CSS学习

# 1.CSS 简介

**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

# 2.CSS 语法

## CSS 语法

CSS 规则由两个主要的部分构成：选择器，以及一条或多条声明:



选择器通常是您需要改变样式的 HTML 元素。

每条声明由一个属性和一个值组成。

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

**CSS 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 "\*/" 结束, 实例如下:

/\*这是个注释\*/  
p  
{  
text-align:center;  
/\*这是另一个注释\*/  
color:black;  
font-family:arial;  
}

# 3.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 <strong> 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>

# 4.CSS Id 和 Class

## 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。

**实例**

#para1  
{  
text-align:center;  
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：

**实例**

.center {text-align:center;}

# 5.CSS 属性选择器

## 对带有指定属性的 HTML 元素设置样式。

可以为拥有指定属性的 HTML 元素设置样式，而不仅限于 class 和 id 属性。

**注释：**只有在规定了 !DOCTYPE 时，IE7 和 IE8 才支持属性选择器。在 IE6 及更低的版本中，不支持属性选择。

## 属性选择器

下面的例子为带有 title 属性的所有元素设置样式：

[title]

{

color:red;

}

## 属性和值选择器 - 多个值

下面的例子为包含指定值的 title 属性的所有元素设置样式。适用于由空格分隔的属性值：

[title~=hello] { color:red; }

## 设置表单的样式

属性选择器在为不带有 class 或 id 的表单设置样式时特别有用：

input[type="text"]

{

width:150px;

display:block;

margin-bottom:10px;

background-color:yellow;

font-family: Verdana, Arial;

}

input[type="button"]

{

width:120px;

margin-left:35px;

display:block;

font-family: Verdana, Arial;

}

**CSS 选择器参考手册**

|  |  |
| --- | --- |
| **选择器** | **描述** |
| [[*attribute*]](http://www.w3school.com.cn/cssref/selector_attribute.asp) | 用于选取带有指定属性的元素。 |
| [[*attribute*=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value.asp) | 用于选取带有指定属性和值的元素。 |
| [[*attribute*~=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp) | 用于选取属性值中包含指定词汇的元素。 |
| [[*attribute*|=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value_start.asp) | 用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。 |
| [[*attribute*^=*value*]](http://www.w3school.com.cn/cssref/selector_attr_begin.asp) | 匹配属性值以指定值开头的每个元素。 |
| [[*attribute*$=*value*]](http://www.w3school.com.cn/cssref/selector_attr_end.asp) | 匹配属性值以指定值结尾的每个元素。 |
| [[*attribute*\*=*value*]](http://www.w3school.com.cn/cssref/selector_attr_contain.asp) | 匹配属性值中包含指定值的每个元素。 |

# 6.CSS 创建

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>

**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>

## 多重样式将层叠为一个

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

### 层叠次序

当同一个 HTML 元素被不止一个样式定义时，会使用哪个样式呢？

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

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

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

# 7.CSS 背景

**CSS 背景属性**

|  |  |
| --- | --- |
| **Property** | **描述** |
| [background](http://www.runoob.com/cssref/css3-pr-background.html) | 简写属性，作用是将背景属性设置在一个声明中。 |
| [background-attachment](http://www.runoob.com/cssref/pr-background-attachment.html) | 背景图像是否固定或者随着页面的其余部分滚动。 |
| [background-color](http://www.runoob.com/cssref/pr-background-color.html) | 设置元素的背景颜色。 |
| [background-image](http://www.runoob.com/cssref/pr-background-image.html) | 把图像设置为背景。 |
| [background-position](http://www.runoob.com/cssref/pr-background-position.html) | 设置背景图像的起始位置。 |
| [background-repeat](http://www.runoob.com/cssref/pr-background-repeat.html) | 设置背景图像是否及如何重复。 |

**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 right top;}

当使用简写属性时，属性值的顺序为：:

* background-color

|  |  |
| --- | --- |
| **值** | **描述** |
| *color\_name* | 规定颜色值为颜色名称的背景颜色（比如 red）。 |
| *hex\_number* | 规定颜色值为十六进制值的背景颜色（比如 #ff0000）。 |
| *rgb\_number* | 规定颜色值为 rgb 代码的背景颜色（比如 rgb(255,0,0)）。 |
| transparent | 默认。背景颜色为透明。 |
| inherit | 规定应该从父元素继承 background-color 属性的设置。 |

* background-image

|  |  |
| --- | --- |
| **值** | **描述** |
| url('*URL*') | 指向图像的路径。 |
| none | 默认值。不显示背景图像。 |
| inherit | 规定应该从父元素继承 background-image 属性的设置。 |

* background-repeat

|  |  |
| --- | --- |
| **值** | **描述** |
| repeat | 默认。背景图像将在垂直方向和水平方向重复。 |
| repeat-x | 背景图像将在水平方向重复。 |
| repeat-y | 背景图像将在垂直方向重复。 |
| no-repeat | 背景图像将仅显示一次。 |
| inherit | 规定应该从父元素继承 background-repeat 属性的设置。 |

* background-attachment

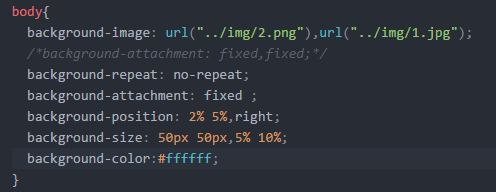
|  |  |
| --- | --- |
| **值** | **描述** |
| scroll | 默认值。背景图像会随着页面其余部分的滚动而移动。 |
| fixed | 当页面的其余部分滚动时，背景图像不会移动。 |
| inherit | 规定应该从父元素继承 background-attachment 属性的设置。 |

* background-position

|  |  |
| --- | --- |
| **值** | **描述** |
| * top left * top center * top right * center left * center center * center right * bottom left * bottom center * bottom right | 如果您仅规定了一个关键词，那么第二个值将是"center"。  默认值：0% 0%。 |
| x% y% | 第一个值是水平位置，第二个值是垂直位置。  左上角是 0% 0%。右下角是 100% 100%。  如果您仅规定了一个值，另一个值将是 50%。 |
| xpos ypos | 第一个值是水平位置，第二个值是垂直位置。  左上角是 0 0。单位是像素 (0px 0px) 或任何其他的 CSS 单位。  如果您仅规定了一个值，另一个值将是50%。  您可以混合使用 % 和 position 值。 |

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 |

  
背景大小和背景定位可以用百分比和长度值来表示，也可以多个背景图片，用”,”隔开

# 8.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](http://www.w3school.com.cn/cssref/pr_text_text-indent.asp)，该属性可以方便地实现文本缩进。

### 可能的值

|  |  |
| --- | --- |
| **值** | **描述** |
| *length* | 定义固定的缩进。默认值：0。 |
| *%* | 定义基于父元素宽度的百分比的缩进。 |
| inherit | 规定应该从父元素继承 text-indent 属性的值。 |

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](http://www.w3school.com.cn/cssref/pr_text_text-align.asp) 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。

### 可能的值

|  |  |
| --- | --- |
| **值** | **描述** |
| left | 把文本排列到左边。默认值：由浏览器决定。 |
| right | 把文本排列到右边。 |
| center | 把文本排列到中间。 |
| justify | 实现两端对齐文本效果。 |
| inherit | 规定应该从父元素继承 text-align 属性的值。 |

#### 值 justify

最后一个水平对齐属性是 justify，它会带来自己的一些问题。

值 justify 可以使文本的两端都对齐。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。不过在 CSS 中，还需要多做些考虑。

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。例如，有些浏览器可能只在单词之间增加额外的空间，而另外一些浏览器可能会平均分布字母间的额外空间（不过 CSS 规范特别指出，如果 [letter-spacing 属性](http://www.w3school.com.cn/cssref/pr_text_letter-spacing.asp)指定为一个长度值，“用户代理不能进一步增加或减少字符间的空间”）。还有一些用户代理可能会减少某些行的空间，使文本挤得更紧密。所有这些做法都会影响元素的外观，甚至改变其高度，这取决于用户代理的对齐选择影响了多少文本行。

CSS 也没有指定应当如何处理连字符（注1）。大多数两端对齐文本都使用连字符将长单词分开放在两行上，从而缩小单词之间的间隔，改善文本行的外观。不过，由于 CSS 没有定义连字符行为，用户代理不太可能自动加连字符。因此，在 CSS 中，两端对齐文本看上去没有打印出来好看，特别是元素可能太窄，以至于每行只能放下几个单词。当然，使用窄设计元素是可以的，不过要当心相应的缺点。

**注1：**CSS 中没有说明如何处理连字符，因为不同的语言有不同的连字符规则。规范没有尝试去调和这样一些很可能不完备的规则，而是干脆不提这个问题。

## 字间隔

[word-spacing 属性](http://www.w3school.com.cn/cssref/pr_text_word-spacing.asp)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 increased.

</p>

<p class="tight">

This is a paragraph. The spaces between words will be decreased.

</p>

|  |  |
| --- | --- |
| **值** | **描述** |
| normal | 默认。定义单词间的标准空间。 |
| *length* | 定义单词间的固定空间。 |
| inherit | 规定应该从父元素继承 word-spacing 属性的值。 |

## 字母间隔

[letter-spacing 属性](http://www.w3school.com.cn/cssref/pr_text_letter-spacing.asp)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>

|  |  |
| --- | --- |
| **值** | **描述** |
| normal | 默认。规定字符间没有额外的空间。 |
| *length* | 定义字符间的固定空间（允许使用负值）。 |
| inherit | 规定应该从父元素继承 letter-spacing 属性的值。 |

## 字符转换

[text-transform 属性](http://www.w3school.com.cn/cssref/pr_text_text-transform.asp)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。

## 文本装饰

接下来，我们讨论 [text-decoration 属性](http://www.w3school.com.cn/cssref/pr_text_text-decoration.asp)，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](http://www.w3school.com.cn/cssref/pr_text_white-space.asp)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。

|  |  |
| --- | --- |
| **值** | **描述** |
| normal | 默认。空白会被浏览器忽略。 |
| pre | 空白会被浏览器保留。其行为方式类似 HTML 中的 <pre> 标签。 |
| nowrap | 文本不会换行，文本会在在同一行上继续，直到遇到 <br> 标签为止。 |
| pre-wrap | 保留空白符序列，但是正常地进行换行。 |
| pre-line | 合并空白符序列，但是保留换行符。 |
| inherit | 规定应该从父元素继承 white-space 属性的值。 |

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](http://www.w3school.com.cn/cssref/pr_text_direction.asp)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位置。

**注释：**对于行内元素，只有当 [unicode-bidi 属性](http://www.w3school.com.cn/cssref/pr_unicode-bidi.asp" \o "CSS 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 文本属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [color](http://www.w3school.com.cn/cssref/pr_text_color.asp) | 设置文本颜色 |
| [direction](http://www.w3school.com.cn/cssref/pr_text_direction.asp) | 设置文本方向。 |
| [line-height](http://www.w3school.com.cn/cssref/pr_dim_line-height.asp) | 设置行高。 |
| [letter-spacing](http://www.w3school.com.cn/cssref/pr_text_letter-spacing.asp) | 设置字符间距。 |
| [text-align](http://www.w3school.com.cn/cssref/pr_text_text-align.asp) | 对齐元素中的文本。 |
| [text-decoration](http://www.w3school.com.cn/cssref/pr_text_text-decoration.asp) | 向文本添加修饰。 |
| [text-indent](http://www.w3school.com.cn/cssref/pr_text_text-indent.asp) | 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](http://www.w3school.com.cn/cssref/pr_text_text-transform.asp) | 控制元素中的字母。 |
| unicode-bidi | 设置文本方向。 |
| [white-space](http://www.w3school.com.cn/cssref/pr_text_white-space.asp) | 设置元素中空白的处理方式。 |
| [word-spacing](http://www.w3school.com.cn/cssref/pr_text_word-spacing.asp) | 设置字间距。 |

# 9.CSS 字体

**CSS 字体属性定义文本的字体系列、大小、加粗、风格（如斜体）和变形（如小型大写字母）。**

**CSS 字体系列**

在 CSS 中，有两种不同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（比如 "Serif" 或 "Monospace"）
* 特定字体系列 - 具体的字体系列（比如 "Times" 或 "Courier"）

除了各种特定的字体系列外，CSS 定义了 5 种通用字体系列：

* Serif 字体
* Sans-serif 字体
* Monospace 字体
* Cursive 字体
* Fantasy 字体

## 指定字体系列

使用 [font-family 属性](http://www.w3school.com.cn/cssref/pr_font_font-family.asp) 定义文本的字体系列。

### 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

除了使用通用的字体系列，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 Georgia 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 style="font-family: Times, TimesNR, 'New Century Schoolbook', Georgia,

'New York', serif;">...</p>

## 字体风格

[font-style 属性](http://www.w3school.com.cn/cssref/pr_font_font-style.asp)最常用于规定斜体文本。

该属性有三个值：

* normal - 文本正常显示
* italic - 文本斜体显示
* oblique - 文本倾斜显示

### 实例

p.normal {font-style:normal;}

p.italic {font-style:italic;}

p.oblique {font-style:oblique;}

### italic 和 oblique 的区别

font-style 非常简单：用于在 normal 文本、italic 文本和 oblique 文本之间选择。唯一有点复杂的是明确 italic 文本和 oblique 文本之间的差别。

斜体（italic）是一种简单的字体风格，对每个字母的结构有一些小改动，来反映变化的外观。与此不同，倾斜（oblique）文本则是正常竖直文本的一个倾斜版本。

通常情况下，italic 和 oblique 文本在 web 浏览器中看上去完全一样。

## 字体变形

[font-variant 属性](http://www.w3school.com.cn/cssref/pr_font_font-variant.asp)可以设定小型大写字母。

小型大写字母不是一般的大写字母，也不是小写字母，这种字母采用不同大小的大写字母。

### 实例

p {font-variant:small-caps;}

## 字体加粗

[font-weight 属性](http://www.w3school.com.cn/cssref/pr_font_weight.asp)设置文本的粗细。

使用 bold 关键字可以将文本设置为粗体。

关键字 100 ~ 900 为字体指定了 9 级加粗度。如果一个字体内置了这些加粗级别，那么这些数字就直接映射到预定义的级别，100 对应最细的字体变形，900 对应最粗的字体变形。数字 400 等价于 normal，而 700 等价于 bold。

如果将元素的加粗设置为 bolder，浏览器会设置比所继承值更粗的一个字体加粗。与此相反，关键词 lighter 会导致浏览器将加粗度下移而不是上移。

### 实例

p.normal {font-weight:normal;}

p.thick {font-weight:bold;}

p.thicker {font-weight:900;}

## 字体大小

[font-size 属性](http://www.w3school.com.cn/cssref/pr_font_font-size.asp)设置文本的大小。

有能力管理文本的大小在 web 设计领域很重要。但是，您不应当通过调整文本大小使段落看上去像标题，或者使标题看上去像段落。

请始终使用正确的 HTML 标题，比如使用 <h1> - <h6> 来标记标题，使用 <p> 来标记段落。

font-size 值可以是绝对或相对值。

绝对值：

* 将文本设置为指定的大小
* 不允许用户在所有浏览器中改变文本大小（不利于可用性）
* 绝对大小在确定了输出的物理尺寸时很有用

相对大小：

* 相对于周围的元素来设置大小
* 允许用户在浏览器改变文本大小

**注意：**如果您没有规定字体大小，普通文本（比如段落）的默认大小是 16 像素 (16px=1em)。

### 使用像素来设置字体大小

通过像素设置文本大小，可以对文本大小进行完全控制：

#### 实例

h1 {font-size:60px;}

h2 {font-size:40px;}

p {font-size:14px;}

在 Firefox, Chrome, and Safari 中，可以重新调整以上例子的文本大小，但是在 Internet Explorer 中不行。

虽然可以通过浏览器的缩放工具调整文本大小，但是这实际上是对整个页面的调整，而不仅限于文本。

### 使用 em 来设置字体大小

如果要避免在 Internet Explorer 中无法调整文本的问题，许多开发者使用 em 单位代替 pixels。

W3C 推荐使用 em 尺寸单位。

1em 等于当前的字体尺寸。如果一个元素的 font-size 为 16 像素，那么对于该元素，1em 就等于 16 像素。在设置字体大小时，em 的值会相对于父元素的字体大小改变。

浏览器中默认的文本大小是 16 像素。因此 1em 的默认尺寸是 16 像素。

可以使用下面这个公式将像素转换为 em：*pixels*/16=*em*

（注：16 等于父元素的默认字体大小，假设父元素的 font-size 为 20px，那么公式需改为：*pixels*/20=*em*）

#### 实例

h1 {font-size:3.75em;} /\* 60px/16=3.75em \*/

h2 {font-size:2.5em;} /\* 40px/16=2.5em \*/

p {font-size:0.875em;} /\* 14px/16=0.875em \*/