* **#noteforCSS**

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

***selector {declaration1; declaration2; ... declarationN}***

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。

***selector {protery: value}***

下面这行代码的作用是将h1元素内的文字颜色定义为红色，同时将字体大小设置为14像素。

在这个例子中，h1是选择器，color和font-size是属性，red和14px是值。

***h1 {color:red; font-size:14px;}***

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。

如果值为若干单词，则要给值加引号：

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

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，这么做的好处是，当你从现有的规则中增减声明时，会尽可能地减少出错的可能性。就像这样：

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

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。

* 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;

}

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;

}

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。

* 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>

* 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>

id 属性只能在每个 HTML 文档中出现一次。

* CSS类选择器

在CSS中，类选择器以一个点号显示：

.center {text-again:center}

在上面的例子中，所有拥有 center 类的 HTML 元素均为居中。

**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）

<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）。

**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;

}

下面的例子为 title="W3School" 的所有元素设置样式：

[title=W3School]

{

border:5px solid blue;

}

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; }

<h1>可以应用样式：</h1>

<h2 title="hello world">Hello world</h2>

<p title="student hello">Hello W3School students!</h1>

<hr />

<h1>无法应用样式：</h1>

<h2 title="world">Hello world</h2>

<p title="student">Hello W3School students!</p>

**#注意：如果由空格分隔，那么放在前后均可**

下面的例子为带有包含指定值的 lang 属性的所有元素设置样式。适用于由连字符分隔的属性值：

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

<h1>可以应用样式：</h1>

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

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

<hr />

<h1>无法应用样式：</h1>

<p lang="bk-en">Hi!</p>

<p lang="zh">Hao!</p>

**#注意：如果由连字符分隔，那么必须在第一个**

**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;

}

## CSS 选择器参考手册

|  |  |
| --- | --- |
| **选择器** | **描述** |
| [[*attribute*]](https://www.w3school.com.cn/cssref/selector_attribute.asp) | 用于选取带有指定属性的元素。 |
| [[*attribute*=*value*]](https://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*]](https://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*]](https://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*]](https://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*]](https://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*]](https://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。 |

## 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。

### 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。

## 背景色

可以使用 [background-color 属性](https://www.w3school.com.cn/cssref/pr_background-color.asp)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;}

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](https://www.w3school.com.cn/cssref/pr_background-image.asp)。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。

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](https://www.w3school.com.cn/cssref/pr_background-repeat.asp)。

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;

}

## 背景定位

可以利用 [background-position 属性](https://www.w3school.com.cn/cssref/pr_background-position.asp)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，另一个对应垂直方向。

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。

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](https://www.w3school.com.cn/cssref/pr_background-attachment.asp)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](https://www.w3school.com.cn/tiy/t.asp?f=csse_background-attachment" \t "_blank)。

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。

## 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](https://www.w3school.com.cn/cssref/pr_text_text-indent.asp)，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>

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](https://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。

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

## 字间隔

[word-spacing 属性](https://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>

## 字母间隔

[letter-spacing 属性](https://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>

## 字符转换

[text-transform 属性](https://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](https://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;}

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](https://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 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](https://www.w3school.com.cn/tiy/t.asp?f=csse_text_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](https://www.w3school.com.cn/tiy/t.asp?f=csse_text_white-space)

### 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](https://www.w3school.com.cn/tiy/t.asp?f=csse_text_white-space_pre-wrap)

[实例 TIY ：white-space: pre-line](https://www.w3school.com.cn/tiy/t.asp?f=csse_text_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](https://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](https://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 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](https://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。

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](https://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](https://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;}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_font-variant)

## 字体加粗

[font-weight 属性](https://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;}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_font-weight)

## 字体大小

[font-size 属性](https://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 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 \*/

## 设置链接的样式

能够设置链接样式的 CSS 属性有很多种（例如 color, font-family, background 等等）。

链接的特殊性在于能够根据它们所处的状态来设置它们的样式。

链接的四种状态：

* a:link - 普通的、未被访问的链接
* a:visited - 用户已访问的链接
* a:hover - 鼠标指针位于链接的上方
* a:active - 链接被点击的时刻

### 实例

a:link {color:#FF0000;} /\* 未被访问的链接 \*/

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

a:hover {color:#FF00FF;} /\* 鼠标指针移动到链接上 \*/

a:active {color:#0000FF;} /\* 正在被点击的链接 \*/

当为链接的不同状态设置样式时，请按照以下次序规则：

* a:hover 必须位于 a:link 和 a:visited 之后
* a:active 必须位于 a:hover 之后

第一种情况：我定义的顺序是a:visited、a:hover、a:link，这时会发现：把鼠标放到未访问过的蓝色链接上时，它并不变成绿色，只有放在已访问的红色链接上，链接才会变绿。   
第二种情况：我把CSS定义顺序调整为：a:link、a:visited、a:hover，这时，无论你鼠标经过的链接有没有被访问过，它都会变成绿色啦。   
  
这是因为，一个鼠标经过的未访问链接同时拥有a:link、a:hover两种属性，在第一种情况下，a:link离它最近，所以它优先满足a:link，而放弃a:hover的重复定义。   
在第二种情况，无论链接有没有被访问过，它首先要检查是否符合a:hover的标准（即是否有鼠标经过它），满足，则变成绿色，不满足，则继续向上查找，一直找到满足条件的定义为止。   
一句话：在CSS中，如果对于相同元素有针对不同条件的定义，宜将最一般的条件放在最上面，并依次向下，保证最下面的是最特殊的条件。

**最后经验补充**：   
1.鼠标经过的“未访问链接”同时拥有a:link、a:hover两种属性，后面的属性会覆盖前面的属性定义；   
2.鼠标经过的“已访问链接”同时拥有a:visited、a:hover两种属性，后面的属性会覆盖前面的属性定义；   
所以说，a:hover定义一定要放在a:link、a:visited的后面

### 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](https://www.w3school.com.cn/cssref/pr_list-style-type.asp)：

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](https://www.w3school.com.cn/cssref/pr_list-style-image.asp) 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](https://www.w3school.com.cn/cssref/pr_list-style-position.asp) 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](https://www.w3school.com.cn/cssref/pr_list-style.asp)，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。

## 表格边框

如需在 CSS 中设置表格边框，请使用 border 属性。

下面的例子为 table、th 以及 td 设置了蓝色边框：

table, th, td

{

border: 1px solid blue;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_table_border)

请注意，上例中的表格具有双线条边框。这是由于 table、th 以及 td 元素都有独立的边框。

如果需要把表格显示为单线条边框，请使用 border-collapse 属性。

## 折叠边框

border-collapse 属性设置是否将表格边框折叠为单一边框：

table

{

border-collapse:collapse;

}

table,th, td

{

border: 1px solid black;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_table_border-collapse)

## 表格文本对齐

text-align 和 vertical-align 属性设置表格中文本的对齐方式。

text-align 属性设置水平对齐方式，比如左对齐、右对齐或者居中：

td

{

text-align:right;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_table_align)

vertical-align 属性设置垂直对齐方式，比如顶部对齐、底部对齐或居中对齐：

td

{

height:50px;

vertical-align:bottom;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_table_vertical-align)

## 表格内边距

如需控制表格中内容与边框的距离，请为 td 和 th 元素设置 padding 属性：

td

{

padding:15px;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_table_padding)

**注：会覆盖verticle-align，但不覆盖text-align.**

## 表格颜色

下面的例子设置边框的颜色，以及 th 元素的文本和背景颜色：

table, td, th

{

border:1px solid green;

}

th

{

background-color:green;

color:white; //注：文本颜色

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_table_color)

# CSS border-spacing 属性

[CSS 参考手册](https://www.w3school.com.cn/cssref/index.asp)

## 实例

为表格设置 border-spacing：

table

{

border-collapse:separate;

border-spacing:10px 50px;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_table_border-spacing)

该属性指定分隔边框模型中单元格边界之间的距离。在指定的两个长度值中，第一个是水平间隔，第二个是垂直间隔。除非 border-collapse 被设置为 separate，否则将忽略这个属性。尽管这个属性只应用于表，不过它可以由表中的所有元素继承。

|  |  |
| --- | --- |
| **默认值：** | *not specified* |
| **继承性：** | yes |
| **版本：** | CSS2 |
| **JavaScript 语法：** | *object*.style.borderSpacing="15px" |

## 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](https://www.w3school.com.cn/cssref/pr_outline.asp) | 在一个声明中设置所有的轮廓属性。 | 2 |
| [outline-color](https://www.w3school.com.cn/cssref/pr_outline-color.asp) | 设置轮廓的颜色。 | 2 |
| [outline-style](https://www.w3school.com.cn/cssref/pr_outline-style.asp) | 设置轮廓的样式。 | 2 |
| [outline-width](https://www.w3school.com.cn/cssref/pr_outline-width.asp) | 设置轮廓的宽度。 | 2 |

## 实例

设置 4 个边框的样式：

p

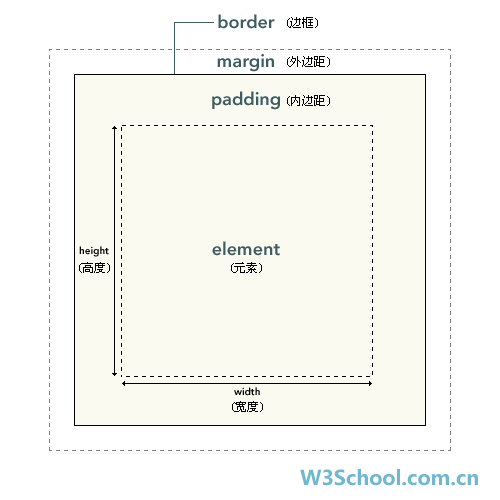
{

outline:#00FF00 dotted thick;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_outline)

## 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。

**提示：**背景应用于由内容和内边距、边框组成的区域。

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：

\* {

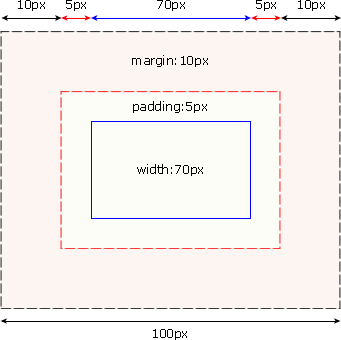
margin: 0;

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总尺寸。

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。

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 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%;} //顺时针

## 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](https://www.w3school.com.cn/cssref/pr_padding-top.asp)
* [padding-right](https://www.w3school.com.cn/cssref/pr_padding-right.asp)
* [padding-bottom](https://www.w3school.com.cn/cssref/pr_padding-bottom.asp)
* [padding-left](https://www.w3school.com.cn/cssref/pr_padding-left.asp)

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 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。

## 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](https://www.w3school.com.cn/cssref/pr_border-style.asp)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](https://www.w3school.com.cn/cssref/pr_border-top_style.asp)
* [border-right-style](https://www.w3school.com.cn/cssref/pr_border-right_style.asp)
* [border-bottom-style](https://www.w3school.com.cn/cssref/pr_border-bottom_style.asp)
* [border-left-style](https://www.w3school.com.cn/cssref/pr_border-left_style.asp)

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](https://www.w3school.com.cn/cssref/pr_border-width.asp)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;}

<b>注释：</b>"border-width" 属性如果单独使用的话是不会起作用的。请首先使用 "border-style" 属性来设置边框。

您也可以通过下列属性分别设置边框各边的宽度：

* [border-top-width](https://www.w3school.com.cn/cssref/pr_border-top_width.asp)
* [border-right-width](https://www.w3school.com.cn/cssref/pr_border-right_width.asp)
* [border-bottom-width](https://www.w3school.com.cn/cssref/pr_border-bottom_width.asp)
* [border-left-width](https://www.w3school.com.cn/cssref/pr_border-left_width.asp)

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](https://www.w3school.com.cn/cssref/pr_border-color.asp)，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;

}

**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](https://www.w3school.com.cn/cssref/pr_border-top_color.asp)
* [border-right-color](https://www.w3school.com.cn/cssref/pr_border-right_color.asp)
* [border-bottom-color](https://www.w3school.com.cn/cssref/pr_border-bottom_color.asp)
* [border-left-color](https://www.w3school.com.cn/cssref/pr_border-left_color.asp)

[所有边框属性在一个声明之中](https://www.w3school.com.cn/tiy/t.asp?f=csse_border)

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。

p

{

border: medium double rgb(250,0,255)

}

|  |  |
| --- | --- |
| [border](https://www.w3school.com.cn/cssref/pr_border.asp) | 简写属性，用于把针对四个边的属性设置在一个声明。 |

|  |  |
| --- | --- |
| [border-bottom](https://www.w3school.com.cn/cssref/pr_border-bottom.asp) | 简写属性，用于把下边框的所有属性设置到一个声明中。 |

## CSS margin 属性

设置外边距的最简单的方法就是使用 [margin 属性](https://www.w3school.com.cn/cssref/pr_margin.asp)。

margin 属性接受任何长度单位，可以是像素、英寸、毫米或 em。

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。

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。

## 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：

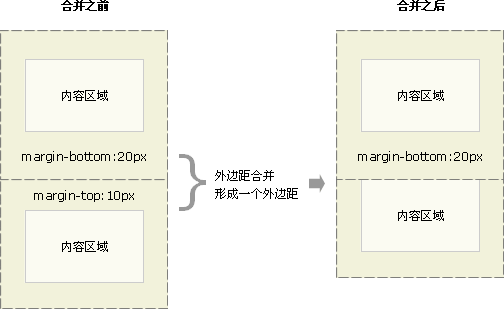
* [margin-top](https://www.w3school.com.cn/cssref/pr_margin-top.asp)
* [margin-right](https://www.w3school.com.cn/cssref/pr_margin-right.asp)
* [margin-bottom](https://www.w3school.com.cn/cssref/pr_margin-bottom.asp)
* [margin-left](https://www.w3school.com.cn/cssref/pr_margin-left.asp)

## 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。

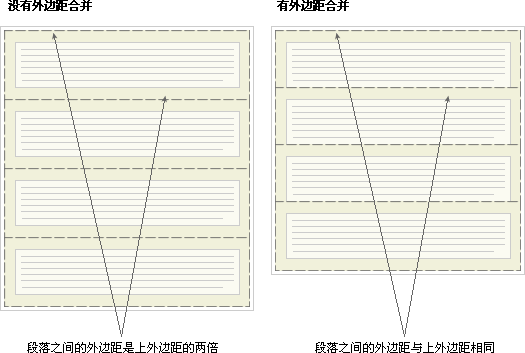
当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。请看下图：



[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_margin_collapsing1)

//注意：会被***border***改变他的行为 ***margin包括边框***。

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。

## 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](https://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。

## 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。

## CSS position 属性

通过使用 [position 属性](https://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。

通过绝对定位，元素可以放置到页面上的任何位置。 //absolute

相对定位会按照元素的原始位置对该元素进行移动 //relative

div

{

background-color:#00FFFF;

width:150px;

height:150px;

overflow: scroll;

}

</style>

</head>

<body>

<p>如果元素中的内容超出了给定的宽度和高度属性，overflow 属性可以确定是否显示滚动条等行为。</p>

这个属性定义溢出元素内容区的内容会如何处理。如果值为 scroll，不论是否需要，用户代理都会提供一种滚动机制。因此，有可能即使元素框中可以放下所有内容也会出现滚动条。默认值是 visible。

//overflow：hidden 可以隐藏超出的部分

//overflow: auto浏览器来自动地处理溢出

clip:rect(0px 50px 200px 0px)

clip 属性剪切了一幅图像

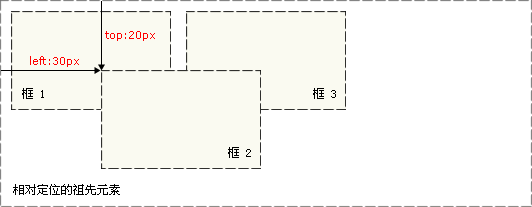
Z-index可被用于将一个元素放置于另一元素之后。

默认的 z-index 是 0。Z-index -1 拥有更低的优先级。  
设置元素的堆叠顺序。

z-index 属性设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。

**注释：**元素可拥有负的 z-index 属性值。

**注释：**Z-index 仅能在定位元素上奏效（例如 position:absolute;）！



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。

**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](https://www.w3school.com.cn/cssref/pr_pos_z-index.asp)来控制这些框的堆放次序。

## 实例

把图像向右浮动：

img

{

float:right;

}

float 属性定义元素在哪个方向浮动。以往这个属性总应用于图像，使文本围绕在图像周围，不过在 CSS 中，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。

如果浮动非替换元素，则要指定一个明确的宽度；否则，它们会尽可能地窄。

**注释：**假如在一行之上只有极少的空间可供浮动元素，那么这个元素会跳至下一行，这个过程会持续到某一行拥有足够的空间为止。

## 可能的值

|  |  |
| --- | --- |
| **值** | **描述** |
| left | 元素向左浮动。 |
| right | 元素向右浮动。 |
| none | 默认值。元素不浮动，并会显示在其在文本中出现的位置。 |
| inherit | 规定应该从父元素继承 float 属性的值。 |

div.left

{

float:left;

width:160px;

margin:0;

padding:1em;

}

div.content

{

margin-left:190px;

border-left:1px solid gray;

padding:1em;

}

要想阻止行框围绕浮动框，需要对该框应用 [clear 属性](https://www.w3school.com.cn/cssref/pr_class_clear.asp)。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。

Clear:等同取消了float的功能

## 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;}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_type_1)

## 类型选择器

在 W3C 标准中，元素选择器又称为类型选择器（type selector）。

“类型选择器匹配文档语言元素类型的名称。类型选择器匹配文档树中该元素类型的每一个实例。”

下面的规则匹配文档树中所有 h1 元素：

h1 {font-family: sans-serif;}

## 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;}

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。

可以将任意多个选择器分组在一起，对此没有任何限制。

## 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;}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_grouping_selector_universal)

这个声明等价于列出了文档中所有元素的一个分组选择器。利用通配选择器，只需敲一次键（仅一个星号）就能使文档中所有元素的 color 属性值指定为 red。

## 声明分组

我们既可以对选择器进行分组，也可以对声明分组。

不过，如果忽略了第二个分号，用户代理就会把这个样式表解释如下：

h1 {

font: 28px Verdana;

color: blue background: red;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_grouping_declaration_error)

因为 background 对 color 来说不是一个合法值，而且由于只能为 color 指定一个关键字，所以用户代理会完全忽略这个 color 声明（包括 background: black 部分）。这样 h1 标题只会显示为蓝色，而没有红色背景，不过更有可能根本得不到蓝色的 h1。相反，这些标题只会显示为默认颜色（通常是黑色），而且根本没有背景色。font: 28px Verdana 声明仍能正常发挥作用，因为它确实正确地以一个分号结尾。

与选择器分组一样，声明分组也是一种便利的方法，可以缩短样式表，使之更清晰，也更易维护。

**提示：**在规则的最后一个声明后也加上分号是一个好习惯。在向规则增加另一个声明时，就不必担心忘记再插入一个分号。

### 语法

然后我们使用以下语法向这些归类的元素应用样式，即类名前有一个点号（.），然后结合通配选择器：

\*.important {color:red;}

如果您只想选择所有类名相同的元素，可以在类选择器中忽略通配选择器，这没有任何不好的影响：

.important {color:red;}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_class_1)

## CSS 多类选择器

在上一节中，我们处理了 class 值中包含一个词的情况。在 HTML 中，一个 class 值中可能包含一个词列表，各个词之间用空格分隔。例如，如果希望将一个特定的元素同时标记为重要（important）和警告（warning），就可以写作：

<p class="important warning">

This paragraph is a very important warning.

</p>

这两个词的顺序无关紧要，写成 warning important 也可以。

我们假设 class 为 important 的所有元素都是粗体，而 class 为 warning 的所有元素为斜体，class 中同时包含 important 和 warning 的所有元素还有一个银色的背景 。就可以写作：

.important {font-weight:bold;}

.warning {font-style:italic;}

.important.warning {background:silver;}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_class_multiple)

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）。

***//注意：应用第三条时,同时应用前面的一二条规则，没有覆盖***

## 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。

## 区分大小写

请注意，类选择器和 ID 选择器可能是区分大小写的。这取决于文档的语言。HTML 和 XHTML 将类和 ID 值定义为区分大小写，所以类和 ID 值的大小写必须与文档中的相应值匹配。

因此，对于以下的 CSS 和 HTML，元素不会变成粗体：

#intro {font-weight:bold;}

<p id="Intro">This is a paragraph of introduction.</p>

由于字母 i 的大小写不同，所以选择器不会匹配上面的元素。

## 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

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](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_attribute_att)

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](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_attribute_att_3)

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](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_attribute_value_1)

### 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](https://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](https://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](https://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 |

## 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](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_attribute_valuelist_hyphen_1)

注意：为什么包括en-

## 定义和用法

[*attribute*|=*value*] 选择器用于选取带有以指定值开头的属性值的元素。

**注释：**该值必须是整个单词，比如 lang="en"，或者后面跟着连字符，比如 lang="en-us"。

## CSS 选择器参考手册

|  |  |
| --- | --- |
| **选择器** | **描述** |
| [[*attribute*]](https://www.w3school.com.cn/cssref/selector_attribute.asp) | 用于选取带有指定属性的元素。 |
| [[*attribute*=*value*]](https://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*]](https://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*]](https://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*]](https://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*]](https://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*]](https://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。 |

## 定义和用法

<!DOCTYPE> 声明必须是 HTML 文档的第一行，位于 <html> 标签之前。

<!DOCTYPE> 声明不是 HTML 标签；***它是指示 web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。***

在 HTML 4.01 中，<!DOCTYPE> 声明引用 DTD，因为 HTML 4.01 基于 SGML。DTD 规定了标记语言的规则，这样浏览器才能正确地呈现内容。

HTML5 不基于 SGML，所以不需要引用 DTD。

**提示：**请始终向 HTML 文档添加 <!DOCTYPE> 声明，这样浏览器才能获知文档类型。

## HTML 4.01 与 HTML5 之间的差异

在 HTML 4.01 中有三种 <!DOCTYPE> 声明。在 HTML5 中只有一种：

<!DOCTYPE html>

## 提示和注释

**注释：**<!DOCTYPE> 声明没有结束标签。

**提示：**<!DOCTYPE> 声明对大小写不敏感。

**提示：**请使用 W3C 的验证器来检查您是否编写了有效的 HTML / XHTML 文档！

**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;}

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](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_descendant_1)

## 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”。

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>

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_descendant)

**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：

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

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

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_child)

## 语法解释

您应该已经注意到了，子选择器使用了大于号（子结合符）。

子结合符两边可以有空白符，这是可选的。因此，以下写法都没有问题：

h1 > strong

h1> strong

h1 >strong

h1>strong

如果从右向左读，选择器 h1 > strong 可以解释为“选择作为 h1 元素子元素的所有 strong 元素”。

## 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。

**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;}

这个选择器读作：“选择紧接在 h1 元素后出现的段落，h1 和 p 元素拥有共同的父元素”。

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_adjacent_sibling)

***注意：只选择到一个具体的元素。***

**语法解释**

相邻兄弟选择器使用了加号（+），即相邻兄弟结合符（Adjacent sibling combinator）。

**注释：**与子结合符一样，相邻兄弟结合符旁边可以有空白符。

请看下面这个文档树片段：

<div>

<ul>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

</ul>

<ol>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

</ol>

</div>

在上面的片段中，div 元素中包含两个列表：一个无序列表，一个有序列表，每个列表都包含三个列表项。这两个列表是相邻兄弟，列表项本身也是相邻兄弟。不过，第一个列表中的列表项与第二个列表中的列表项不是相邻兄弟，因为这两组列表项不属于同一父元素（最多只能算堂兄弟）。

请记住，用一个结合符只能选择两个相邻兄弟中的第二个元素。请看下面的选择器：

li + li {font-weight:bold;}

上面这个选择器只会把列表中的第二个和第三个列表项变为粗体。第一个列表项不受影响。

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_selector_adjacent_sibling_2)

***注意：只选择到一个具体的元素。***

## 结合其他选择器

相邻兄弟结合符还可以结合其他结合符：

html > body table + ul {margin-top:20px;}

这个选择器解释为：选择紧接在 table 元素后出现的所有兄弟 ul 元素，该 table 元素包含在一个 body 元素中，body 元素本身是 html 元素的子元素。

[:first-child（首个子对象）](https://www.w3school.com.cn/tiy/t.asp?f=csse_first-child)

本例演示 :first-child 伪类的用法。

<style type="text/css">

q:lang(no)

{

quotes: "~" "~"

}

</style>

</head>

<body>

<p>:lang 伪类允许您为不同的语言定义特殊的规则。在下面的例子中，在下面的例子中，:lang 类为带有值为 "no" 的 lang 属性的 q 元素定义引号的类型：</p>

<p>一些文本 <q lang="no">段落中的引用</q> 一些文本。</p>

</body>

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 \*/

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_link)

**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。

## 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](https://www.w3school.com.cn/tiy/t.asp?f=csse_first-child)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>](https://www.w3school.com.cn/tags/tag_doctype.asp)，这样 :first-child 才能在 IE 中生效。

### 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](https://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 {

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

# CSS 伪元素 (Pseudo-elements)

**CSS 伪元素用于向某些选择器设置特殊效果。**

## 语法

伪元素的语法：

selector:pseudo-element {property:value;}

CSS 类也可以与伪元素配合使用：

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

## :first-line 伪元素

"first-line" 伪元素用于向文本的首行设置特殊样式。

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

### 实例

p:first-line

{

color:#ff0000;

font-variant:small-caps;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_firstline)

**注释：**"first-line" 伪元素只能用于块级元素。

**注释：**下面的属性可应用于 "first-line" 伪元素：

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

## :first-letter 伪元素

"first-letter" 伪元素用于向文本的首字母设置特殊样式：

p:first-letter

{

color:#ff0000;

font-size:xx-large;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_firstletter)

**注释：**"first-letter" 伪元素只能用于块级元素。

## 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">This is a paragraph in an article。</p>

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

## 多重伪元素

可以结合多个伪元素来使用。

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

p:first-letter

{

color:#ff0000;

font-size:xx-large;

}

p:first-line

{

color:#0000ff;

font-variant:small-caps;

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_firstletter_firstline)

## CSS2 - :before 伪元素

":before" 伪元素可以在元素的内容前面插入新内容。

下面的例子在每个 <h1> 元素前面插入一幅图片：

h1:before

{

content:url(logo.gif);

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_before)

## CSS2 - :after 伪元素

":after" 伪元素可以在元素的内容之后插入新内容。

下面的例子在每个 <h1> 元素后面插入一幅图片：

h1:after

{

content:url(logo.gif);

}

[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_after)