CSS 教程经典教程

注： 本教程由网络整理，版权归愿所有人

Ctrl+单击查看更多实例

CSS 经典 教程基础篇

通过使用 **CSS** 来提升工作效率！

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。

开始学习 **CSS** ！

CSS 实例

学习 **70** 个实例。您可以对 **CSS** 代码进行编辑，然后单击测试按钮来查看结果。

亲自试一下吧 ！

CSS 测验

在 **W3School** 测试您的 **CSS** 技能！

开始 **CSS** 测验！

CSS 参考手册

在 **W3School**，我们提供完整的 **CSS2** 参考手册（已升级为 **CSS2.1**）。

CSS2 参考手册

CSS 简介

需要具备的基础知识

在继续学习之前，你需要对下面的知识有基本的了解：

- **HTML**
- **XHTML**

CSS 概述

- **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

样式解决了一个普遍的问题

HTML 标签原本被设计为用于定义文档内容。通过使用 **<h1>**、**<p>**、**<table>** 这样的标签，**HTML** 的初衷是表达**"**这是标题**"**、**"**这是段落**"**、**"**这是表格**"**之类的信息。同时文档布局由浏览器来完成，而不使用任何的格式化标签。

由于两种主要的浏览器（**Netscape** 和 **Internet Explorer**）不断地将新的 **HTML** 标签和属性（比如字体标签和颜色属性）添加到 **HTML** 规范中，创建文档内容清晰地独立于文档表现层的站点变得越来越困难。

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**）。

所有的主流浏览器均支持层叠样式表。

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，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。

由于允许同时控制多重页面的样式和布局，**CSS** 可以称得上 **WEB** 设计领域的一个突破。作为网站开发者，你能够为每个 **HTML** 元素定义样式，并将之应用于你希望的任意多的页面中。如需进行全局的更新，只需简单地改变样式，然后网站中的所有元素均会自动地更新。

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：**<head>** 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）。

CSS 基础语法

CSS 语法

CSS 语法由三部分构成：选择器、属性和值：

```
selector {property: value}
```

选择器 **(selector)** 通常是你希望定义的 **HTML** 元素或标签，属性 **(property)** 是你希望改变的属性，并且每个属性都有一个值。属性和值被冒号分开，并由花括号包围，这样就组成了一个完整的样式声明（**declaration**）：

```
body {color: blue}
```

上面这行代码的作用是将 **body** 元素内的文字颜色定义为蓝色。在上述例子中，**body** 是选择器，而包括在花括号内的的部分是声明。声明依次由两部分构成：属性和值，**color** 为属性，**blue** 为值。

值的不同写法和单位

除了英文单词 **red**，我们还可以使用十六进制的颜色值 **#ff0000**：

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

为了节约字节，我们可以使用 **CSS** 的缩写形式：

```
p { color: #f00; }
```

我们还可以通过两种方法使用 **RGB** 值：

```
p { color: rgb(255,0,0); }
p { color: rgb(100%,0%,0%); }
```

请注意，当使用 **RGB** 百分比时，即使当值为 **0** 时也要写百分比符号。但是在其他的情况下就不需要这么做了。比如说，当尺寸为 **0** 像素时，**0** 之后不需要使用 **px** 单位，因为 **0** 就是 **0**，无论单位是什么。

记得写引号

提示：如果值为若干单词，则要给值加引号：

```
p {font-family: "sans serif";}
```

多重声明：

提示：如果要定义不止一个声明，则需要用分号将每个声明分开。下面的例子展示出如何定义一个红色文字的居中段落。最后一条规则是不需要加分号的，因为分号在英语中是一个分隔符号，不是结束符号。然

而，大多数有经验的设计师会在每条声明的末尾都加上分号，这么的好处是，当你从现有的规则中增减声明时，会尽可能的减少出错的可能性。就像这样：

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

你应该在每行只描述一个属性，这样可以增强样式定义的可读性，就像这样：

```
p {
  text-align: center;
  color: black;
  font-family: arial;
}
```

空格和大小写

大多数样式表包含不止一条规则，而大多数规则包含不止一个声明。多重声明和空格的使用使得样式表更容易被编辑：

```
body {
  color: #000;
  background: #fff;
  margin: 0;
  padding: 0;
  font-family: Georgia, Palatino, serif;
  }
```

是否包含空格不会影响 **CSS** 在浏览器的工作效果，同样，与 **XHTML** 不同，**CSS** 对大小写不敏感。不过存在一个例外：如果涉及到与 **HTML** 文档一起工作的话，**class** 和 **id** 名称对大小写是敏感的。

CSS 高级语法

选择器的分组

你可以对选择器进行分组，这样，被分组的选择器就可以分享相同的声明。用逗号将需要分组的选择器分开。在下面的例子中，我们对所有的标题元素进行了分组。所有的标题元素都是绿色的。

```
h1,h2,h3,h4,h5,h6 {
  color: green;
  }
```

继承及其问题

根据 **CSS**，子元素从父元素继承属性。但是它并不总是按此方式工作。看看下面这条规则：

```
body {
     font-family: Verdana, sans-serif;
     }
```

根据上面这条规则，站点的 **body** 元素将使用 **Verdana** 字体（假如访问者的系统中存在该字体的话）。

通过 **CSS** 继承，子元素将继承最高级元素（在本例中是 **body**）所拥有的属性（这些子元素诸如 **p, td, ul, ol, ul, li, dl, dt,**和 **dd**）。不需要另外的规则，所有 **body** 的子元素都应该显示 **Verdana** 字体，子元素的子元素也一样。并且在大部分的现代浏览器中，也确实是这样的。

但是在那个浏览器大战的血腥年代里，这种情况就未必会发生，那时候对标准的支持并不是企业的优先选择。比方说，**Netscape 4** 就不支持继承，它不仅忽略继承，而且也忽略应用于 **body** 元素的规则。**IE/Windows** 直到 **IE6** 还存在相关的问题，在表格内的字体样式会被忽略。我们又该如何是好呢？

友善地对待 Netscape 4

幸运地是，你可以通过使用我们称为 **"Be Kind to Netscape 4"** 的冗余法则来处理旧式浏览器无法理解继承的问题。

```
body  {
     font-family: Verdana, sans-serif;
     }

p, td, ul, ol, li, dl, dt, dd  {
     font-family: Verdana, sans-serif;
     }
```

4.0 浏览器无法理解继承，不过他们可以理解组选择器。这么做虽然会浪费一些用户的带宽，但是如果需要对 **Netscape 4** 用户进行支持，就不得不这么做。

继承是一个诅咒吗？

如果你不希望 **"Verdana, sans-serif"** 字体被所有的子元素继承，又该怎么做呢？比方说，你希望段落的字体是 **Times**。没问题。创建一个针对 **p** 的特殊规则，这样它就会摆脱父元素的规则：

```
body  {
     font-family: Verdana, sans-serif;
     }

td, ul, ol, ul, li, dl, dt, dd  {
     font-family: Verdana, sans-serif;
     }

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

CSS 派生选择器

派生选择器

通过依据元素在其位置的上下文关系来定义样式，你可以使标记更加简洁。

在 **CSS1** 中，通过这种方式来应用规则的选择器被称为上下文选择器 **(contextual selectors)**，这是由于它们依赖于上下文关系来应用或者避免某项规则。在 **CSS2** 中，它们称为派生选择器，但是无论你如何称呼它们，它们的作用都是相同的。

派生选择器允许你根据文档的上下文关系来确定某个标签的样式。通过合理地使用派生选择器，我们可以使 **HTML** 代码变得更加整洁。

比方说，你希望列表中的 **strong** 元素变为斜体字，而不是通常的粗体字，可以这样定义一个派生选择器：

```
li strong {
    font-style: italic;
    font-weight: normal;
  }
```

请注意标记为 **** 的蓝色代码的上下文关系：

```
<p><strong>我是粗体字，不是斜体字，因为我不在列表当中，所以这个规则对我不起作用
</strong></p><ol><li><strong>我是斜体字。这是因为 strong 元素位于 li 元素内。
</strong></li><li>我是正常的字体。</li></ol>
```

在上面的例子中，只有 **li** 元素中的 **strong** 元素的样式为斜体字，无需为 **strong** 元素定义特别的 **class** 或 **id**，代码更加简洁。

再看看下面的 **CSS** 规则：

```
strong {
     color: red;
     }

h2 {
     color: red;
     }

h2 strong {
     color: blue;
     }
```

下面是它施加影响的 **HTML**：

```
<p>The strongly emphasized word in this paragraph
is<strong>red</strong>.</p><h2>This subhead is also red.</h2><h2>The
strongly emphasized word in this subhead is<strong>blue</strong>.</h2>
```

CSS id 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。

id 选择器以 "#" 来定义。

下面的两个 **id** 选择器，第一个可以定义元素的颜色为红色，第二个定义元素的颜色为绿色：

```
#red {color:red;}
#green {color:green;}
```

下面的 **HTML** 代码中，**id** 属性为 **red** 的 **p** 元素显示为红色，而 **id** 属性为 **green** 的 **p** 元素显示为绿色。

```
<p id="red">这个段落是红色。</p><p id="green">这个段落是绿色。</p>
```

注意：id 属性只能在每个 **HTML** 文档中出现一次。想知道原因吗，请参阅 **XHTML:** 网站重构。

id 选择器和派生选择器

在现代布局中，id 选择器常常用于建立派生选择器。

```
#sidebar p {
       font-style: italic;
       text-align: right;
       margin-top: 0.5em;
       }
```

上面的样式只会应用于出现在 **id** 是 **sidebar** 的元素内的段落。这个元素很可能是 **div** 或者是表格单元，尽管它也可能是一个表格或者其他块级元素。它甚至可以是一个内联元素，比如 **** 或者 ****，不过这样的用法是非法的，因为不可以在内联元素 **** 中嵌入 **<p>**（如果你忘记了原因，请参阅 **XHTML:** 网站重构）。

一个选择器，多种用法

即使被标注为 sidebar 的元素只能在文档中出现一次，这个 id 选择器作为派生选择器也可以被使用很多次：

```
#sidebar p {
       font-style: italic;
       text-align: right;
       margin-top: 0.5em;
       }

#sidebar h2 {
       font-size: 1em;
       font-weight: normal;
       font-style: italic;
       margin: 0;
```

```
      line-height: 1.5;
      text-align: right;
      }
```

在这里，与页面中的其他 **p** 元素明显不同的是，**sidebar** 内的 **p** 元素得到了特殊的处理，同时，与页面中其他所有 **h2** 元素明显不同的是，**sidebar** 中的 **h2** 元素也得到了不同的特殊处理。

单独的选择器

id 选择器即使不被用来创建派生选择器，它也可以独立发挥作用：

```
#sidebar {
      border: 1px dotted #000;
      padding: 10px;
      }
```

根据这条规则，**id** 为 **sidebar** 的元素将拥有一个像素宽的黑色点状边框，同时其周围会有 **10** 个像素宽的内边距（**padding**，内部空白）。老版本的 **Windows/IE** 浏览器可能会忽略这条规则，除非你特别地定义这个选择器所属的元素：

```
div#sidebar {
      border: 1px dotted #000;
      padding: 10px;
      }
```

CSS 类选择器

在 CSS 中，类选择器以一个点号显示：

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

在上面的例子中，所有拥有 **center** 类的 **HTML** 元素均为居中。

在下面的 **HTML** 代码中，**h1** 和 **p** 元素都有 **center** 类。这意味着两者都将遵守 **".center"** 选择器中的规则。

```
<h1 class="center">
This heading will be center-aligned
</h1><p class="center">
This paragraph will also be center-aligned.
</p>
```

注意：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。

和 id 一样，class 也可被用作派生选择器：

```
.fancy td {
      color: #f60;
```

```
      background: #666;
      }
```

在上面这个例子中，类名为 **fancy** 的更大的元素内部的表格单元都会以灰色背景显示橙色文字。（名为 **fancy** 的更大的元素可能是一个表格或者一个 **div**）

元素也可以基于它们的类而被选择：

```
td.fancy {
      color: #f60;
      background: #666;
      }
```

在上面的例子中，类名为 **fancy** 的表格单元将是带有灰色背景的橙色。

```
<td class="fancy">
```

你可以将类 **fancy** 分配给任何一个表格元素任意多的次数。那些以 **fancy** 标注的单元格都会是带有灰色背景的橙色。那些没有被分配名为 **fancy** 的类的单元格不会受这条规则的影响。还有一点值得注意，**class** 为 **fancy** 的段落也不会是带有灰色背景的橙色，当然，任何其他被标注为 **fancy** 的元素也不会受这条规则的影响。这都是由于我们书写这条规则的方式，这个效果被限制于被标注为 **fancy** 的表格单元（即使用 **td** 元素来选择 **fancy** 类）。

如何创建 CSS

如何插入样式表

当读到一个样式表时，浏览器会根据它来格式化 HTML 文档。插入样式表的方法有三种：

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("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 type="text/css">
  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。

CSS 背景

CSS 允许应用纯色作为背景，也允许使用背景图像创建相当复杂的效果。

CSS 在这方面的能力远远在 HTML 之上。

背景色

可以使用 **background-color** 属性为元素设置背景色。这个属性接受任何合法的颜色值。

这条规则把元素的背景设置为灰色：

```
p {background-color: gray;}
```

如果您希望背景色从元素中的文本向外少有延伸，只需增加一些内边距：

```
p {background-color: gray; padding: 20px;}
```

如需查看本例的效果，可以亲自试一试！

可以为所有元素设置背景色，这包括 **body** 一直到 **em** 和 **a** 等行内元素。

background-color 不能继承，其默认值是 **transparent**。**transparent** 有"透明"之意。也就是说，如果一个元素没有指定背景色，那么背景就是透明的，这样其祖先元素的背景才能可见。

背景图像

要把图像放入背景，需要使用 **background-image** 属性。**background-image** 属性的默认值是 **none**，表示背景上没有放置任何图像。

如果需要设置一个背景图像，必须为这个属性设置一个 **URL** 值：

```
body {background-image: url(/i/eg_bg_04.gif);}
```

大多数背景都应用到 **body** 元素，不过并不仅限于此。

下面例子为一个段落应用了一个背景，而不会对文档的其他部分应用背景：

```
p.flower {background-image: url(/i/eg_bg_03.gif);}
```

您甚至可以为行内元素设置背景图像，下面的例子为一个链接设置了背景图像：

```
a.radio {background-image: url(/i/eg_bg_07.gif);}
```

如需查看上述例子的效果，可以亲自试一试！

理论上讲，甚至可以向 **textareas** 和 **select** 等替换元素的背景应用图像，不过并不是所有用户代理都能很好地处理这种情况。

另外还要补充一点，**background-image** 也不能继承。事实上，所有背景属性都不能继承。

背景重复

如果需要在页面上对背景图像进行平铺，可以使用 **background-repeat** 属性。

属性值 **repeat** 导致图像在水平垂直方向上都平铺，就像以往背景图像的通常做法一样。**repeat-x** 和 **repeat-y** 分别导致图像只在水平或垂直方向上重复，**no-repeat** 则不允许图像在任何方向上平铺。

默认地，背景图像将从一个元素的左上角开始。请看下面的例子：

```
body
  {
  background-image: url(/i/eg_bg_03.gif);
  background-repeat: repeat-y;
  }
```

如需查看上例的效果，可以亲自试一试。

背景定位

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

下面的例子在 **body** 元素中将一个背景图像居中放置：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:center;
  }
```

为 **background-position** 属性提供值有很多方法。首先，可以使用一些关键字：**top**、**bottom**、**left**、**right** 和 **center**。通常，这些关键字会成对出现，不过也不总是这样。还可以使用长度值，如 **100px** 或 **5cm**，最后也可以使用百分数值。不同类型的值对于背景图像的放置稍有差异。

关键字

图像放置关键字最容易理解，其作用如其名称所表明的。例如，**top right** 使图像放置在元素内边距区的右上角。

根据规范，位置关键字可以按任何顺序出现，只要保证不超过两个关键字 - 一个对应水平方向，另一个对象垂直方向。

如果只出现一个关键字，则认为另一个关键字是 **center**。

所以，如果希望每个段落的中部上方出现一个图像，只需声明如下：

```
p
  {
    background-image:url('bgimg.gif');
    background-repeat:no-repeat;
    background-position:top;
```

```
  }
```

下面是等价的位置关键字：

单一关键字	等价的关键字
center	center center
top	top center 或 center top
bottom	bottom center 或 center bottom
right	right center 或 center right
left	left center 或 center left

百分数值

百分数值的表现方式更为复杂。假设你希望用百分数值将图像在其元素中居中，这很容易：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:50% 50%;
  }
```

这会导致图像适当放置，其中心与其元素的中心对齐。**换句话说，百分数值同时应用于元素和图像。**也就是说，图像中描述为 **50% 50%** 的点（中心点）与元素中描述为 **50% 50%** 的点（中心点）对齐。

如果图像位于 **0% 0%**，其左上角将放在元素内边距区的左上角。如果图像位置是 **100% 100%**，会使图像的右下角放在右边距的右下角。

因此，如果你想把一个图像放在水平方向 **2/3**、垂直方向 **1/3** 处，可以这样声明：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:66% 33%;
  }
```

如果只提供一个百分数值，所提供的这个值将用作水平值，垂直值将假设为 **50%**。这一点与关键字类似。

background-position 的默认值是 **0% 0%**，在功能上相当于 **top left**。这就解释了背景图像为什么总是从元素内边距区的左上角开始平铺，除非您设置了不同的位置值。

长度值

长度值解释的是元素内边距区左上角的偏移。偏移点是图像的左上角。

比如，如果设置值为 **50px 100px**，图像的左上角将在元素内边距区左上角向右 **50** 像素、向下 **100** 像素的位置上：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:50px 100px;
  }
```

注意，这一点与百分数值不同，因为偏移只是从一个左上角到另一个左上角。也就是说，图像的左上角与 **background-position** 声明中的指定的点对齐。

背景关联

如果文档比较长，那么当文档向下滚动时，背景图像也会随之滚动。当文档滚动到超过图像的位置时，图像就会消失。

您可以通过 **background-attachment** 属性防止这种滚动。通过这个属性，可以声明图像相对于可视区是固定的（**fixed**），因此不会受到滚动的影响：

```
body
  {
  background-image:url(/i/eg_bg_02.gif);
  background-repeat:no-repeat;
  background-attachment:fixed
  }
```

如需查看上例的效果，可以亲自试一试。

background-attachment 属性的默认值是 **scroll**，也就是说，在默认的情况下，背景会随文档滚动。

CSS 背景实例

设置背景颜色

本例演示如何为元素设置背景颜色。

设置文本的背景颜色

本例颜色如何设置部分文本的背景颜色。

将图像设置为背景

本例演示如何将图像设置为背景。

将图像设置为背景 2

本例演示如何为多个元素同时设置背景图像。

如何重复背景图像

本例演示如何重复背景图像。

如何在垂直方向重复背景图像

本例演示如何垂直地重复背景图像。

如何在水平方向重复背景图像

本例演示如何水平地重复背景图像。

如何仅显示一次背景图像

本例演示如何仅显示一次背景图像。

如何放置背景图像

本例演示如何在页面上放置背景图像。

如何使用%来定位背景图像

本例演示如何使用百分比来在页面上定位背景图像。

如何使用像素来定位背景图像

本例演示如何使用像素来在页面上定位背景图像。

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。

所有背景属性在一个声明之中

本例演示如何使用简写属性来将所有背景属性设置在一个声明之中。

CSS 背景属性

属性	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 文本

CSS 文本属性可定义文本的外观。

通过文本属性，您可以改变文本的颜色、字符间距，对齐文本，装饰文本，对文本进行缩进，等等。

缩进文本

把 **Web** 页面上的段落的第一行缩进，这是一种最常用的文本格式化效果。

CSS 提供了 **text-indent** 属性，该属性可以方便地实现文本缩进。

通过使用 **text-indent** 属性，所有元素的第一行都可以缩进一个给定的长度，甚至该长度可以是负值。

这个属性最常见的用途是将段落的首行缩进，下面的规则会使所有段落的首行缩进 **5 em**：

```
p {text-indent: 5em;}
```

注意：一般来说，可以为所有块级元素应用 **text-indent**，但无法将该属性应用于行内元素，图像之类的替换元素上也无法应用 **text-indent** 属性。不过，如果一个块级元素（比如段落）的首行中有一个图像，它会随该行的其余文本移动。

提示：如果想把一个行内元素的第一行"缩进"，可以用左内边距或外边距创造这种效果。

使用负值

text-indent 还可以设置为负值。利用这种技术，可以实现很多有趣的效果，比如"悬挂缩进"，即第一行悬挂在元素中余下部分的左边：

```
p {text-indent: -5em;}
```

不过在为 **text-indent** 设置负值时要当心，如果对一个段落设置了负值，那么首行的某些文本可能会超出浏览器窗口的左边界。为了避免出现这种显示问题，建议针对负缩进再设置一个外边距或一些内边距：

```
p {text-indent: -5em; padding-left: 5em;}
```

使用百分比值

text-indent 可以使用所有长度单位，包括百分比值。

百分数要相对于缩进元素父元素的宽度。换句话说，如果将缩进值设置为 **20%**，所影响元素的第一行会缩进其父元素宽度的 **20%**。

在下例中，缩进值是父元素的 **20%**，即 **100** 个像素：

```
div {width: 500px;}
p {text-indent: 20%;}

<div><p>this is a paragragh</p></div>
```

继承

text-indent 属性可以继承，请考虑如下标记：

```
div#outer {width: 500px;}
div#inner {text-indent: 10%;}
p {width: 200px;}

<div id="outer"><div id="inner">some text. some text. some text.
<p>this is a paragragh.</p></div></div>
```

以上标记中的段落也会缩进 **50** 像素，这是因为这个段落继承了 **id** 为 **inner** 的 **div** 元素的缩进值。

水平对齐

text-align 是一个基本的属性，它会影响一个元素中的**文本行**互相之间的对齐方式。它的前 **3** 个值相当直接，不过第 **4** 个和第 **5** 个则略有些复杂。

值 **left**、**right** 和 **center** 会导致元素中的文本分别左对齐、右对齐和居中。

西方语言都是从左向右读，所有 **text-align** 的默认值是 **left**。文本在左边界对齐，右边界呈锯齿状（称为"从左到右"文本）。对于希伯来语和阿拉伯语之类的的语言，**text-align** 则默认为 **right**，因为这些语言从右向左读。不出所料，**center** 会使每个文本行在元素中居中。

提示：将块级元素或表元素居中，要通过在这些元素上适当地设置左、右外边距来实现。

text-align:center 与 <CENTER>

您可能会认为 **text-align:center** 与 **<CENTER>** 元素的作用一样，但实际上二者大不相同。

<CENTER> 不仅影响文本，还会把整个元素居中。**text-align** 不会控制元素的对齐，而只影响内部内容。元素本身不会从一段移到另一端，只是其中的文本受影响。

justify

最后一个水平对齐属性是 **justify**。

在两端对齐文本中，文本行的左右两端都放在父元素的内边界上。然后，调整单词和字母间的间隔，使各行的长度恰好相等。您也许已经注意到了，两端对齐文本在打印领域很常见。

需要注意的是，要由用户代理（而不是 **CSS**）来确定两端对齐文本如何拉伸，以填满父元素左右边界之间的空间。如需了解详情，请参阅 **CSS text-align** 属性参考页。

字间隔

word-spacing 属性可以改变字（单词）之间的标准间隔。其默认值 **normal** 与设置值为 **0** 是一样的。

word-spacing 属性接受一个正长度值或负长度值。如果提供一个正长度值，那么字之间的间隔就会增加。为 **word-spacing** 设置一个负值，会把它拉近：

```
p.spread {word-spacing: 30px;}
p.tight {word-spacing: -0.5em;}

<p class="spread">
This is a paragraph. The spaces between words will be decreased.
</p><p class="tight">
This is a paragraph. The spaces between words will be increased.
</p>
```

实例 **TIY**：增加或减少单词间距（字间隔）

注释：如需深入理解 **CSS** 对"字"（**word**）的定义，请访问 **CSS word-spacing** 属性参考页。

字母间隔

letter-spacing 属性与 **word-spacing** 的区别在于，字母间隔修改的是字符或字母之间的间隔。

与 **word-spacing** 属性一样，**letter-spacing** 属性的可取值包括所有长度。默认关键字是 **normal**（这与 **letter-spacing: 0** 相同）。输入的长度值会使字母之间的间隔增加或减少指定的量：

```
h1 {letter-spacing: -0.5em}
h4 {letter-spacing: 20px}

<h1>This is header 1</h1><h4>This is header 4</h4>
```

实例 **TIY**：规定字符间距（字母间隔）

字符转换

text-transform 属性处理文本的大小写。这个属性有 **4** 个值：

- **none**
- **uppercase**
- **lowercase**
- **capitalize**

默认值 **none** 对文本不做任何改动，将使用源文档中的原有大小写。顾名思义，**uppercase** 和 **lowercase** 将文本转换为全大写和全小写字符。最后，**capitalize** 只对每个单词的首字母大写。

作为一个属性，**text-transform** 可能无关紧要，不过如果您突然决定把所有 **h1** 元素变为大写，这个属性就很有用。不必单独地修改所有 **h1** 元素的内容，只需使用 **text-transform** 为你完成这个修改：

```
h1 {text-transform: uppercase}
```

使用 **text-transform** 有两方面的好处。首先，只需写一个简单的规则来完成这个修改，而无需修改 **h1** 元素本身。其次，如果您以后决定将所有大小写再切换为原来的大小写，可以更容易地完成修改。

实例 **TIY**：控制文本中字母的大小写

文本装饰

接下来，我们讨论 **text-decoration** 属性，这是一个很有意思的属性，它提供了很多非常有趣的行为。

text-decoration 有 **5** 个值：

- **none**
- **underline**
- **overline**
- **line-through**
- **blink**

不出所料，**underline** 会对元素加下划线，就像 **HTML** 中的 **U** 元素一样。**overline** 的作用恰好相反，会在文本的顶端画一个上划线。值 **line-through** 则在文本中间画一个贯穿线，等价于 **HTML** 中的 **S** 和 **strike** 元素。**blink** 会让文本闪烁，类似于 **Netscape** 支持的颇招非议的 **blink** 标记。

none 值会关闭原本应用到一个元素上的所有装饰。通常，无装饰的文本是默认外观，但也不总是这样。例如，链接默认地会有下划线。如果您希望去掉超链接的下划线，可以使用以下 **CSS** 来做到这一点：

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

注意：如果显式地用这样一个规则去掉链接的下划线，那么锚与正常文本之间在视觉上的唯一差别就是颜色（至少默认是这样的，不过也不能完全保证其颜色肯定有区别）。

还可以在一个规则中结合多种装饰。如果希望所有超链接既有下划线，又有上划线，则规则如下：

```
a:link a:visited {text-decoration: underline overline;}
```

不过要注意的是，如果两个不同的装饰都与同一元素匹配，胜出规则的值会完全取代另一个值。请考虑以下的规则：

```
h2.stricken {text-decoration: line-through;}
h2 {text-decoration: underline overline;}
```

对于给定的规则，所有 **class** 为 **stricken** 的 **h2** 元素都只有一个贯穿线装饰，而没有下划线和上划线，因为 **text-decoration 值会替换而不是累积起来**。

处理空白符

white-space 属性会影响到用户代理对源文档中的空格、换行和 **tab** 字符的处理。

通过使用该属性，可以影响浏览器处理字之间和文本行之间的空白符的方式。从某种程度上讲，默认的 **XHTML** 处理已经完成了空白符处理：它会把所有空白符合并为一个空格。所以给定以下标记，它在 **Web** 浏览器中显示时，各个字之间只会显示一个空格，同时忽略元素中的换行：

```
<p>This     paragraph has     many
    spaces           in it.</p>
```

可以用以下声明显式地设置这种默认行为：

```
p {white-space: normal;}
```

上面的规则告诉浏览器按照平常的做法去处理：丢掉多余的空白符。如果给定这个值，换行字符（回车）会转换为空格，一行中多个空格的序列也会转换为一个空格。

实例 **TIY：white-space: normal**

值 pre

不过，如果将 **white-space** 设置为 **pre**，受这个属性影响的元素中，空白符的处理就有所不同，其行为就像 **XHTML** 的 **pre** 元素一样；空白符不会被忽略。

如果 **white-space** 属性的值为 **pre**，浏览器将会注意额外的空格，甚至回车。在这个方面，而且仅在这个方面，任何元素都可以相当于一个 **pre** 元素。

实例 **TIY：white-space: pre**

注意：经测试，**IE 7** 以及更早版本的浏览器不支持该值，因此请使用非 **IE** 的浏览器来查看上面的实例。

值 nowrap

与之相对的值是 **nowrap**，它会防止元素中的文本换行，除非使用了一个 **br** 元素。在 **CSS** 中使用 **nowrap** 非常类似于 **HTML 4** 中用 **<td nowrap>** 将一个表单元格设置为不能换行，不过 **white-space** 值可以应用到任何元素。

实例 **TIY：white-space: nowrap**

值 pre-wrap 和 pre-line

CSS2.1 引入了值 **pre-wrap** 和 **pre-line**，这在以前版本的 **CSS** 中是没有的。这些值的作用是允许创作人员更好地控制空白符处理。

如果元素的 **white-space** 设置为 **pre-wrap**，那么该元素中的文本会保留空白符序列，但是文本行会正常地换行。如果设置为这个值，源文本中的行分隔符以及生成的行分隔符也会保留。**pre-line** 与 **pre-wrap** 相反，会像正常文本中一样合并空白符序列，但保留换行符。

实例 **TIY：white-space: pre-wrap**

实例 **TIY：white-space: pre-line**

注意：我们在 **IE7** 和 **FireFox2.0** 浏览器中测试了上面的两个实例，但是结果是，值 **pre-wrap** 和 **pre-line** 都没有得到很好的支持。

总结

下面的表格总结了 **white-space** 属性的行为：

值	空白符	换行符	自动换行
pre-line	合并	保留	允许
normal	合并	忽略	允许
nowrap	合并	忽略	不允许
pre	保留	保留	不允许
pre-wrap	保留	保留	允许

文本方向

如果您阅读的是英文书籍，就会从左到右、从上到下地阅读，这就是英文的流方向。不过，并不是所有语言都如此。我们知道古汉语就是从右到左来阅读的，当然还包括希伯来语和阿拉伯语等等。**CSS2** 引入了一个属性来描述其方向性。

direction 属性影响块级元素中文本的书写方向、表中列布局的方向、内容水平填充其元素框的方向、以及两端对齐元素中最后一行的为止。

注释：对于行内元素，只有当 **unicode-bidi** 属性设置为 **embed** 或 **bidi-override** 时才会应用 **direction** 属性。

direction 属性有两个值：**ltr** 和 **rtl**。大多数情况下，默认值是 **ltr**，显示从左到右的文本。如果显示从右到左的文本，应使用值 **rtl**。

CSS 文本实例：

设置文本颜色

本例演示如何设置文本的颜色。

设置文本的背景颜色

本例颜色如何设置部分文本的背景颜色。

规定字符间距

本例演示如何增加或减少字符间距。

使用百分比设置行间距

本例演示如何使用百分比值来设置段落中的行间距。

使用像素值设置行间距

本例演示如何使用像素值来设置段落中的行间距。

使用数值来设置行间距

本例演示如何使用一个数值来设置段落中的行间距。

对齐文本

本例演示如何对齐文本。

修饰文本

本例演示如何向文本添加修饰。

缩进文本

本例演示如何缩进文本首行。

控制文本中的字母

本例演示如何控制文本中的字母。

在元素中禁止文本折行

本例演示如何禁止在元素中的文本折行。

增加单词间距

本例演示如何增加段落中单词间的距离。

CSS 文本属性

属性	描述
color	设置文本颜色
direction	设置文本方向。
line-height	设置行高。
letter-spacing	设置字符间距。
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。CSS2 包含该属性，但是 CSS2.1 没有保留该属性。
text-transform	控制元素中的字母。
unicode-bidi	设置文本方向。
white-space	设置元素中空白的处理方式。
word-spacing	设置字间距。

CSS 字体

CSS 字体 (font) 属性定义文本中的字体。

设置字体属性是样式表的最常见用途之一。**CSS** 字体属性允许您设置字体系列 **(font-family)** 和字体加粗 **(font-weight)**，您还可以设置字体的大小、字体风格（如斜体）和字体变形（如小型大写字母）。

指定字体

可以使用 **font-family** 属性在文档中采用某种字体系列。

使用通用字体系列

如果你希望文档使用一种 **sans-serif** 字体，但是你并不关心是哪一种字体，以下就是一个合适的声明：

```
body {font-family: sans-serif;}
```

这样用户代理就会从 **sans-serif** 字体系列中选择一个字体（如 **Helvetica**），并将其应用到 **body** 元素。因为有继承，这种字体选择还将应用到 **body** 元素中包含的所有元素，除非有一种更特定的选择器将其覆盖。

指定字体系列

除了指定通用的字体系列，您还可以通过 **font-family** 属性设置更具体的字体。

下面的例子为所有 **h1** 元素设置了 **Verdana** 字体：

```
h1 {font-family: Georgia;}
```

这样的规则同时会产生另外一个问题，如果用户代理上没有安装 **Georgia** 字体，就只能使用用户代理的默认字体来显示 **h1** 元素。

我们可以通过结合特定字体名和通用字体系列来解决这个问题：

```
h1 {font-family: Georgia, serif;}
```

如果读者没有安装 **Georgia**，但安装了 **Times** 字体（**serif** 字体系列中的一种字体），用户代理就可能对 **h1** 元素使用 **Times**。尽管 **Times** 与 **Georgia** 并不完全匹配，但至少足够接近。

因此，我们建议在所有 **font-family** 规则中都提供一个通用字体系列。这样就提供了一条后路，在用户代理无法提供与规则匹配的特定字体时，就可以选择一个候选字体。

如果您对字体非常熟悉，也可以为给定的元素指定一系列类似的字体。要做到这一点，需要把这些字体按照优先顺序排列，然后用逗号进行连接：

```
p {font-family: Times, TimesNR, 'New Century Schoolbook',
     Georgia, 'New York', serif;}
```

根据这个列表，用户代理会按所列的顺序查找这些字体。如果列出的所有字体都不可用，就会简单地选择一种可用的 **serif** 字体。

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。

CSS 字体实例：

设置文本的字体

本例演示如何设置文本字体。

设置字体尺寸

本例演示如何设置字体尺寸。

设置字体风格

本例演示如何设置字体风格。

设置字体的异体

本例演示如何设置字体的异体。

设置字体的粗细

本例演示如何设置字体的粗细。

所有字体属性在一个声明之内

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。

CSS 字体属性

属性	描述
font	简写属性。作用是把所有针对字体的属性设置在一个声明中。
font-family	设置字体系列。
font-size	设置字体的尺寸。
font-size-adjust	当首选字体不可用时，对替换字体进行智能缩放。（CSS2.1 已删除该属性。）
font-stretch	对字体进行水平拉伸。（CSS2.1 已删除该属性。）
font-style	设置字体风格。
font-variant	以小型大写字体或者正常字体显示文本。
font-weight	设置字体的粗细。

CSS 列表

CSS 列表属性允许你放置、改变列表项标志，或者将图像作为列表项标志。

CSS 列表

从某种意义上讲，不是描述性的文本的任何内容都可以认为是列表。人口普查、太阳系、家谱、参观菜单，甚至你的所有朋友都可以表示为一个列表或者是列表的列表。

由于列表如此多样，这使得列表相当重要，所以说，**CSS** 中列表样式不太丰富确实是一大憾事。

列表类型

要影响列表的样式，最简单（同时支持最充分）的办法就是改变其标志类型。

例如，在一个无序列表中，列表项的标志 **(marker)** 是出现在各列表项旁边的圆点。在有序列表中，标志可能是字母、数字或另外某种计数体系中的一个符号。

要修改用于列表项的标志类型，可以使用属性 **list-style-type**：

```
ul {list-style-type : square}
```

上面的声明把无序列表中的列表项标志设置为方块。

列表项图像

有时，常规的标志是不够的。你可能想对各标志使用一个图像，这可以利用 **list-style-image** 属性做到：

```
ul li {list-style-image : url(xxx.gif)}
```

只需要简单地使用一个 **url()** 值，就可以使用图像作为标志。

列表标志位置

CSS2.1 可以确定标志出现在列表项内容之外还是内容内部。这是利用 **list-style-position** 完成的。

简写列表样式

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**，就像这样：

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

list-style 的值可以按任何顺序列出，而且这些值都可以忽略。只要提供了一个值，其它的就会填入其默认值。

CSS 列表实例：

在无序列表中的不同类型的列表标记

本例演示在 **CSS** 中不同类型的列表项标记。

在有序列表中不同类型的列表项标记

本例演示在 **CSS** 中不同类型的列表项标记。

所有的列表样式类型

本例演示在 **CSS** 中所有不同类型的列表项标记。

将图像作为列表项标记

本例演示如何将图像作为列表项标记。

放置列表标记

本例演示在何处放置列表标记。

在一个声明中定义所有的列表属性

本例演示将所有针对列表的属性设置于一个简写属性。

CSS 列表属性(list)

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。
marker-offset	

CSS 表格

CSS 表格属性允许你设置表格的布局。

实例：

设置表格的布局

本例演示如何设置表格的布局。

显示表格中的空单元

本例演示是否显示表格中的空单元。（请在非 **IE** 浏览器中浏览）

合并表格边框

本例演示是否把表格边框显示为一条单独的边框，还是像标准的 **HTML** 中那样分开显示。

设置表格边框之间的空白

本例演示如何设置单元格边框之间的距离。（请在非 **IE** 浏览器中浏览）

设置表格标题的位置

本例演示如何定位表格的标题。（请在非 **IE** 浏览器中浏览）

CSS Table 属性

CSS 表格属性允许你设置表格的布局。（请注意，本节介绍的不是如何使用表来建立布局，而是要介绍 **CSS** 中表本身如何布局。）

属性	描述

border-collapse	设置是否把表格边框合并为单一的边框。
border-spacing	设置分隔单元格边框的距离。（仅用于 "separated borders" 模型）
caption-side	设置表格标题的位置。
empty-cells	设置是否显示表格中的空单元格。（仅用于 "separated borders" 模型）
table-layout	设置显示单元、行和列的算法。

CSS 轮廓

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。

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。

设置轮廓的颜色

本例演示如何设置轮廓的颜色。

设置轮廓的样式

本例演示如何设置轮廓的样式。

设置轮廓的宽度

本例演示如何设置轮廓的宽度。

CSS 边框属性

"CSS" 列中的数字指示哪个 **CSS** 版本定义了该属性。

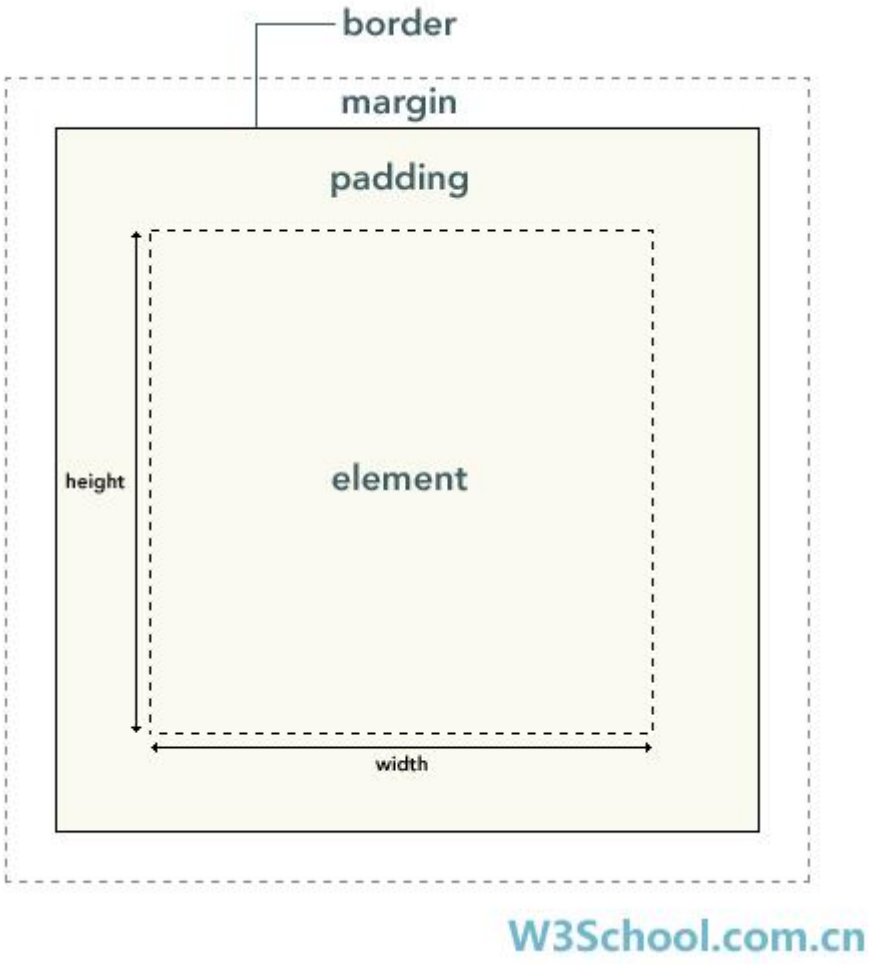
属性	描述	CSS
outline	在一个声明中设置所有的轮廓属性。	2
outline-color	设置轮廓的颜色。	2
outline-style	设置轮廓的样式。	2
outline-width	设置轮廓的宽度。	

CSS 经典 教程提高篇--CSS 框模型

CSS 框模型概述

CSS 框模型 (Box Model) 规定了元素框处理元素内容、内边距、边框 和 外边距 的方式。

CSS 框模型概述



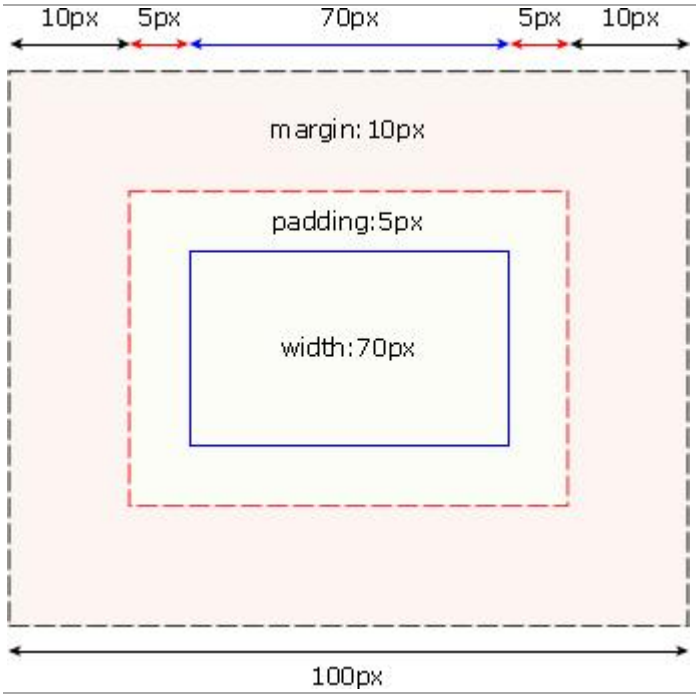
元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

内边距、边框和外边距都是可选的，默认值是零。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 **margin** 和 **padding** 设置为零来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器对所有元素进行设置：

```
* {
  margin: 0;
  padding: 0;
}
```

在 **CSS** 中，**width** 和 **height** 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的尺寸。

假设框的每个边上有 **10** 个像素的外边距和 **5** 个像素的内边距。如果希望这个元素框达到 **100** 个像素，就需要将内容的宽度设置为 **70** 像素，请看下图：




```
#box {
  width: 70px;
  margin: 10px;
  padding: 5px;
}
```

提示：内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边。

提示：外边距可以是负值，而且在很多情况下都要使用负值的外边距。

浏览器兼容性

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。

术语翻译

- **element：**元素。
- **padding：**内边距，也有资料将其翻译为填充。
- **border：**边框。
- **margin：**外边距，也有资料将其翻译为空白或空白边。

在 **w3school**，我们把 **padding** 和 **margin** 统一地称为内边距和外边距。边框内的空白是内边距，边框外的空白是外边距，很容易记吧：）

CSS 内边距

元素的内边距在边框和内容区之间。控制该区域最简单的属性是 padding 属性。

CSS padding 属性定义元素边框与元素内容之间的空白区域。

CSS padding 属性

CSS padding 属性定义元素的内边距。**padding** 属性接受长度值或百分比值，但不允许使用负值。

例如，如果您希望所有 **h1** 元素的各边都有 **10** 像素的内边距，只需要这样：

```
h1 {padding: 10px;}
```

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

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

单边内边距属性

也通过使用下面四个单独的属性，分别设置上、右、下、左内边距：

- **padding-top**
- **padding-right**
- **padding-bottom**
- **padding-left**

您也许已经想到了，下面的规则实现的效果与上面的简写规则是完全相同的：

```
h1 {
  padding-top: 10px;
  padding-right: 0.25em;
  padding-bottom: 2ex;
  padding-left: 20%;
  }
```

内边距的百分比数值

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

下面这条规则把段落的内边距设置为父元素 **width** 的 **10%**：

```
p {padding: 10%;}
```

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

```
<div style="width: 200px;"><p>This paragragh is contained within a DIV that has a width of 200 pixels.</p></div>
```

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

CSS 内边距实例：

所有内边距属性在一个声明中

本例演示使用简写属性将所有的内边距属性设置于一个声明中，可以有一到四个值。

设置下内边距 **1**

本例演示如何使用厘米值来设置单元格的下内边距。

设置下内边距 **2**

本例演示如何使用百分比值来设置单元格的下内边距。

设置左内边距 **1**

本例演示如何使用厘米值来设置单元格的左内边距。

设置左内边距 **2**

本例演示如何使用百分比值来设置单元格的左内边距。

设置右内边距 **1**

本例演示如何使用厘米值来设置单元格的右内边距。

设置右内边距 **2**

本例演示如何使用百分比值来设置单元格的右内边距。

设置上内边距 **1**

本例演示如何使用厘米值来设置单元格的上内边距。

设置上内边距 **2**

本例演示如何使用百分比值来设置单元格的上内边距。

CSS 内边距属性

属性	描述
padding	简写属性。作用是在一个声明中设置元素的所内边距属性。
padding-bottom	设置元素的下内边距。
padding-left	设置元素的左内边距。
padding-right	设置元素的右内边距。
padding-top	设置元素的上内边距。

CSS 边框

元素的边框 (border) 是围绕元素内容和内边距的一条或多条线。

CSS border 属性允许你规定元素边框的样式、宽度和颜色。

CSS 边框

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

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

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

边框与背景

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

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

边框的样式

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

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

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

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

定义多种样式

您可以为一个边框定义多个样式，例如：

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

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

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

定义单边样式

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

- **border-top-style**

- **border-right-style**
- **border-bottom-style**
- **border-left-style**

因此这两种方法是等价的：

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

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

边框的宽度

您可以通过 **border-width** 属性为边框指定宽度。

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

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

所以，我们可以这样设置边框的宽度：

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

或者：

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

定义单边宽度

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

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

上面的例子也可以简写为（这样写法称为**值复制**）：

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

您也可以通过下列属性分别设置边框各边的宽度：

- **border-top-width**
- **border-right-width**
- **border-bottom-width**
- **border-left-width**

因此，下面的规则与上面的例子是等价的：

```
p {
  border-style: solid;
  border-top-width: 15px;
  border-right-width: 5px;
  border-bottom-width: 15px;
  border-left-width: 5px;
  }
```

没有边框

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

那么如果把 **border-style** 设置为 **none** 会出现什么情况：

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

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

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

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

```
h1 {border-width: 20px;}
```

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

边框的颜色

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

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

```
p {
  border-style: solid;
  border-color: blue rgb(25%,35%,45%) #909090 red;
  }
```

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

```
p {
  border-style: solid;
  border-color: blue red;
  }
```

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

定义单边颜色

还有一些单边边框颜色属性。它们的原理与单边样式和宽度属性相同：

- **border-top-color**
- **border-right-color**
- **border-bottom-color**
- **border-left-color**

要为 **h1** 元素指定实线黑色边框，而右边框为实线红色，可以这样指定：

```
h1 {
  border-style: solid;
  border-color: black;
  border-right-color: red;
  }
```

透明边框

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

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

```
<a href="#">AAA</a><a href="#">BBB</a><a href="#">CCC</a>
```

我们为上面的链接定义了如下样式：

```
a:link, a:visited {
  border-style: solid;
  border-width: 5px;
  border-color: transparent;
  }
a:hover {border-color: gray;}
```

如需查看以上样式的效果，请点击：**TIY**。

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

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

CSS 边框实例：

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。

设置四边框样式

本例演示如何设置四边框样式。

设置每一边的不同边框

本例演示如何在元素的各边设置不同的边框。

所有边框宽度属性在一个声明之中

本例演示用简写属性来将所有边框宽度属性设置于同一声明中。

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。

所有下边框属性在一个声明中

本例演示用简写属性来将所有下边框属性设置在同一声明中。

设置下边框的颜色

本例演示如何设置下边框的颜色。

设置下边框的样式

本例演示如何设置下边框的样式。

设置下边框的宽度

本例演示如何设置下边框的宽度。

所有左边框属性在一个声明之中

所有左边框属性在一个声明之中

设置左边框的颜色

本例演示如何设置左边框的颜色。

设置左边框的样式

本例演示如何设置左边框的样式。

设置左边框的宽度

本例演示如何设置左边框的宽度。

所有右边框属性在一个声明之中

本例演示一个简写属性，用于把所有右边框属性设置在一条声明中。

设置右边框的颜色

本例演示如何设置右边框的颜色。

设置右边框的样式

本例演示如何设置右边框的样式。

设置右边框的宽度

本例演示如何设置右边框的宽度。

所有上边框属性在一个声明之中

本例演示用简写属性来将所有上边框属性设置于同一声明之中。

设置上边框的颜色

本例演示如何设置上边框的颜色。

设置上边框的样式

本例演示如何设置上边框的样式。

设置上边框的宽度

本例演示如何设置上边框的宽度。

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。
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 外边距

围绕在元素边框的空白区域是外边距。设置外边距会在元素外创建额外的"空白"。

设置外边距的最简单的方法就是使用 margin 属性，这个属性接受任何长度单位、百分数值甚至负值。

CSS margin 属性

设置外边距的最简单的方法就是使用 **margin** 属性。

margin 属性接受任何长度单位，可以是像素、英寸、毫米或 **em**。

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

```
h1 {margin : 0.25in;}
```

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

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

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

```
margin: top right bottom left
```

另外，还可以为 **margin** 设置一个百分比数值：

```
p {margin : 10%;}
```

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

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

值复制

还记得吗？我们曾经在前两节中提到过值复制。下面我们为您讲解如何使用值复制。

有时，我们会输入一些重复的值：

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

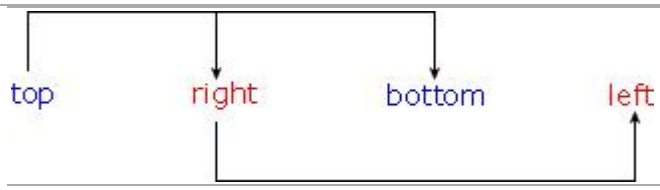
通过值复制，您可以不必重复地键入这对数字。上面的规则与下面的规则是等价的：

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

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

- 如果缺少左外边距的值，则使用右外边距的值。
- 如果缺少下外边距的值，则使用上外边距的值。
- 如果缺少右外边距的值，则使用上外边距的值。

下图提供了更直观的方法来了解这一点：



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

利用这个简单的机制，您只需指定必要的值，而不必全部都应用 **4** 个值，例如：

```
h1 {margin: 0.25em 1em 0.5em;}    /* 等价于 0.25em 1em 0.5em 1em */
h2 {margin: 0.5em 1em;}           /* 等价于 0.5em 1em 0.5em 1em */
p {margin: 1px;}                  /* 等价于 1px 1px 1px 1px */
```

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

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

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

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

```
p {margin: auto auto auto 20px;}
```

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

单边外边距属性

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

```
p {margin-left: 20px;}
```

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

- **margin-top**
- **margin-right**
- **margin-bottom**
- **margin-left**

一个规则中可以使用多个这种单边属性，例如：

```
h2 {
  margin-top: 20px;
  margin-right: 30px;
  margin-bottom: 30px;
  margin-left: 20px;
  }
```

当然，对于这种情况，使用 **margin** 可能更容易一些：

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

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

提示和注释

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

CSS 外边距实例：

设置文本的左外边距

本例演示如何设置文本的左外边距。

设置文本的右外边距

本例演示如何设置文本的右外边距。

设置文本的上外边距

本例演示如何设置文本的上外边距。

设置文本的下外边距

本例演示如何设置文本的下外边距。

所有的外边距属性在一个声明中。

本例演示如何将所有的外边距属性设置于一个声明中。

CSS 外边距属性

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 外边距合并

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。

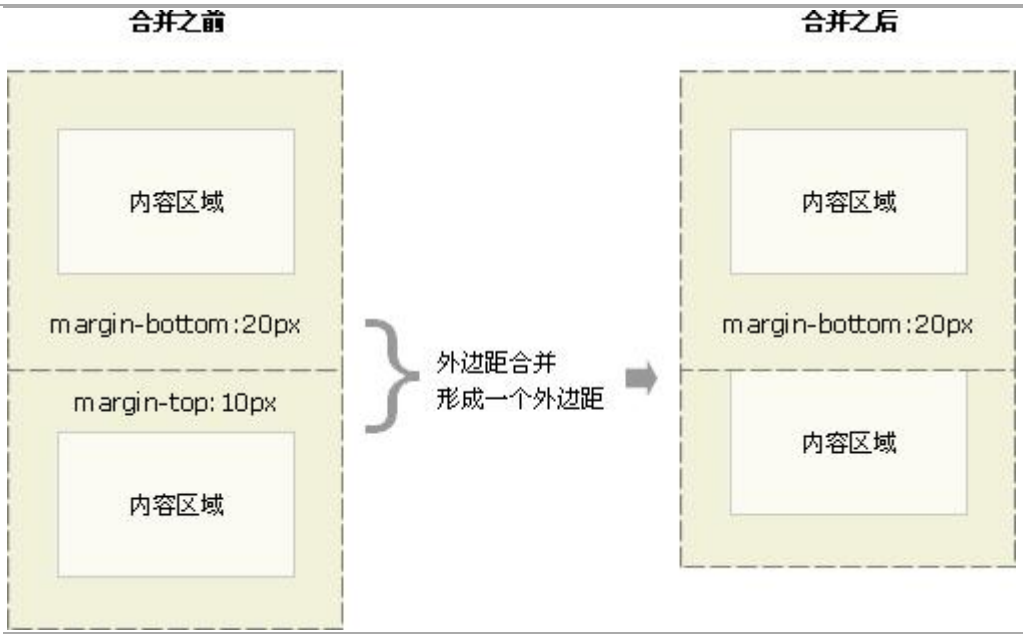
合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

外边距合并

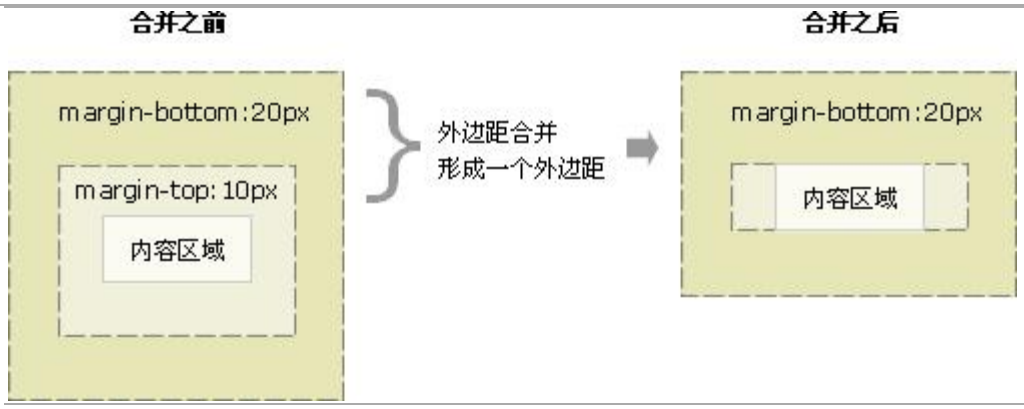
外边距合并（叠加）是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。

简单地说，外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。请看下图：



当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并。请看下图：



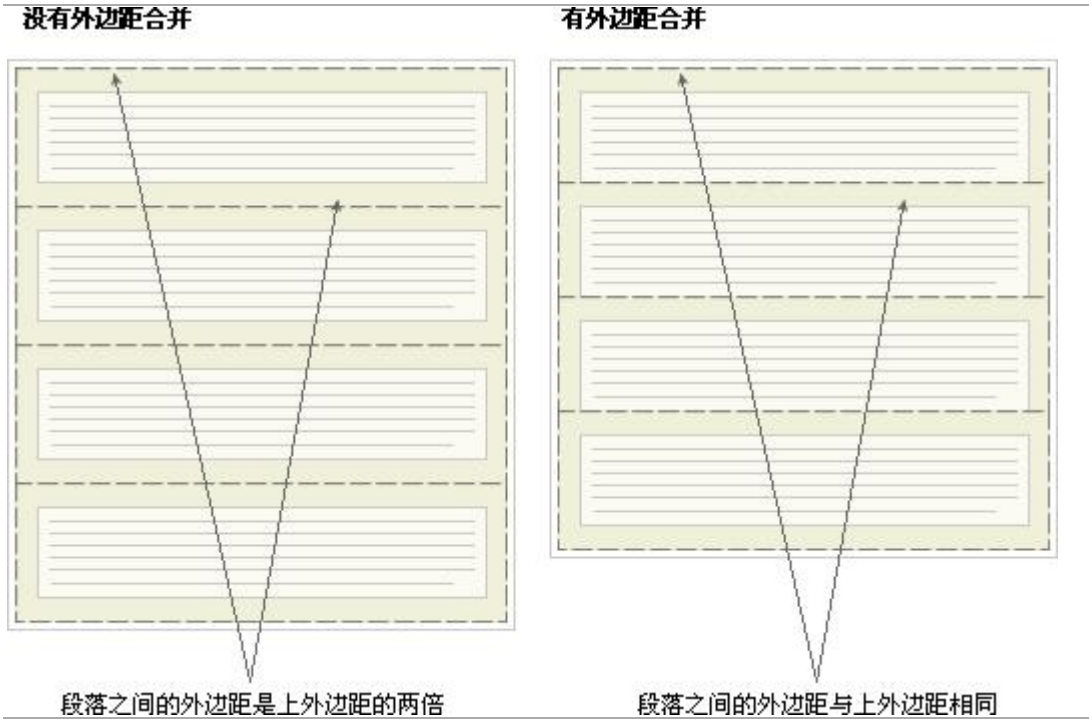
尽管看上去有些奇怪，但是外边距甚至可以与自身发生合并。

假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并：

如果这个外边距遇到另一个元素的外边距，它还会发生合并：

这就是一系列的段落元素占用空间非常小的原因，因为它们的所有外边距都合并到一起，形成了一个小的外边距。

外边距合并初看上去可能有点奇怪，但是实际上，它是有意义的。以由几个段落组成的典型文本页面为例。第一个段落上面的空间等于段落的上外边距。如果没有外边距合并，后续所有段落之间的外边距都将是相邻上外边距和下外边距的和。这意味着段落之间的空间是页面顶部的两倍。如果发生外边距合并，段落之间的上外边距和下外边距就合并在一起，这样各处的距离就一致了。



注释：只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

CSS 经典 教程提高篇--CSS 定位

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，我们可以选择 **4** 中不同类型的定位，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。

实例

定位：相对定位

本例演示如何相对于一个元素的正常位置来对其定位。

定位：绝对定位

本例演示如何使用绝对值来对元素进行定位。

定位：固定定位

本例演示如何相对于浏览器窗口来对元素进行定位。

使用固定值设置图像的上边缘

本例演示如何使用固定值设置图像的上边缘。

使用百分比设置图像的上边缘

本例演示如何使用百分比值设置图像的上边缘。

使用像素值设置图像的底部边缘

本例演示如何使用像素值设置图像的底部边缘。

使用百分比设置图像的底部边缘

本例演示如何使用百分比值设置图像的底部边缘。

使用固定值设置图像的左边缘

本例演示如何使用固定值设置图像的左边缘。

使用百分比设置图像的左边缘

本例演示如何使用百分比值设置图像的左边缘。

使用固定值设置图像的右边缘

本例演示如何使用固定值设置图像的右边缘。

使用百分比设置图像的右边缘

本例演示如何使用百分比值设置图像的右边缘。

如何使用滚动条来显示元素内溢出的内容

本例演示当元素内容太大而超出规定区域时，如何设置溢出属性来规定相应的动作。

如何隐藏溢出元素中溢出的内容

本例演示在元素中的内容太大以至于无法适应指定的区域时，如何设置 **overflow** 属性来隐藏其内容。

如何设置浏览器来自动地处理溢出

本例演示如何设置浏览器来自动地处理溢出。

设置元素的形状

本例演示如何设置元素的形状。此元素被剪裁到这个形状内，并显示出来。

垂直排列图象

本例演示如何在文本中垂直排列图象。

Z-index

Z-index 可被用于将在一个元素放置于另一元素之后。

Z-index

上面的例子中的元素已经更改了 **Z-index**。

CSS 定位属性

CSS 定位属性允许你对元素进行定位。

属性	描述
position	把元素放置到一个静态的、相对的、绝对的、或固定的位置中。
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。
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。
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。
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。
overflow	设置当元素的内容溢出其区域时发生的事情。
clip	设置元素的形状。元素被剪入这个形状之中，然后显示出来。
vertical-align	设置元素的垂直对齐方式。
z-index	设置元素的堆叠顺序。

CSS 相对定位

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。

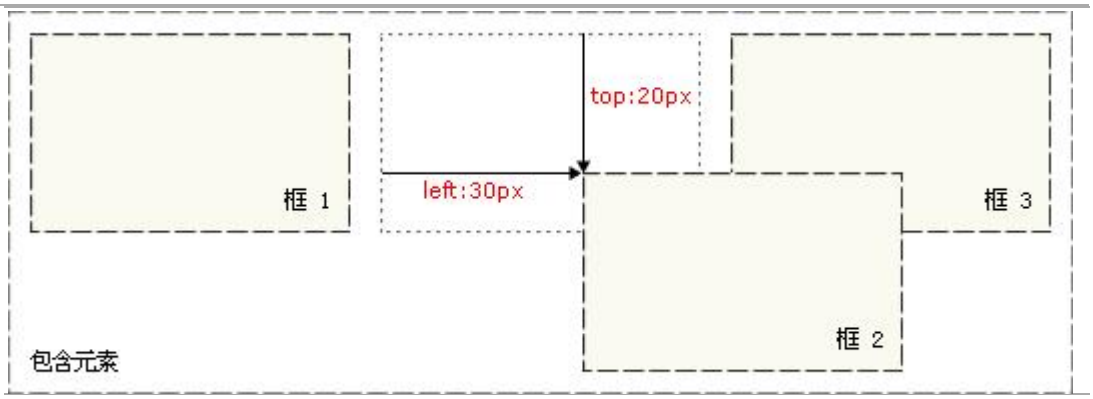
CSS 相对定位

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 相对定位实例

定位：相对定位

本例演示如何相对于一个元素的正常位置来对其定位。

CSS 绝对定位

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

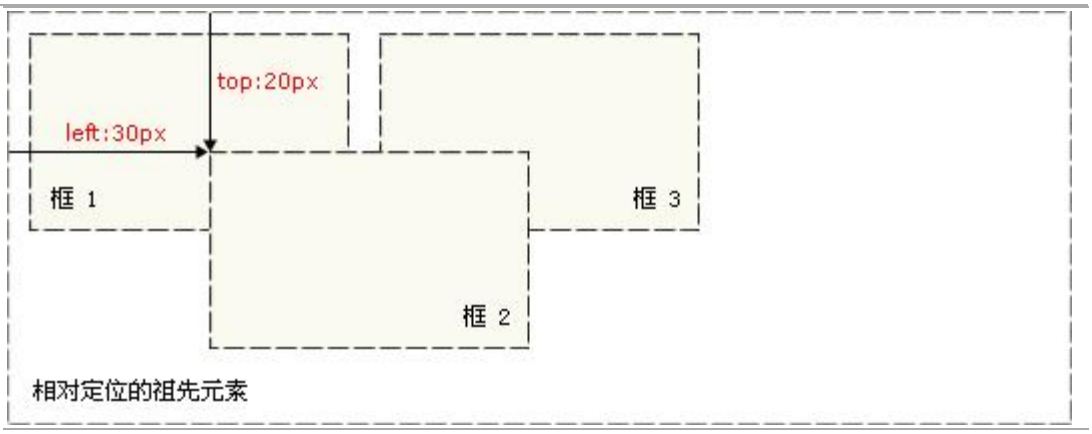
绝对定位使元素的位置与文档流无关，因此不占据空间。这一点与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

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** 属性来控制这些框的堆放次序。

CSS 绝对定位实例

定位：绝对定位

本例演示如何使用绝对值来对元素进行定位。

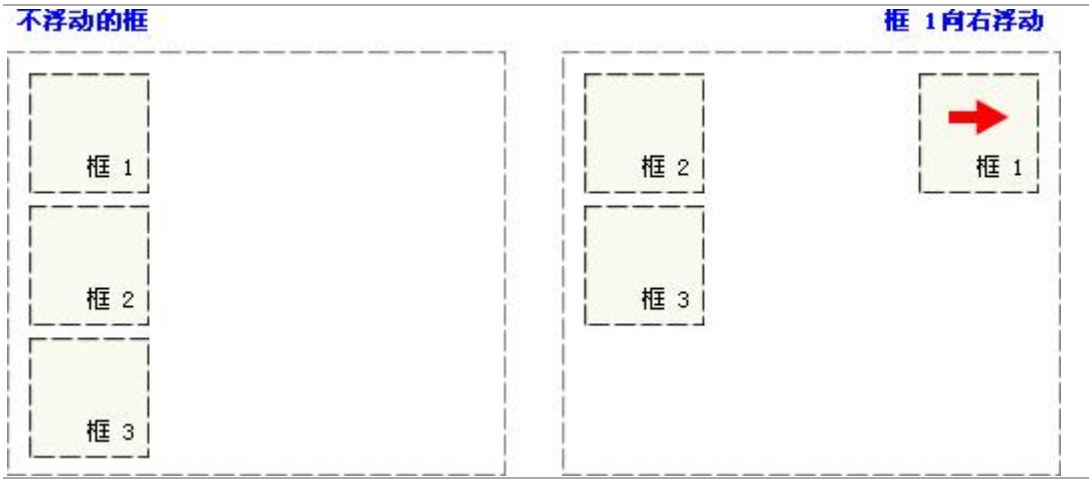
CSS 浮动

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

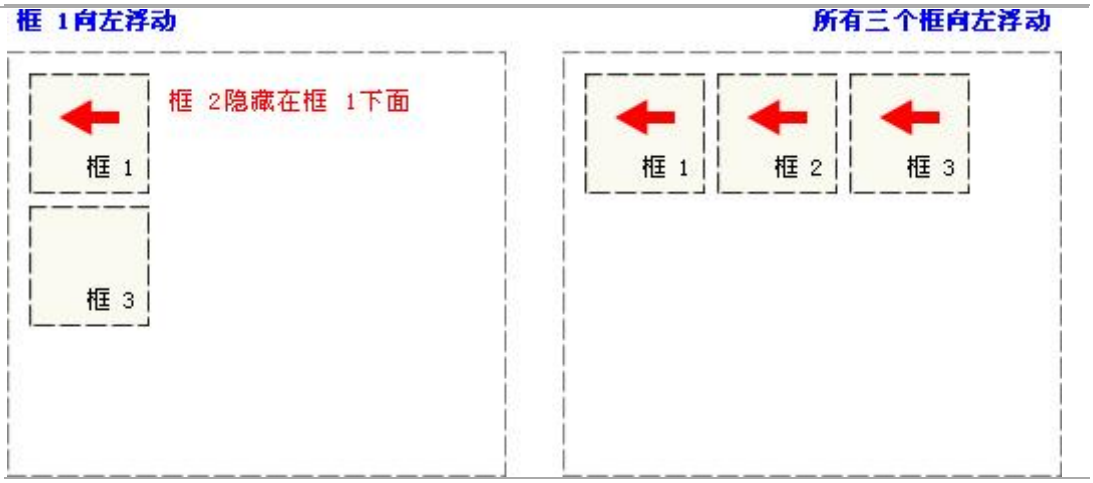
CSS 浮动

请看下图，当把框 **1** 向右浮动时，它脱离文档流并且向右移动，直到它的右边缘碰到包含框的右边缘：

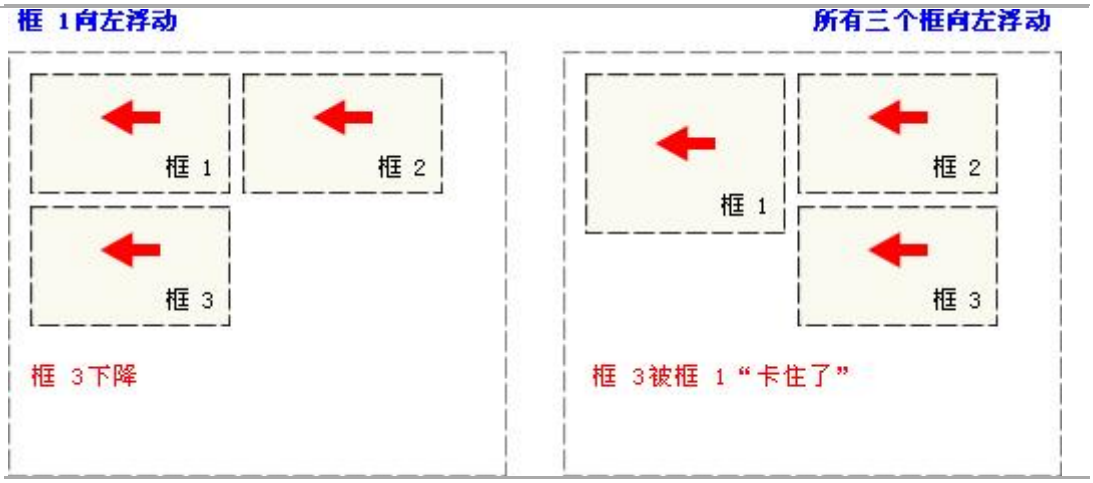


再请看下图，当框 **1** 向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框 **2**，使框 **2** 从视图中消失。

如果把所有三个框都向左移动，那么框 **1** 向左浮动直到碰到包含框，另外两个框向左浮动直到碰到前一个浮动框。



如下图所示，如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动，直到有足够的空间。如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素"卡住"：



CSS float 属性

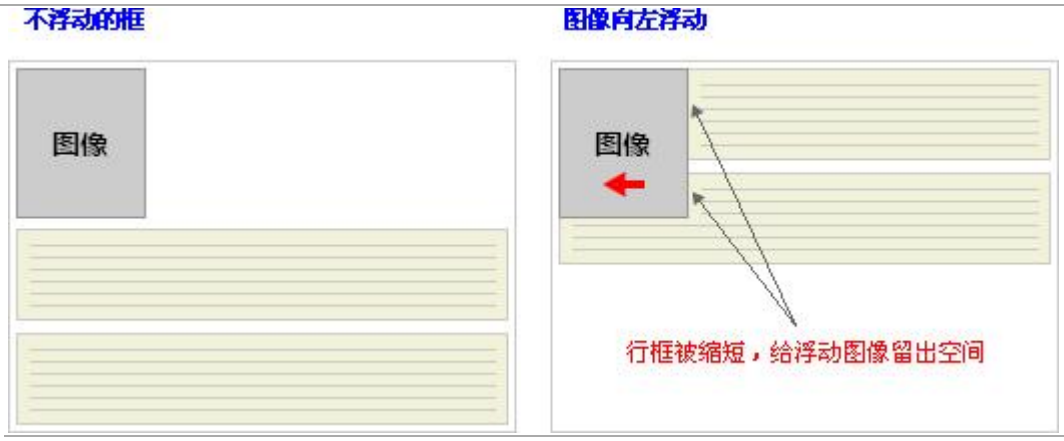
在 **CSS** 中，我们通过 **float** 属性实现元素的浮动。

如需更多有关 **float** 属性的知识，请访问参考手册：**CSS float** 属性。

行框和清理

浮动框旁边的行框被缩短，从而给浮动框留出空间，行框围绕浮动框。

因此，创建浮动框可以使文本围绕图像：



要想阻止行框围绕浮动框，需要对该框应用 **clear** 属性。**clear** 属性的值可以是 **left**、**right**、**both** 或 **none**，它表示框的哪些边不应该挨着浮动框。

为了实现这种效果，在被清理的元素的**上外边距**上添加足够的空间，使元素的顶边缘垂直下降到浮动框下面：

这是一个有用的工具，它让周围的元素为浮动元素留出空间。

让我们更详细地看看浮动和清理。假设希望让一个图片浮动到文本块的左边，并且希望这幅图片和文本包含在另一个具有背景颜色和边框的元素中。您可能编写下面的代码：

```
.news {
  background-color: gray;
  border: solid 1px black;
  }

.news img {
```

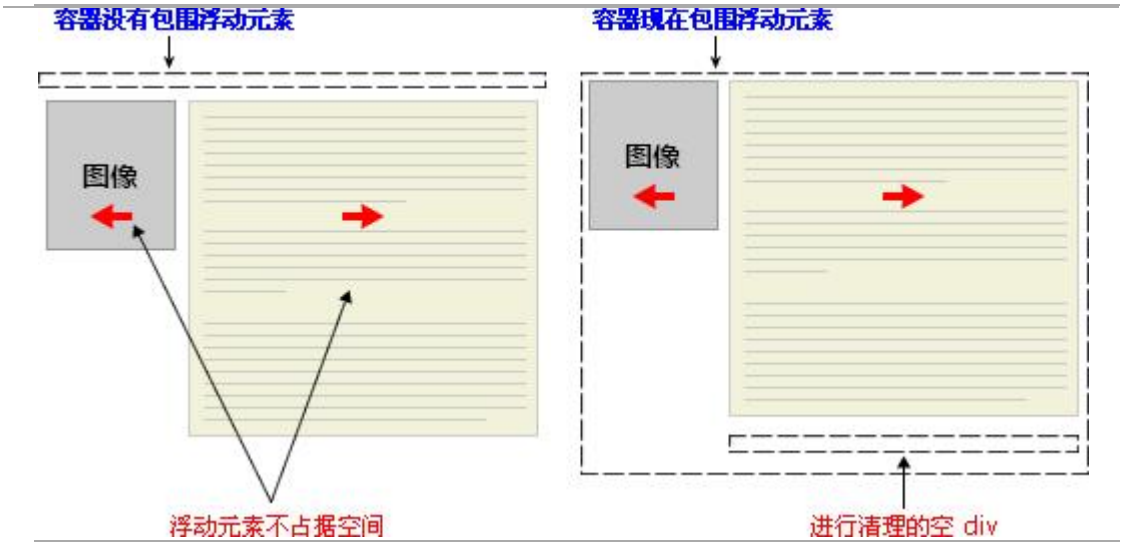
```
  float: left;
  }

.news p {
  float: right;
  }

<div class="news"><img src="news-pic.jpg" /><p>some text</p></div>
```

这种情况下，出现了一个问题。因为浮动元素脱离了文档流，所以包围图片和文本的 **div** 不占据空间。

如何让包围元素在视觉上包围浮动元素呢？需要在这个元素中的某个地方应用 **clear**：



不幸的是出现了一个新的问题，由于没有现有的元素可以应用清理，所以我们只能添加一个空元素并且清理它。

```
.news {
  background-color: gray;
  border: solid 1px black;
  }

.news img {
  float: left;
  }

.news p {
  float: right;
  }

.clear {
  clear: both;
  }<div class="news"><img src="news-pic.jpg" /><p>some text</p><div
class="clear"></div></div>
```

这样可以实现我们希望的效果，但是需要添加多余的代码。常常有元素可以应用 **clear**，但是有时候不得不为了进行布局而添加无意义的标记。

不过我们还有另一种办法，那就是对容器 **div** 进行浮动：

```
.news {
  background-color: gray;
  border: solid 1px black;
  float: left;
  }

.news img {
  float: left;
  }

.news p {
  float: right;
  }

<div class="news"><img src="news-pic.jpg" /><p>some text</p></div>
```

这样会得到我们希望的效果。不幸的是，下一个元素会受到这个浮动元素的影响。为了解决这个问题，有些人选择对布局中的所有东西进行浮动，然后使用适当的有意义的元素（常常是站点的页脚）对这些浮动进行清理。这有助于减少或消除不必要的标记。

事实上，**W3School** 站点上的所有页面都采用了这种技术，如果您打开我们使用 **CSS** 文件，您会看到我们对页脚的 **div** 进行了清理，而页脚上面的三个 **div** 都向左浮动。

CSS clear 属性

我们刚才详细讨论了 **CSS** 清理的工作原理和 **clear** 属性应用方法。如果您希望学习更多有关 **clear** 属性的知识，请访问参考手册：**CSS clear** 属性。

浮动和清理 实例

float 属性的简单应用

使图像浮动于一个段落的右侧。

将带有边框和边界的图像浮动于段落的右侧

使图像浮动于段落的右侧。向图像添加边框和边界。

带标题的图像浮动于右侧

使带有标题的图像浮动于右侧

使段落的首字母浮动于左侧

使段落的首字母浮动于左侧，并向这个字母添加样式。

创建水平菜单

使用具有一栏超链接的浮动来创建水平菜单。

创建无表格的首页

使用浮动来创建拥有页眉、页脚、左侧目录和主体内容的首页。

清除元素的侧面

本例演示如何使用清除元素侧面的浮动元素。

CSS 经典 教程提高篇--CSS 高级

CSS 尺寸 (Dimension)

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。

CSS 尺寸实例：

使用像素值设置图像的高度

本例演示如何使用像素值设置元素的高度。

使用百分比设置图像的高度

本例演示如何使用百分比值来设置元素的高度。

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。

使用百分比来设置元素的宽度

本例演示如何使用百分比值来设置元素的宽度。

设置元素的最大高度

本例演示如何设置一个元素的最大高度。

使用像素值来设置元素的最大宽度

本例演示如何使用像素值来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高度。

使用像素值来设置元素的最小高度

本例演示如何使用像素值来设置元素的最小高度。

使用像素值来设置元素的最小宽度

本例演示如何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

本例演示如何使用百分比值来设置元素的最小宽度。

使用百分比设置行间距

本例演示如何使用百分比值来设置段落中的行间距。

使用像素值设置行间距

本例演示如何使用像素值来设置段落中的行间距。

使用数值来设置行间距

本例演示如何使用一个数值来设置段落中的行间距。

CSS 尺寸属性

CSS 尺寸属性允许你控制元素的高度和宽度。同样，还允许你增加行间距。

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 分类（Classification）

CSS 分类属性允许你规定如何以及在何处显示元素。

CSS 分类(Classification)实例：

如何把元素显示为内联元素

本例演示如何把元素显示为内联元素。

如何把元素显示为块级元素

本例演示如何把元素显示为块级元素。

float 属性的简单应用

使图像浮动于一个段落的右侧。

将带有边框和边界的图像浮动于段落的右侧

使图像浮动于段落的右侧。向图像添加边框和边界。

带标题的图像浮动于右侧

使带有标题的图像浮动于右侧

使段落的首字母浮动于左侧

使段落的首字母浮动于左侧，并向这个字母添加样式。

创建水平菜单

使用具有一栏超链接的浮动来创建水平菜单。

创建无表格的首页

使用浮动来创建拥有页眉、页脚、左侧目录和主体内容的首页。

定位：相对定位

本例演示如何相对于一个元素的正常位置来对其定位。

定位：绝对定位

本例演示如何使用绝对值来对元素进行定位。

定位：固定定位

本例演示如何相对于浏览器窗口来对元素进行定位。

如何使元素不可见

本例演示如何使元素不可见。你希望元素被显示出来，还是不呢？

把表格元素设置为 **collapse**（请在非 **IE** 的浏览器中查看）

本例演示如何使表格元素叠加？

改变光标

本例演示如何改变光标。

清除元素的侧面

本例演示如何使用清除元素侧面的浮动元素。

CSS 分类属性 (Classification)

CSS 分类属性允许你控制如何显示元素，设置图像显示于另一元素中的何处，相对于其正常位置来定位元素，使用绝对值来定位元素，以及元素的可见度。

属性	描述
clear	设置一个元素的侧面是否允许其他的浮动元素。
cursor	规定当指向某元素之上时显示的指针类型。
display	设置是否及如何显示元素。
float	定义元素在哪个方向浮动。
position	把元素放置到一个静态的、相对的、绝对的、或固定的位置中。
visibility	设置元素是否可见或不可见。

CSS 伪类 (Pseudo-classes)

CSS 伪类用于向某些选择器添加特殊的效果。

CSS 伪类 (Pseudo-classes)实例：

超链接

本例演示如何向文档中的超链接添加不同的颜色。

超链接 2

本例演示如何向超链接添加其他样式。

超链接 - :focus 的使用

本例演示如何对超链接应用 :focus 伪类（无法在 IE 中工作）。

:first-child（首个子对象）

本例演示 :first-child 伪类的用法。

:lang（语言）

本例演示 :lang 伪类的用法。

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。

请访问该链接，来查看这个 **:first-child** 实例的效果。

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>**，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

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

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 {
  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

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

浏览器支持：IEInternet Explorer, F: Firefox, N: Netscape。

W3C：“W3C” 列的数字显示出伪类属性由哪个 **CSS** 标准定义（**CSS1** 还是 **CSS2**）。

伪类	作用	IE	F	N	W3C
:active	将样式添加到被激活的元素	4	1	8	1
:focus	将样式添加到被选中的元素	-	-	-	2
:hover	当鼠标悬浮在元素上方时，向元素添加样式	4	1	7	1
:link	将特殊的样式添加到未被访问过的链接	3	1	4	1
:visited	将特殊的样式添加到被访问过的链接	3	1	4	1
:first-child	将特殊的样式添加到元素的第一个子元素		1	7	2
:lang	允许创作者来定义指定的元素中使用的语言		1	8	2

CSS 伪元素 (Pseudo-elements)

CSS 伪元素用于将特殊的效果添加到某些选择器。

CSS 伪元素 (Pseudo-elements)实例：

制作首字母特效

本例演示如何向文本的首字母添加特效。

制作首行特效

本例演示如何向文本的首行添加特效。

语法：

伪元素的语法：

```
选择器 ： 伪元素 { 属性： 值 }
```

CSS 类也可以与伪元素配合使用：

```
选择器 . 类： 伪元素 { 属性： 值 }
```

:first-line 伪元素

"first-line" 伪元素用于向某个选择器中的文字的首行添加特殊样式：

```
p {font-size: 12pt}
p:first-line {color: #0000FF; font-variant: small-caps}

<p>Some text that ends up on two or more lines</p>
```

在上面的例子中，浏览器显示根据 **first-line** 伪元素格式化的第一行。浏览器是依靠浏览器窗口的尺寸来进行分行的。

提示：**first-line** 伪元素仅能被用于块级元素。

提示：下面的属性可以被应用到 **first-line** 伪元素。

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

:first-letter 伪元素

first-letter 伪元素用于向某个选择器中的文本的首字母添加特殊的样式：

```
p {font-size: 12pt}
p:first-letter {font-size: 200%; float: left}

<p>The first words of an article.</p>
```

输出的效果类似于：

```
——
  |  he first
  |  words of an
article.
```

- **font** 属性
- **color** 属性
- **background** 属性
- **margin** 属性
- **padding** 属性
- **border** 属性
- **text-decoration**
- **vertical-align (**仅当 **float** 为 **none** 时**)**
- **text-transform**
- **line-height**
- **float**
- **clear**

伪元素和 CSS 类

伪元素可以与 **CSS** 类配合使用：

```
p.article:first-letter {color: #FF0000}
<p class="article">文章中的一个段落。</p>
```

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

多重伪元素

多个伪元素可以配合在一起使用：

```
p {font-size: 12pt;}
p:first-letter {color: #FF0000; font-size: 24pt;}
p:first-line {color: #0000FF;}
<p>The first words of an article</p>
```

输出的效果类似于：

```
——
  |  he first
  |  words of an
article.
```

在上面的例子中，段落的首字母将是字号为 **24pt** 的红色。首行的其余部分将会是蓝色，而段落的其余部分会是默认的颜色。

CSS2 - :before 伪元素

before 伪元素可用于在某个元素之前插入某些内容。

下面的样式会在标题之前播放音频：

```
h1:before
{
content: url(beep.wav)
}
```

CSS2 - :after 伪元素

after 伪类可用于在某个元素之后插入某些内容。

下面的样式会在标题之后播放音频：

```
h1:after
{
content: url(beep.wav)
}
```

伪元素

浏览器支持：IE: Internet Explorer, F: Firefox, N: Netscape。

W3C: "W3C"列的数字显示出属性背景由哪个 CSS 标准定义（CSS1 还是 CSS2）。

伪元素	作用	IE	F	N	W3C
:first-letter	将特殊的样式添加到文本的首字母	5	1	8	1
:first-line	将特殊的样式添加到文本的首行	5	1	8	1
:before	在某元素之前插入某些内容		1.5	8	2
:after	在某元素之后插入某些内容		1.5	8	2

CSS2 媒介类型

媒介类型(Media Types)允许你定义以何种媒介来提交文档。文档可以被显示在显示器、纸媒介或者听觉浏览器等等。

媒介类型

某些 **CSS** 属性仅仅被设计为针对某些媒介。比方说 **"voice-family"** 属性被设计为针对听觉用户终端。其他的属性可被用于不同的媒介。例如，**"font-size"** 属性可被用于显示器以及印刷媒介，但是也许会带有不同的值。显示器上面的显示的文档通常会需要比纸媒介文档更大的字号，同时，在显示器上，**sans-serif** 字体更易阅读，而在纸媒介上，**serif** 字体更易阅读。

@media 规则

@media 规则使你有能力在相同的样式表中，使用不同的样式规则来针对不同的媒介。

下面这个例子中的样式告知浏览器在显示器上显示 **14** 像素的 **Verdana** 字体。但是假如页面需要被打印，将使用 **10** 个像素的 **Times** 字体。注意：**font-weight** 被设置为粗体，不论显示器还是纸媒介：

```
<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>
```

不同的媒介类型

注释：媒介类型名称对大小写不敏感。

浏览器支持：IE: Internet Explorer, F: Firefox, N: Netscape。

W3C: **"W3C"** 列的数字显示出属性背景由哪个 **CSS** 标准定义（**CSS1** 还是 **CSS2**）。

媒介类型	描述
all	用于所有的媒介设备。
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介，比如电传打字机和终端。

tv	用于电视机类型的设备。

CSS Don't

本节列出了在使用 CSS 时尽量避免使用的技术。

Internet Explorer Behaviors

它是什么？Internet Explorer 5 引入了行为 **(behaviors)**。**behaviors** 是一种通过使用 **CSS** 向 **HTML** 元素添加行为的方法。

为什么要避免它？只有 **Internet Explorer** 支持 **behavior** 属性。

用什么代替？请使用 **JavaScript** 和 **HTML DOM** 取而代之。

例子 1 - Mouseover Highlight

下面的 **HTML** 文件中有一个 **<style>** 元素，它为 **<h1>** 元素定义了一个行为：

```
<html><head><style type="text/css">
h1 { behavior: url(behave.htc) }
</style></head><body><h1>Mouse over me!!!</h1></body></html>
```

下面是 **XML** 文档 **"behave.htc"**：

```
<attach for="element" event="onmouseover" handler="hig_lite" /><attach
for="element" event="onmouseout" handler="low_lite" /><script
type="text/javascript">
function hig_lite()
{
element.style.color='red';
}

function low_lite()
{
element.style.color='blue';
}
</script>
```

behavior 文件包含了针对元素的 **JavaScript** 和 事件句柄。

如果您使用 **Internet Explorer**，可以亲自试一下（把鼠标放在例子中的文本上）。

例子 2 - Typewriter Simulation

下面的 **HTML** 文件中有一个 **<style>** 元素，它为 **id** 为 **"typing"** 的元素定义了一个行为：

```
<html><head><style type="text/css">
#typing
{
behavior:url(behave_typing.htc);
font-family:'courier new';
```

```
}
</style></head><body><span id="typing" speed="100">IE5 introduced DHTML
behaviors.
Behaviors are a way to add DHTML functionality to HTML elements
with the ease of CSS.<br /><br />How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p></span></body></html>
```

下面是 **XML** 文档 **"behave.htc"**:

```
<attach for="window" event="onload" handler="beginTyping" /><method
name="type" /><script type="text/javascript">
var i,text1,text2,textLength,t;

function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
  {
  clearInterval(t);
  }
}
</script>
```

如果您使用 **Internet Explorer**，可以亲自试一下。

你已经学习了 CSS，下一步学习什么呢？

CSS 概要

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，请参阅我们对 **CSS** 实例 和 **CSS** 参考手册 。

你已经学习了 CSS，下一步学习什么呢？

下一步是 **XHTML** 和 **JavaScript** 。

XHTML

XHTML 是新的 **HTML**。最新的 **HTML** 标准是 **HTML 4.01**，也是最终的版本。

HTML 将被 **XHTML** 取而代之，而 **XHTTML** 是更严格更纯净的 **HTML**。

加入你希望学习更多关于 **XHTML** 的知识，请访问我们对 **XHTML** 教程 。

JavaScript

JavaScript 可使你的站点动态性更强。

当你只是希望展示单一的内容时，静态站点也可以应付得了。但是动态的站点可以对事件作出反应，并允许用户进行交互。

JavaScript 是因特网上最流行的脚本语言，它可以工作于所有的主流浏览器中。

加入你希望学习更多关于 **JavaScript** 的知识，请访问我们的 **JavaScript** 教程 。

CSS 经典 教程—CSS 实例

提示： 以下例子中的 **CSS** 代码均位于 **HTML** 的 **head** 部分，这样做的目的是为了利于演示例子本身。在实际的开发中，使用 **CSS** 最好的方式是引用外部样式表。

CSS 背景实例：

设置背景颜色

本例演示如何为元素设置背景颜色。

设置文本的背景颜色

本例颜色如何设置部分文本的背景颜色。

将图像设置为背景

本例演示如何将图像设置为背景。

如何重复背景图像

本例演示如何重复背景图像。

如何在垂直方向重复背景图像

本例演示如何垂直地重复背景图像。

如何在水平方向重复背景图像

本例演示如何水平地重复背景图像。

如何仅显示一次背景图像

本例演示如何仅显示一次背景图像。

如何放置背景图像

本例演示如何在页面上放置背景图像。

如何使用%来定位背景图像

本例演示如何使用百分比来在页面上定位背景图像。

如何使用像素来定位背景图像

本例演示如何使用像素来在页面上定位背景图像。

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。

所有背景属性在一个声明之中

本例演示如何使用简写属性来将所有背景属性设置在一个声明之中。

例子解释

CSS 文本实例：

设置文本颜色

本例演示如何设置文本的颜色。

设置文本的背景颜色

本例颜色如何设置部分文本的背景颜色。

规定字符间距

本例演示如何增加或减少字符间距。

使用百分比设置行间距

本例演示如何使用百分比值来设置段落中的行间距。

使用像素值设置行间距

本例演示如何使用像素值来设置段落中的行间距。

使用数值来设置行间距

本例演示如何使用一个数值来设置段落中的行间距。

对齐文本

本例演示如何对齐文本。

修饰文本

本例演示如何向文本添加修饰。

缩进文本

本例演示如何缩进文本首行。

控制文本中的字母

本例演示如何控制文本中的字母。

在元素中禁止文本折行

本例演示如何禁止在元素中的文本折行。

增加单词间距

本例演示如何增加段落中单词间的距离。

例子解释

CSS 字体(font)实例：

设置文本的字体

本例演示如何设置文本字体。

设置字体尺寸

本例演示如何设置字体尺寸。

设置字体风格

本例演示如何设置字体风格。

设置字体的异体

本例演示如何设置字体的异体。

设置字体的粗细

本例演示如何设置字体的粗细。

所有字体属性在一个声明之内

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。

例子解释

CSS 边框(border)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。

设置四边框样式

本例演示如何设置四边框样式。

设置每一边的不同边框

本例演示如何在元素的各边设置不同的边框。

所有边框宽度属性在一个声明之中

本例演示用简写属性来将所有边框宽度属性设置于同一声明中。

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。

所有下边框属性在一个声明中

本例演示用简写属性来将所有下边框属性设置在同一声明中。

设置下边框的颜色

本例演示如何设置下边框的颜色。

设置下边框的样式

本例演示如何设置下边框的样式。

设置下边框的宽度

本例演示如何设置下边框的宽度。

所有左边框属性在一个声明之中

所有左边框属性在一个声明之中

设置左边框的颜色

本例演示如何设置左边框的颜色。

设置左边框的样式

本例演示如何设置左边框的样式。

设置左边框的宽度

本例演示如何设置左边框的宽度。

所有右边框属性在一个声明之中

本例演示一个简写属性，用于把所有右边框属性设置在一条声明中。

设置右边框的颜色

本例演示如何设置右边框的颜色。

设置右边框的样式

本例演示如何设置右边框的样式。

设置右边框的宽度

本例演示如何设置右边框的宽度。

所有上边框属性在一个声明之中

本例演示用简写属性来将所有上边框属性设置于同一声明之中。

设置上边框的颜色

本例演示如何设置上边框的颜色。

设置上边框的样式

本例演示如何设置上边框的样式。

设置上边框的宽度

本例演示如何设置上边框的宽度。

例子解释

CSS 外边距 (margin) 实例：

设置文本的左外边距

本例演示如何设置文本的左外边距。

设置文本的右外边距

本例演示如何设置文本的右外边距。

设置文本的上外边距

本例演示如何设置文本的上外边距。

设置文本的下外边距

本例演示如何设置文本的下外边距。

所有的外边距属性在一个声明中。

本例演示如何将所有的外边距属性设置于一个声明中。

例子解释

CSS 内边距 (padding) 实例：

所有填充属性在一个声明中

本例演示使用简写属性将所有的填充属性设置于一个声明中，可以有一到四个值。

设置下内边距 **1**

本例演示如何使用厘米值来设置单元格的下内边距。

设置下内边距 **2**

本例演示如何使用百分比值来设置单元格的下内边距。

设置左内边距 **1**

本例演示如何使用厘米值来设置单元格的左内边距。

设置左内边距 **2**

本例演示如何使用百分比值来设置单元格的左内边距。

设置右内边距 **1**

本例演示如何使用厘米值来设置单元格的右内边距。

设置右内边距 **2**

本例演示如何使用百分比值来设置单元格的右内边距。

设置上内边距 **1**

本例演示如何使用厘米值来设置单元格的上内边距。

设置上内边距 **2**

本例演示如何使用百分比值来设置单元格的上内边距。

例子解释

CSS 列表实例：

在无序列表中的不同类型的列表标记

本例演示在 **CSS** 中不同类型的列表项标记。

在有序列表中不同类型的列表项标记

本例演示在 **CSS** 中不同类型的列表项标记。

所有的列表样式类型

本例演示在 **CSS** 中所有不同类型的列表项标记。

将图像作为列表项标记

本例演示如何将图像作为列表项标记。

放置列表标记

本例演示在何处放置列表标记。

在一个声明中定义所有的列表属性

本例演示将所有针对列表的属性设置于一个简写属性。

例子解释

CSS 表格实例：

设置表格的布局

本例演示如何设置表格的布局。

显示表格中的空单元

本例演示是否显示表格中的空单元。（请在非 **IE** 浏览器中浏览）

合并表格边框

本例演示是否把表格边框显示为一条单独的边框，还是像标准的 **HTML** 中那样分开显示。

设置表格边框之间的空白

本例演示如何设置单元格边框之间的距离。（请在非 **IE** 浏览器中浏览）

设置表格标题的位置

本例演示如何定位表格的标题。（请在非 **IE** 浏览器中浏览）

例子解释

轮廓（Outlin） 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。

设置轮廓的颜色

本例演示如何设置轮廓的颜色。

设置轮廓的样式

本例演示如何设置轮廓的样式。

设置轮廓的宽度

本例演示如何设置轮廓的宽度。

CSS 尺寸 (Dimension) 实例：

使用像素值设置图像的高度

本例演示如何使用像素值设置元素的高度。

使用百分比设置图像的高度

本例演示如何使用百分比值来设置元素的高度。

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。

使用百分比来设置元素的宽度

本例演示如何使用百分比值来设置元素的宽度。

设置元素的最大高度

本例演示如何设置一个元素的最大高度。

使用像素值来设置元素的最大宽度

本例演示如何使用像素值来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高度。

使用像素值来设置元素的最小高度

本例演示如何使用像素值来设置元素的最小高度。

使用像素值来设置元素的最小宽度

本例演示如何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

本例演示如何使用百分比值来设置元素的最小宽度。

使用百分比设置行间距

本例演示如何使用百分比值来设置段落中的行间距。

使用像素值设置行间距

本例演示如何使用像素值来设置段落中的行间距。

使用数值来设置行间距

本例演示如何使用一个数值来设置段落中的行间距。

例子解释

CSS 分类 (Classification) 实例：

如何把元素显示为内联元素

本例演示如何把元素显示为内联元素。

如何把元素显示为块级元素

本例演示如何把元素显示为块级元素。

浮动属性的简单应用

使图像浮动于一个段落的右侧。

将带有边框和边界的图像浮动于段落的右侧

使图像浮动于段落的右侧。向图像添加边框和边界。

带标题的图像浮动于右侧

使带有标题的图像浮动于右侧

使段落的首字母浮动于左侧

使段落的首字母浮动于左侧，并向这个字母添加样式。

创建水平菜单

使用具有一栏超链接的浮动来创建水平菜单。

创建无表格的首页

使用浮动来创建拥有页眉、页脚、左侧目录和主体内容的首页。

定位：相对定位

本例演示如何相对于一个元素的正常位置来对其定位。

定位：绝对定位

本例演示如何使用绝对值来对元素进行定位。

定位：固定定位

本例演示如何相对于浏览器窗口来对元素进行定位。

如何使元素不可见

本例演示如何使元素不可见。你希望元素被显示出来，还是不呢？

把表格元素设置为 **collapse**（请在非 **IE** 的浏览器中查看）

本例演示如何使表格元素叠加？

改变光标

本例演示如何改变光标。

清除元素的侧面

本例演示如何使用清除元素侧面的浮动元素。

例子解释

CSS 定位 (Positioning) 实例：

定位：相对定位

本例演示如何相对于一个元素的正常位置来对其定位。

定位：绝对定位

本例演示如何使用绝对值来对元素进行定位。

定位：固定定位

本例演示如何相对于浏览器窗口来对元素进行定位。

设置元素的形状

本例演示如何设置元素的形状。此元素被剪裁到这个形状内，并显示出来。

如何使用滚动条来显示元素内溢出的内容

本例演示当元素内容太大而超出规定区域时，如何设置溢出属性来规定相应的动作。

如何隐藏溢出元素中溢出的内容

本例演示在元素中的内容太大以至于无法适应指定的区域时，如何设置 **overflow** 属性来隐藏其内容。

如何设置浏览器来自动地处理溢出

本例演示如何设置浏览器来自动地处理溢出。

垂直排列图象

本例演示如何在文本中垂直排列图象。

Z-index

Z-index 可被用于将在一个元素放置于另一元素之后。

Z-index

上面的例子中的元素已经更改了 **Z-index**。

使用固定值设置图像的上边缘

本例演示如何使用固定值设置图像的上边缘。

使用百分比设置图像的上边缘

本例演示如何使用百分比值设置图像的上边缘。

使用像素值设置图像的底部边缘

本例演示如何使用像素值设置图像的底部边缘。

使用百分比设置图像的底部边缘

本例演示如何使用百分比值设置图像的底部边缘。

使用固定值设置图像的左边缘

本例演示如何使用固定值设置图像的左边缘。

使用百分比设置图像的左边缘

本例演示如何使用百分比值设置图像的左边缘。

使用固定值设置图像的右边缘

本例演示如何使用固定值设置图像的右边缘。

使用百分比设置图像的右边缘

本例演示如何使用百分比值设置图像的右边缘。

例子解释

CSS 伪类 (Pseudo-classes)实例：

超链接

本例演示如何向文档中的超链接添加不同的颜色。

超链接 2

本例演示如何向超链接添加其他样式。

超链接：:focus 的使用

本例演示如何使用 :focus 伪类（无法在 IE 中工作）。

:first-child（首个子对象）

本例演示 :first-child 伪类的用法。

:lang（语言）

本例演示 :lang 伪类的用法。

例子解释

CSS 伪元素 (Pseudo-elements)实例：

制作首字母特效

本例演示如何向文本的首字母添加特效。

制作首行特效

本例演示如何向文本的首行添加特效。

例子解释

CSS 经典 教程—参考手册

CSS 参考手册

请点击表格中属性列的链接，可以查看相关属性的详细信息。

CSS 属性组：

- 背景
- 文本
- 字体
- 边框和轮廓

- 外边距
- 内边距
- 列表
- 内容生成
- 尺寸
- 定位
- 打印
- 表格
- 伪类
- 伪元素

提示和注释：

属性："属性"列指向语法、实例、浏览器支持等内容。

CSS："CSS" 列指示属性是在哪个 **CSS** 版本中定义的（**CSS1** 还是 **CSS2**）。

提示：W3School 的 **CSS** 参考手册定期在所有主流浏览器中进行测试。最后测试时间：**2009** 年 **10** 月 **8** 日。

CSS 背景属性（Background）

属性	描述	CSS
background	在一个声明中设置所有的背景属性。	1
background-attachment	设置背景图像是否固定或者随着页面的其余部分滚动。	1
background-color	设置元素的背景颜色。	1
background-image	设置元素的背景图像。	1
background-position	设置背景图像的开始位置。	1
background-repeat	设置是否及如何重复背景图像。	1

CSS 边框属性（Border 和 Outline）

属性	描述	CSS
border	在一个声明中设置所有的边框属性。	1
border-bottom	在一个声明中设置所有的下边框属性。	1
border-bottom-color	设置下边框的颜色。	2

border-bottom-style	设置下边框的样式。	2
border-bottom-width	设置下边框的宽度。	1
border-color	设置四条边框的颜色。	1
border-left	在一个声明中设置所有的左边框属性。	1
border-left-color	设置左边框的颜色。	2
border-left-style	设置左边框的样式。	2
border-left-width	设置左边框的宽度。	1
border-right	在一个声明中设置所有的右边框属性。	1
border-right-color	设置右边框的颜色。	2
border-right-style	设置右边框的样式。	2
border-right-width	设置右边框的宽度。	1
border-style	设置四条边框的样式。	1
border-top	在一个声明中设置所有的上边框属性。	1
border-top-color	设置上边框的颜色。	2
border-top-style	设置上边框的样式。	2
border-top-width	设置上边框的宽度。	1
border-width	设置四条边框的宽度。	1
outline	在一个声明中设置所有的轮廓属性。	2
outline-color	设置轮廓的颜色。	2
outline-style	设置轮廓的样式。	2
outline-width	设置轮廓的宽度。	2

CSS 文本属性（Text）

属性	描述	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-shadow	规定添加到文本的阴影效果。	2
text-transform	控制文本的大小写。	1
unicode-bidi	设置文本方向。	2
white-space	规定如何处理元素中的空白。	1
word-spacing	设置单词间距。	1

CSS 字体属性（Font）

属性	描述	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-size-adjust	为元素规定 aspect 值。	2
font-stretch	收缩或拉伸当前的字体系列。	2
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

CSS 外边距属性（Margin）

属性	描述	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

CSS 内边距属性（Padding）

属性	描述	CSS
padding	在一个声明中设置所有内边距属性。	1
padding-bottom	设置元素的下内边距。	1
padding-left	设置元素的左内边距。	1
padding-right	设置元素的右内边距。	1
padding-top	设置元素的上内边距。	1

CSS 列表属性（List）

属性	描述	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
marker-offset		2

内容生成（Generated Content）

属性	描述	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

CSS 尺寸属性（Dimension）

属性	描述	CSS
height	设置元素高度。	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

CSS 定位属性（Positioning）

属性	描述	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	2

	偏移。	
vertical-align	设置元素的垂直对齐方式。	1
visibility	规定元素是否可见。	2
z-index	设置元素的堆叠顺序。	2

CSS 打印属性（Print）

属性	描述	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

CSS 表格属性（Table）

属性	描述	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

CSS 伪类（Pseudo-classes）

属性	描述	CSS
:active	向被激活的元素添加样式。	1
:focus	向拥有键盘输入焦点的元素添加样式。	2
:hover	当鼠标悬浮在元素上方时，向元素添加样式。	1
:link	向未被访问的链接添加样式。	1

:visited	向已被访问的链接添加样式。	1
:first-child	向元素的第一个子元素添加样式。	2
:lang	向带有指定 lang 属性的元素添加样式。	2

CSS 伪元素（Pseudo elements）

属性	描述	CSS
:first-letter	向文本的第一个字母添加特殊样式。	1
:first-line	向文本的首行添加特殊样式。	1
:before	在元素之前添加内容。	2
:after	在元素之后添加内容。	2

CSS2 打印参考

打印属性

打印 **HTML** 文档总是会出现问题。在 **CSS2** 中，我们可以借助打印属性让打印 **web** 内容更容易一些。

属性	描述	值
orphans	设置元素放在页面底部时所允许的最少文本行数。	number
marks	设置是否在内容区之外但是在画布的可打印区域内放“十字标志”。 请注意，CSS2.1 已删除该属性。	• none • crop • cross
page	这个属性与 size 属性结合可以指定打印一个元素时所用的特定页面类型。 请注意，CSS2.1 已删除该属性。	auto identifier
page-break-after	设置元素后是否应当放置分页符。	• auto • always • avoid • left • right
page-break-before	设置元素前否应当放置分页符。	• auto • always • avoid

		• left • right
page-break-inside	设置元素内部是否应当放置分页符。	• auto • avoid
size	利用这个属性，创作人员可以声明打印一个元素时所用页框的大小和方向。它可以与 page 结合使用。不过并不要求一定如此。 请注意，CSS2.1 已删除该属性。	• auto • portrait • landscape
widows	设置元素放在页面顶部时所允许的最少文本行数。	number

CSS2 听觉参考

听觉样式表

听觉样式表可把语音合成与音响效果相组合，使用户可以听到信息，而无需进行阅读。

听觉呈现可用于：

- 视觉能力低弱的人士
- 帮助用户学习阅读
- 帮助有阅读障碍的用户
- 家庭娱乐
- 在汽车中使用

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")
}
```

上面的例子可以让语音合成器演奏一段声音，然后用男性的声音读出标题。

CSS2 听觉参考

W3C :"W3C" 列的数字显示出属性由哪个 **CSS** 标准定义（**CSS1** 还是 **CSS2**）。

属性	描述	值	W3C

azimuth	Sets where the sound/voices should come from (horizontally)	• angle • left-side • far-left • left • center-left • center • center-right • right • far-right • right-side • behind • leftwards • rightwards	2
cue	A shorthand property for setting the cue-before and cue-after properties in one declaration	• cue-before • cue-after	2
cue-after	Specifies a sound to be played after speaking an element's content to delimit it from other	• none • url	2
cue-before	Specifies a sound to be played before speaking an element's content to delimit it from other	• none • url	2
elevation	Sets where the sound/voices should come from (vertically)	• angle • below • level • above • higher • lower	2
pause	A shorthand property for setting the pause-before and pause-after properties in one declaration	• pause-before • pause-after	2
pause-after	Specifies a pause after speaking an element's content	• time • %	2
pause-before	Specifies a pause before speaking an element's content	• time • %	2

pitch	Specifies the speaking voice	• frequency • x-low • low • medium • high • x-high	2
pitch-range	Specifies the variation in the speaking voice. (Monotone voice or animated voice?)	• number	2
play-during	Specifies a sound to be played while speaking an element's content	• auto • none • url • mix • repeat	2
richness	Specifies the richness in the speaking voice. (Rich voice or thin voice?)	• number	2
speak	Specifies whether content will render aurally	• normal • none • spell-out	2
speak-header	Specifies how to handle table headers. Should the headers be spoken before every cell, or only before a cell with a different header than the previous cell	• always • once	2
speak-numeral	Specifies how to speak numbers	• digits • continuous	2
speak-punctuation	Specifies how to speak punctuation characters	• none • code	2
speech-rate	Specifies the speed of the speaking	• number • x-slow • slow • medium • fast • x-fast	2

		• faster • slower	
stress	Specifies the "stress" in the speaking voice	• number	2
voice-family	A prioritized list of voice family names that contain specific voices	• specific-voice • generic-voice	2
volume	Specifies the volume of the speaking	• number • % • silent • x-soft • soft • medium • loud • x-loud	2

CSS 单位

尺寸

单位	描述
%	百分比
in	英寸
cm	厘米
mm	毫米
em	1em 等于当前的字体尺寸。 2em 等于当前字体尺寸的两倍。 例如，如果某元素以 12pt 显示，那么 2em 是 24pt。 在 CSS 中，em 是非常有用的单位，因为它可以自动适应用户所使用的字体。
ex	一个 ex 是一个字体的 x-height。 (x-height 通常是字体尺寸的一半。)
pt	磅 (1 pt 等于 1/72 英寸)
pc	12 点活字 (1 pc 等于 12 点)

px	像素 (计算机屏幕上的一个点)

颜色

单位	描述
(颜色名)	颜色名称 (比如 red)
rgb(x,x,x)	RGB 值 (比如 rgb(255,0,0))
rgb(x%,x%,x%)	RGB 百分比值 (比如 rgb(100%,0%,0%))
#rrggbb	十六进制数 (比如 #ff0000)

CSS 经典 教程—CSS 测验

您可以通过 W3SCHOOL 的测验程序来测试您的 CSS 技能。

关于本测验

本测验包含 **20** 道题，每道题的最长答题时间是 **20** 分钟（这是由于每个 **session** 的默认有效时间是 **20** 钟）。

本测验是非官方的测试，它仅仅提供了一个了解您对 **CSS** 的掌握程度的工具。

测验会被记分

每道题的分值是 **1** 分。在您完成全部的 **20** 道题之后，系统会为您的测验打分，并提供您做错的题目的正确答案。其中，绿色为正确答案，而红色为错误答案。

现在就开始测验！祝您好运。