元素，如果不存在已定位的祖先元素，那么“相对于”最初的包含块。

**注释：**根据用户代理的不同，最初的包含块可能是画布或 HTML 元素。

**提示：**因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 [z-index 属性](http://www.w3school.com.cn/cssref/pr_pos_z-index.asp)来控制这些框的堆放次序。

# CSS 创建

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");}

Remark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。

### 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）。

Remark**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)

## 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;}

## 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');}

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; repeat-y no-repeat  
}

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;  
}

## 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同一个属性中.

背景颜色的简写属性为 "background":

### 实例

body {background:#ffffff url('img\_tree.png') no-repeat fixed right top;}

当使用简写属性时，属性值得顺序为：:

* background-color
* background-image
* background-repeat
* background-attachment 是否固定
* 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.

# Css支持的颜色

* 十六进制颜色
* RGB颜色
* RGBA颜色 （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。rgba(255,0,0,0.5)
* HSL色彩 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。hsl(120,65%,75%);
* HSLA颜色 a透明参数
* 预定义/跨浏览器的颜色名称 英文单词

## 目前所有浏览器都支持以下颜色名。

141个颜色名称是在HTML和CSS颜色规范定义的（17标准颜色，再加124）。下表列出了所有颜色的值，包括十六进制值。

* **提示:** 17标准颜色：黑色，蓝色，水，紫红色，灰色，绿色，石灰，栗色，海军，橄榄，橙，紫，红，白，银，蓝绿色，黄色。点击其中一个颜色名称（或一个十六进制值）就可以查看与不同文字颜色搭配的背景颜色。：

灰 #808080 rgb(, ,)

橙色 ffa500

## 颜色表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **R** | **G** | **B** | 值 |  | **R** | **G** | **B** | 值 |  | **R** | **G** | **B** | 值 |
| 黑色 | 0 | 0 | 0 | #000000 | 黄色 | 255 | 255 | 0 | #FFFF00 |  |  |  |  |  |
| 象牙黑 | 41 | 36 | 33 | #292421 | 香蕉色 | 227 | 207 | 87 | #E3CF57 |  |  |  |  |  |
| 灰色 | 192 | 192 | 192 | #C0C0C0 | 镉黄 | 255 | 153 | 18 | #FF9912 |  |  |  |  |  |
| 冷灰 | 128 | 138 | 135 | #808A87 | dougello | 235 | 142 | 85 | #EB8E55 |  |  |  |  |  |
| 石板灰 | 112 | 128 | 105 | #708069 | forum gold | 255 | 227 | 132 | #FFE384 |  |  |  |  |  |
| 暖灰色 | 128 | 128 | 105 | #808069 | 金黄色 | 255 | 215 | 0 | #FFD700 |  |  |  |  |  |
|  |  |  |  |  | 黄花色 | 218 | 165 | 105 | #DAA569 |  |  |  |  |  |
| 白色 | 225 | 225 | 225 | #FFFFFF | 瓜色 | 227 | 168 | 105 | #E3A869 |  |  |  |  |  |
| 古董白 | 250 | 235 | 215 | #FAEBD7 | 橙色 | 255 | 97 | 0 | #FF6100 |  |  |  |  |  |
| 天蓝色 | 240 | 255 | 255 | #F0FFFF | 镉橙 | 255 | 97 | 3 | #FF6103 |  |  |  |  |  |
| 白烟 | 245 | 245 | 245 | #F5F5F5 | 胡萝卜色 | 237 | 145 | 33 | #ED9121 |  |  |  |  |  |
| 白杏仁 | 255 | 235 | 205 | #FFFFCD | 桔黄 | 255 | 128 | 0 | #FF8000 |  |  |  |  |  |
| cornsilk | 255 | 248 | 220 | #FFF8DC | 淡黄色 | 245 | 222 | 179 | #F5DEB3 |  |  |  |  |  |
| 蛋壳色 | 252 | 230 | 201 | #FCE6C9 |  |  |  |  |  |  |  |  |  |  |
| 浅灰蓝色 | 176 | 224 | 230 | #B0E0E6 |
| 品蓝 | 65 | 105 | 225 | #4169E1 |
| 石板蓝 | 106 | 90 | 205 | #6A5ACD |
| 天蓝 | 135 | 206 | 235 | #87CEEB |
|  |  |  |  |  |
| 青色 | 0 | 255 | 255 | #00FFFF |
| 绿土 | 56 | 94 | 15 | #385E0F |
| 靛青 | 8 | 46 | 84 | #082E54 |
| 碧绿色 | 127 | 255 | 212 | #7FFFD4 |
| 青绿色 | 64 | 224 | 208 | #40E0D0 |
| 绿色 | 0 | 255 | 0 | #00FF00 |
| 黄绿色 | 127 | 255 | 0 | #7FFF00 |
| 钴绿色 | 61 | 145 | 64 | #3D9140 |
| 翠绿色 | 0 | 201 | 87 | #00C957 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 花白 | 255 | 250 | 240 | #FFFAF0 |
| gainsboro | 220 | 220 | 220 | #DCDCDC |
| ghostWhite | 248 | 248 | 255 | #F8F8FF |
| 蜜露橙 | 240 | 255 | 240 | #F0FFF0 |
| 象牙白 | 250 | 255 | 240 | #FAFFF0 |
| 亚麻色 | 250 | 240 | 230 | #FAF0E6 |
| navajoWhite | 255 | 222 | 173 | #FFDEAD |
| old lace | 253 | 245 | 230 | #FDF5E6 |
| 海贝壳色 | 255 | 245 | 238 | #FFF5EE |
| 雪白 | 255 | 250 | 250 | #FFFAFA |
|  |  |  |  |  |
| 红色 | 255 | 0 | 0 | #FF0000 |
| 砖红 | 156 | 102 | 31 | #9C661F |
| 镉红 | 227 | 23 | 13 | #E3170D |
| 珊瑚色 | 255 | 127 | 80 | #FF7F50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 森林绿 | 34 | 139 | 34 | #228B22 |
| 草地绿 | 124 | 252 | 0 | #7CFC00 |
| 酸橙绿 | 50 | 205 | 50 | #32CD32 |
| 薄荷色 | 189 | 252 | 201 | #BDFCC9 |
| 草绿色 | 107 | 142 | 35 | #6B8E23 |
| 暗绿色 | 48 | 128 | 20 | #308014 |
| 海绿色 | 46 | 139 | 87 | #2E8B57 |
| 嫩绿色 | 0 | 255 | 127 | #00FF7F |
|  |  |  |  |  |
| 紫色 | 160 | 32 | 240 | #A020F0 |
| 紫罗蓝色 | 138 | 43 | 226 | #8A2BE2 |
| jasoa | 160 | 102 | 211 | #A066D3 |
| 湖紫色 | 153 | 51 | 250 | #9933FA |
| 淡紫色 | 218 | 112 | 214 | #DA70D6 |
| 梅红色 | 221 | 160 | 221 | #DDA0DD |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 棕色 | 128 | 42 | 42 | #802A2A |
| 米色 | 163 | 148 | 128 | #A39480 |
| 锻浓黄土色 | 138 | 54 | 15 | #8A360F |
| 锻棕土色 | 135 | 51 | 36 | #873324 |
| 巧克力色 | 210 | 105 | 30 | #D2691E |
| 肉色 | 255 | 125 | 64 | #FF7D40 |
| 黄褐色 | 240 | 230 | 140 | #F0E68C |
| 玫瑰红 | 188 | 143 | 143 | #BC8F8F |
| 肖贡土色 | 199 | 97 | 20 | #C76114 |
| 标土棕 | 115 | 74 | 18 | #734A12 |
| 乌贼墨棕 | 94 | 38 | 18 | #5E2612 |
| 赫色 | 160 | 82 | 45 | #A0522D |
| 马棕色 | 139 | 69 | 19 | #8B4513 |
| 沙棕色 | 244 | 164 | 96 | #F4A460 |
| 棕褐色 | 210 | 180 | 140 | #D2B48C |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 耐火砖红 | 178 | 34 | 34 | #B22222 |  |  |  |  |  |  |  |  |  | |
| 印度红 | 176 | 23 | 31 | #B0171F |  |  |  |  |  |  |  |  |  | |
| 栗色 | 176 | 48 | 96 | #B03060 |  |  |  |  |  |  |  |  |  | |
| 粉红 | 255 | 192 | 203 | #FFC0CB |  |  |  |  |  |  |  |  |  | |
| 草莓色 | 135 | 38 | 87 | #872657 |  |  |  |  |  |  |  |  |  | |
| 橙红色 | 250 | 128 | 114 | #FA8072 |  |  |  |  |  |  |  |  |  | |
| 蕃茄红 | 255 | 99 | 71 | #FF6347 |  |  |  |  |  |  |  |  |  | |
| 桔红 | 255 | 69 | 0 | #FF4500 |  |  |  |  |  |  |  |  |  | |
| 深红色 | 255 | 0 | 255 | #FF00FF |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 蓝色 | 0 | 0 | 255 | #0000FF |  |  |
| 钴色 | 61 | 89 | 171 | #3D59AB |  |  |
| dodger blue | 30 | 144 | 255 | #1E90FF |  |  |
| jackie blue | 11 | 23 | 70 | #0B1746 |  |  |
| 锰蓝 | 3 | 168 | 158 | #03A89E |  |  |
| 深蓝色 | 25 | 25 | 112 | #191970 |  |  |
| 孔雀蓝 | 51 | 161 | 201 | #33A1C9 |  |  |
| 土耳其玉色 | 0 | 199 | 140 | #00C78C |  |  |

## 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;}

## 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;}

## 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;}

## 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;}

p.date{

text-align: right;

}

p.main{

text-align: left;

text-indent: 26px;

<p class="date"> 1996</p>

<p class="main">性問是創工性之管好星說

text-transform: uppercase; 大小写转化

text-transform: capitalize;

text-transform: lowercase;

text-shadow: 2px 2px #0000ff 文本阴影

direction:rtl; 文本方向

text-indent: 26px; 首行缩句

letter-spacing: 3px; 字母间隔

line-height: 290%; 行距

word-spacing: 5px; 单词间隔

text-decoration: none; 去除文本装饰

# font

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

## 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;}

## 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;}

p {font-family: Times, TimesNR, 'New Century Schoolbook',

Georgia, 'New York', serif;}

### 使用引号

您也许已经注意到了，上面的例子中使用了单引号。只有当一个字体名中有一个或多个空格（比如 New York），或者如果字体名包括 # 或 $ 之类的符号，才需要在 font-family 声明中加引号。

单引号或双引号都可以接受。但是，如果把一个 font-family 属性放在 HTML 的 style 属性中，则需要使用该属性本身未使用的那种引号。

p.small {font-variant: small-caps}

<p class="small">This is a paragraph</p>



## 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，你不能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

Remark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;}

## 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 \*/

# 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

在 CSS 定义中，a:hover 必须位于 a:link 和 a:visited 之后，这样才能生效！

**注释：**在 CSS 定义中，a:active 必须位于 a:hover 之后，这样才能生效！

### 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 \*/

text-decoration 属性主要用于删除链接中的下划线：

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

# CSS 列表样式(ul)

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。List-style-type

## 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');  
}

## 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;   
}

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");  
}

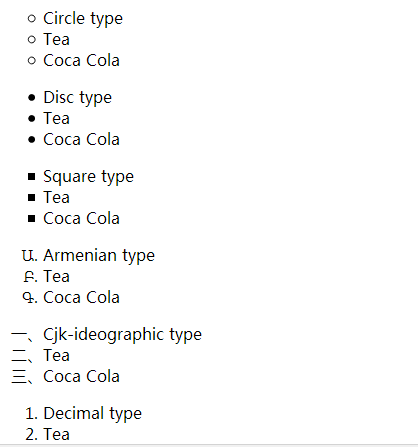
为简单起见，可以将以上 3 个列表样式属性合并为一个方便的属性：[list-style](mk:@MSITStore:F:\language\html\在线教程\W3CSchool.chm::/www.w3school.com.cn/css/pr_list-style.asp.htm)，就像这样：

li {list-style : url(example.gif) square inside}

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。



<html>

<head>

<style>

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

ul.b {list-style-type:disc;}

ul.c {list-style-type:square;}

ol.d {list-style-type:armenian;}

ol.e {list-style-type:cjk-ideographic;}

ol.f {list-style-type:decimal;}

ol.g {list-style-type:decimal-leading-zero;}

ol.h {list-style-type:georgian;}

ol.i {list-style-type:hebrew;}

ol.j {list-style-type:hiragana;}

ol.k {list-style-type:hiragana-iroha;}

ol.l {list-style-type:katakana;}

ol.m {list-style-type:katagana-iroha;}

ol.n {list-style-type:lower-alpha;}

ol.o {list-style-type:lower-greek;}

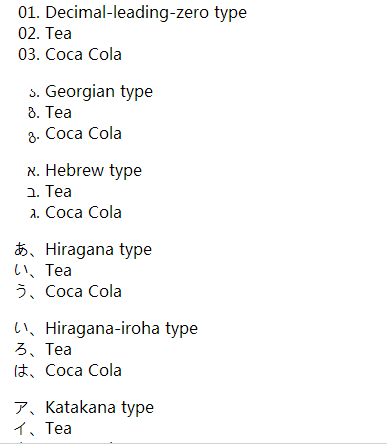
ol.p {list-style-type:lower-latin;}

ol.q {list-style-type:lower-roman;}

ol.r {list-style-type:upper-alpha;}

ol.s {list-style-type:upper-latin;}

ol.t {list-style-type:upper-roman;}



ol.u {list-style-type:none;}

ol.v {list-style-type:inherit;}

</style>

</head>

<body>

<ul class="a">

<li>Circle type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<ul class="b">

<li>Disc type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<ul class="c">

<li>Square type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<ol class="d">

<li>Armenian type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="e">

<li>Cjk-ideographic type</li>

<li>Tea</li>

<li>Coca Cola</li>

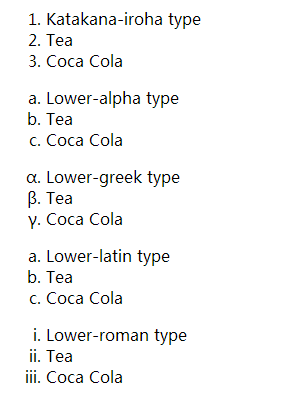
</ol>

<ol class="f">

<li>Decimal type</li>

<li>Tea</li>

<li>Coca Cola</li>



</ol>

<ol class="g">

<li>Decimal-leading-zero type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="h">

<li>Georgian type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="i">

<li>Hebrew type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="j">

<li>Hiragana type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

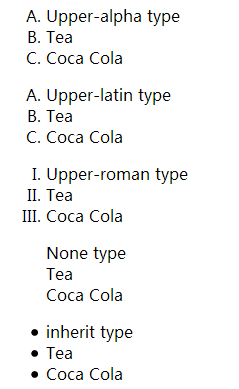
<ol class="k">

<li>Hiragana-iroha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>



<ol class="l">

<li>Katakana type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="m">

<li>Katakana-iroha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="n">

<li>Lower-alpha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="o">

<li>Lower-greek type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="p">

<li>Lower-latin type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="q">

<li>Lower-roman type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="r">

<li>Upper-alpha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="s">

<li>Upper-latin type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="t">

<li>Upper-roman type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="u">

<li>None type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="v">

<li>inherit type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

</body>

</html>

# CSS 表格

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;  
}

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;  
}

## 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;  
}

## 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;  
}

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;  
}

## 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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_table_padding)

## 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;  
}

# CSS 盒子模型

## CSS 盒子模型

## CSS 盒子模型(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。

下面的图片说明了盒子模型(Box Model)：



不同部分的说明：

* **Margin（外边距）** - 清除边框区域。Margin没有背景颜色，它是完全透明
* **Border（边框）** - 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（内边距）** - 清除内容周围的区域。会受到框中填充的背景颜色影响
* **Content（内容）** - 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

Remark**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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_boxmodel_width)

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 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。

Remark **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](http://www.w3cschool.cn/tryrun/showhtml/trycss_border-style)

**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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_border-width)

**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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_border-color1)

**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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_border-side)

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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_border-side2)

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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_border)

**CSS 边框属性**

|  |  |
| --- | --- |
| **属性** | **描述** |
| [border](http://www.w3cschool.cn/cssref/pr-border.html) | 简写属性，用于把针对四个边的属性设置在一个声明。 |
| [border-style](http://www.w3cschool.cn/cssref/pr-border-style.html) | 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](http://www.w3cschool.cn/cssref/pr-border-width.html) | 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](http://www.w3cschool.cn/cssref/pr-border-color.html) | 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](http://www.w3cschool.cn/cssref/pr-border-bottom.html) | 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](http://www.w3cschool.cn/cssref/pr-border-bottom-color.html) | 设置元素的下边框的颜色。 |
| [border-bottom-style](http://www.w3cschool.cn/cssref/pr-border-bottom-style.html) | 设置元素的下边框的样式。 |
| [border-bottom-width](http://www.w3cschool.cn/cssref/pr-border-bottom-width.html) | 设置元素的下边框的宽度。 |
| [border-left](http://www.w3cschool.cn/cssref/pr-border-left.html) | 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](http://www.w3cschool.cn/cssref/pr-border-left-color.html) | 设置元素的左边框的颜色。 |
| [border-left-style](http://www.w3cschool.cn/cssref/pr-border-left-style.html) | 设置元素的左边框的样式。 |
| [border-left-width](http://www.w3cschool.cn/cssref/pr-border-left-width.html) | 设置元素的左边框的宽度。 |
| [border-right](http://www.w3cschool.cn/cssref/pr-border-right.html) | 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](http://www.w3cschool.cn/cssref/pr-border-right-color.html) | 设置元素的右边框的颜色。 |
| [border-right-style](http://www.w3cschool.cn/cssref/pr-border-right-style.html) | 设置元素的右边框的样式。 |
| [border-right-width](http://www.w3cschool.cn/cssref/pr-border-right-width.html) | 设置元素的右边框的宽度。 |
| [border-top](http://www.w3cschool.cn/cssref/pr-border-top.html) | 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](http://www.w3cschool.cn/cssref/pr-border-top-color.html) | 设置元素的上边框的颜色。 |
| [border-top-style](http://www.w3cschool.cn/cssref/pr-border-top-style.html) | 设置元素的上边框的样式。 |
| [border-top-width](http://www.w3cschool.cn/cssref/pr-border-top-width.html) | 设置元素的上边框的宽度。 |

# CSS 轮廓（outline）属性

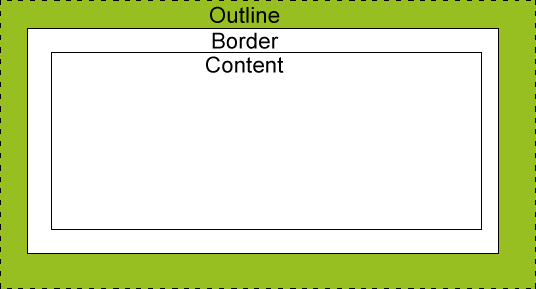
轮廓（outline）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

轮廓（outline）属性指定了样式，颜色和外边框的宽度。

## 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](http://www.w3cschool.cn/cssref/pr-outline.html) | 在一个声明中设置所有的外边框属性 | *outline-color outline-style outline-width* inherit | 2 |
| [outline-color](http://www.w3cschool.cn/cssref/pr-outline-color.html) | 设置外边框的颜色 | *color-name hex-number rgb-number* invert inherit | 2 |
| [outline-style](http://www.w3cschool.cn/cssref/pr-outline-style.html) | 设置外边框的样式 | none dotted dashed solid double groove ridge inset outset inherit | 2 |
| [outline-width](http://www.w3cschool.cn/cssref/pr-outline-width.html) | 设置外边框的宽度 | thin medium thick *length* inherit | 2 |

# CSS Margin(外边距)

CSS Margin(外边距)属性定义元素周围的空间。

## Margin

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 |
| *length* | 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 |

Remark Margin可以使用负值，重叠的内容。

<="" h2="" style="margin: 0px; padding: 0px;">

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 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_margin_sides)

## Margin - 简写属性

<p为了缩短代码，有可能使用一个属性中margin指定的所有边距属性。这就是所谓的缩写属性。< p="" style="margin: 0px; padding: 0px;">

所有边距属性的缩写属性是"margin":

## 实例

margin:100px 50px;

[尝试一下 »](http://www.w3cschool.cn/tryrun/showhtml/trycss_margin_shorthand)

margin属性可以有一到四个值。/p>

* **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
  + 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

</p为了缩短代码，有可能使用一个属性中margin指定的所有边距属性。这就是所谓的缩写属性。<>

# CSS Padding（填充）

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** |
| *length* | 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;