**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）

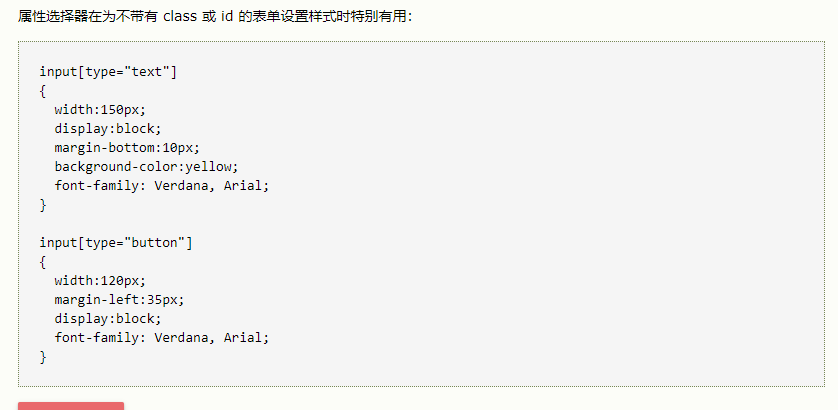


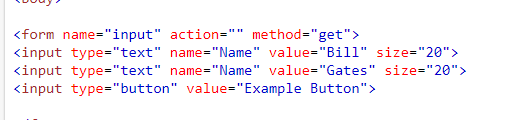


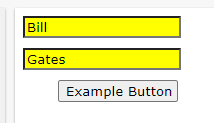


1号用于包含空格的，比如hello world可以触发规则

2.号用于给连接符，比如en-american之类的

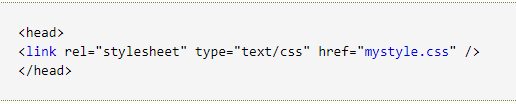




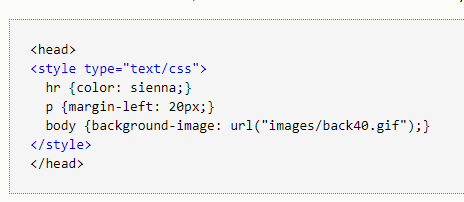


外部样式表：

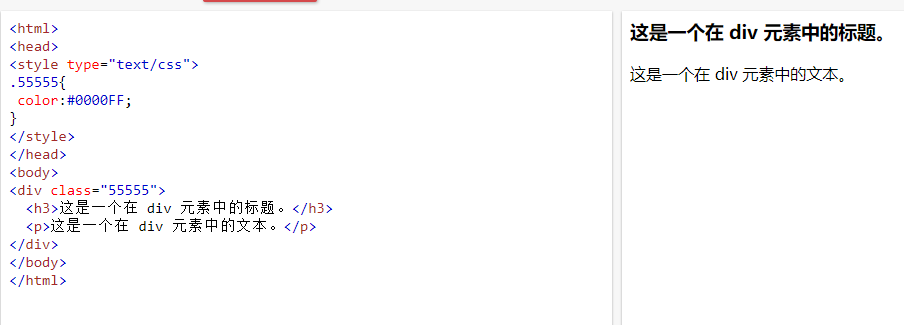
用link

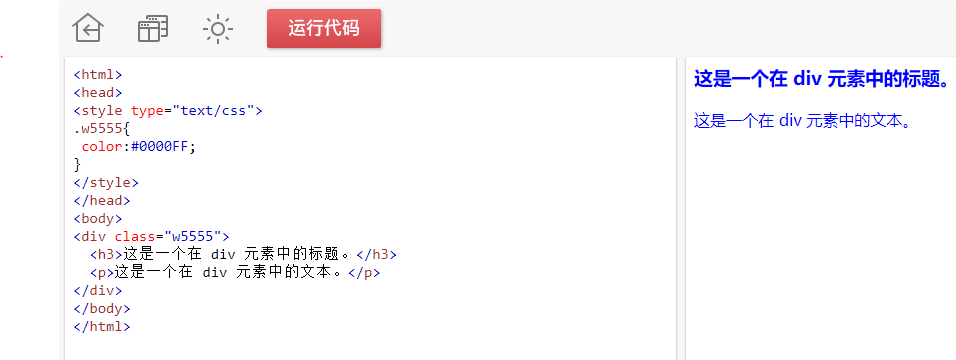


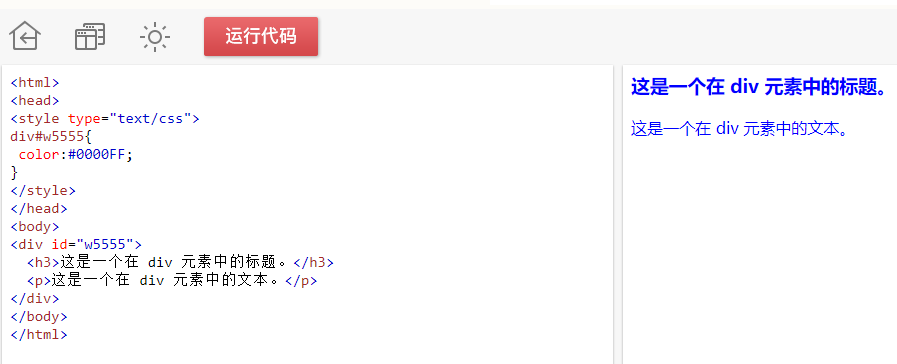
内部样式表：



关于内部样式的几种使用（包括正确和错误方法，外部样式也可参考）

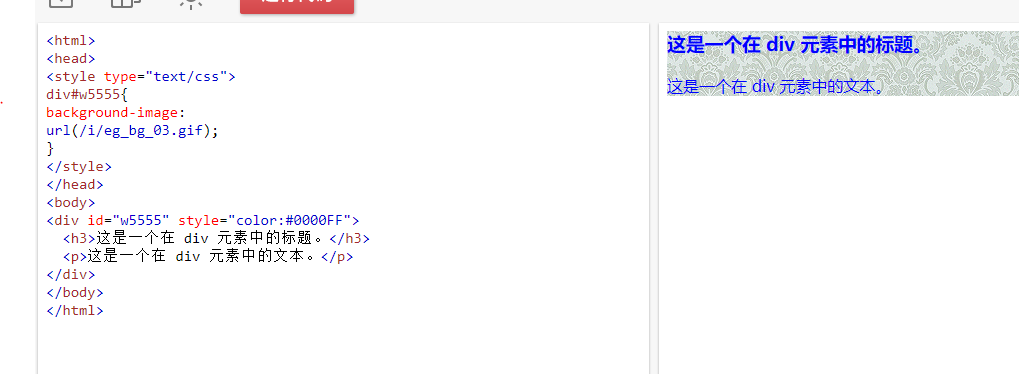






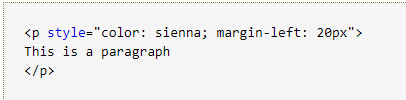
Tips：id和class的首字母必须不能是数字，否则不能生效

注意class和id使用方式的异同



<div id="w5555" style="color:#0000FF">在一个<>内同时用id 和style的顺序问题

内联样式：



多重样式：

若外部样式和内部样式分别定义了同一个属性，

那么重叠部分（比如都对font-size进行了设置）就依据内部样式，而内部没定义，外部有定义的，依据外部样式表

CSS背景：

body

{

background-image:

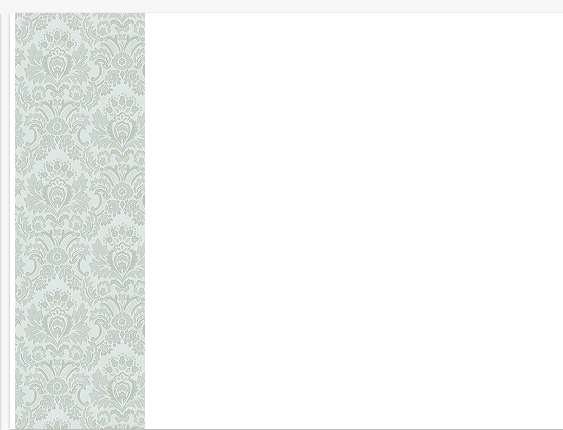
url(/i/eg\_bg\_03.gif);

background-position:1% 1%;

background-repeat: repeat-y

}

效果：：：



如果先background-repeat: repeat-y，再background-position:1% 1%;

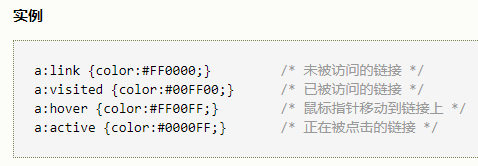
则：

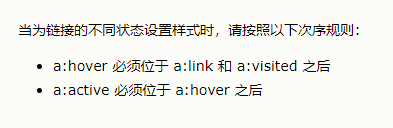


background-position:50px 100px;

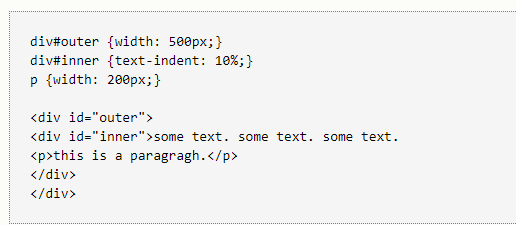
此为设置像素位置

Css链接：





CSS文本：



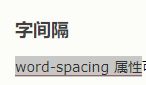
Special：注意div#的用法

**justify**

用法：

<https://www.cnblogs.com/zjjDaily/p/9525350.html>

关于间隔





关于字符转换：

Text-transform四个值

None：不变

Uppercase：大写

Lowercase：小写

Capitalize：首字母大写

a {text-decoration: none;}

文本装饰：

还可以text-decoration 有 5 个值：

* None//不变
* Underline//下方画线
* Overline//上方画线
* line-through//中间画线
* blink//使得文本闪烁

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：

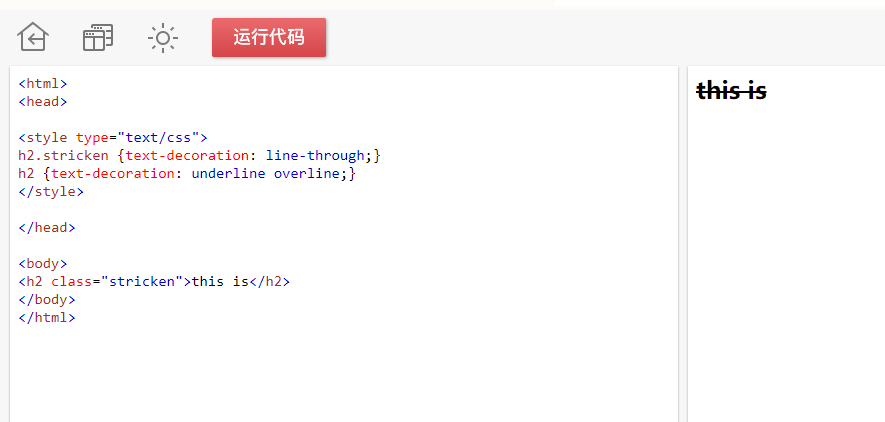
a:link a:visited {text-decoration: underline overline;}

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**。





**text-decoration 值会替换而不是累积起来**。啥意思？？？

**可能的值**

|  |  |
| --- | --- |
| **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。 |

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;}

不懂？？？？

**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>

[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;}

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。

更多详细设定：

<https://www.w3school.com.cn/css/css_font.asp>

css列表：

**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}

css表格：

<https://www.w3school.com.cn/css/css_table.asp>

css:框架模型（啥呀）

但是，许多元素将由用户代理样式表设置外边距和内边距。？？？（啥意思）

<https://www.w3school.com.cn/css/css_boxmodel.asp>不懂

css内边距：

<https://www.w3school.com.cn/css/css_padding.asp>不懂

以后看

CSS内边距：

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%;}

**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>

**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%：

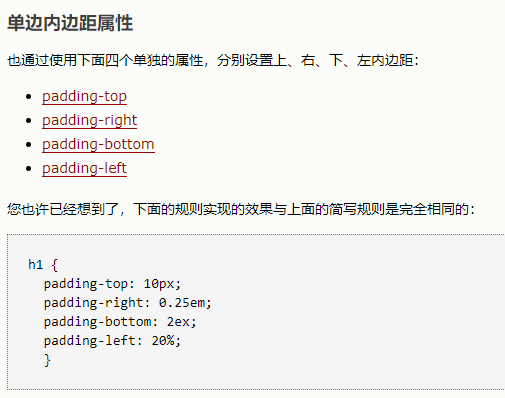
p {padding: 10%;}

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>



实例：



什么时候用margin什么时候用padding？

用padding会使element在框内移动移动（若有空间），而使用margin不会（若有空间）

CSS边框：

**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;}

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)

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**

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

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;}

**边框的颜色**

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。

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;}

**CSS 外边距**

**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：

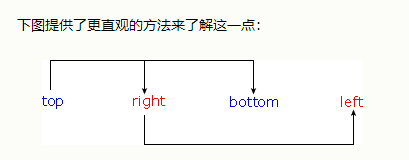
p {margin: 0.5em 1em 0.5em 1em;}

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。



**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)

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;

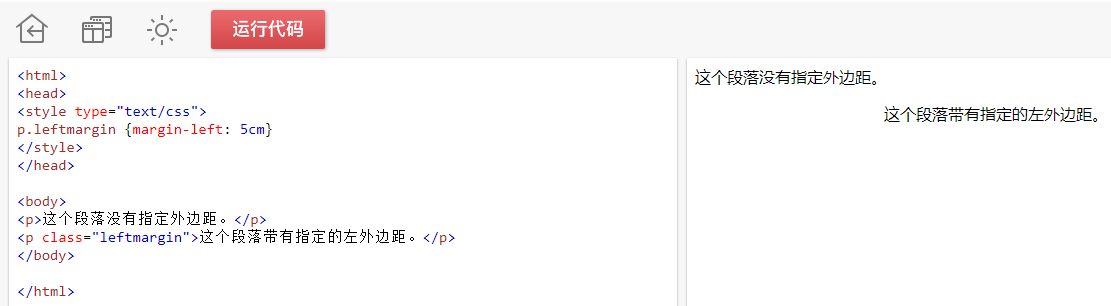
}

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。

实例：





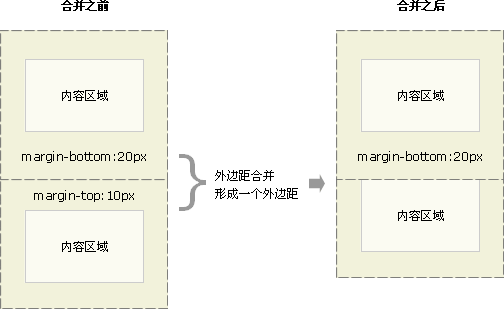
**CSS 外边距合并**

<https://www.w3school.com.cn/css/css_margin_collapsing.asp>

**外边距合并**

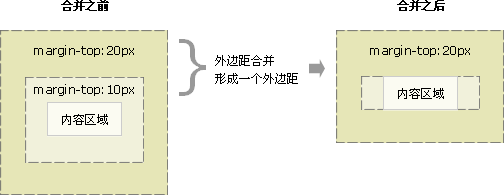
外边距合并（叠加）是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。

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)

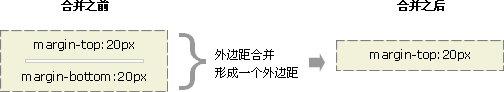
当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并。请看下图：



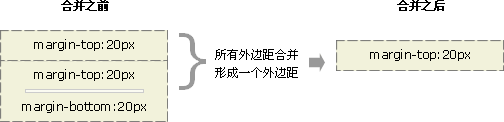
[亲自试一试](https://www.w3school.com.cn/tiy/t.asp?f=csse_margin_collapsing2)

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：

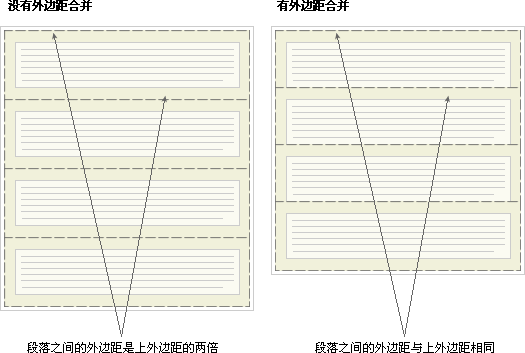


如果这个外边距遇到另一个元素的外边距，它还会发生合并：



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。



所有元素会被合并成一个吗？？

CSS定位概述：

**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>

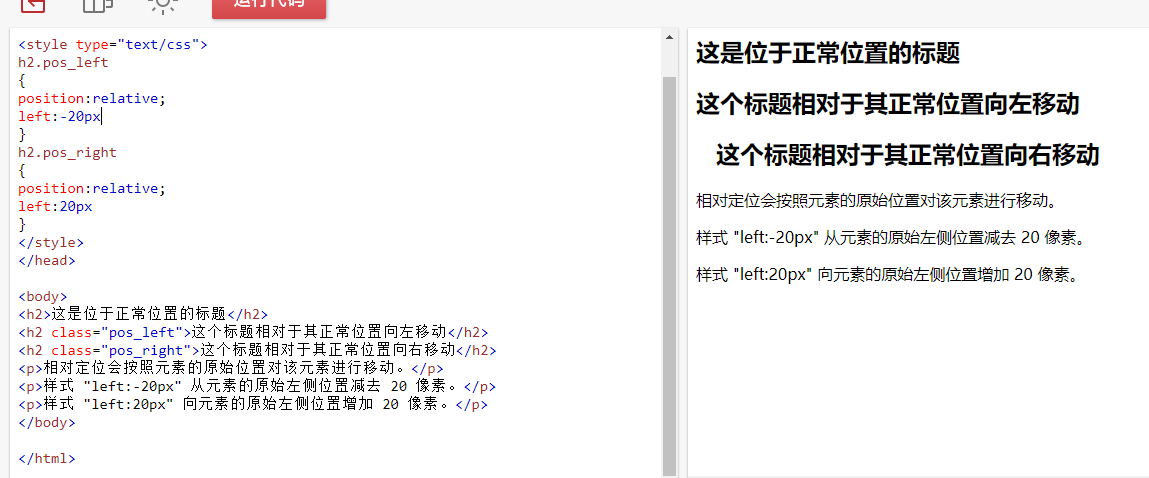
</div>

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。

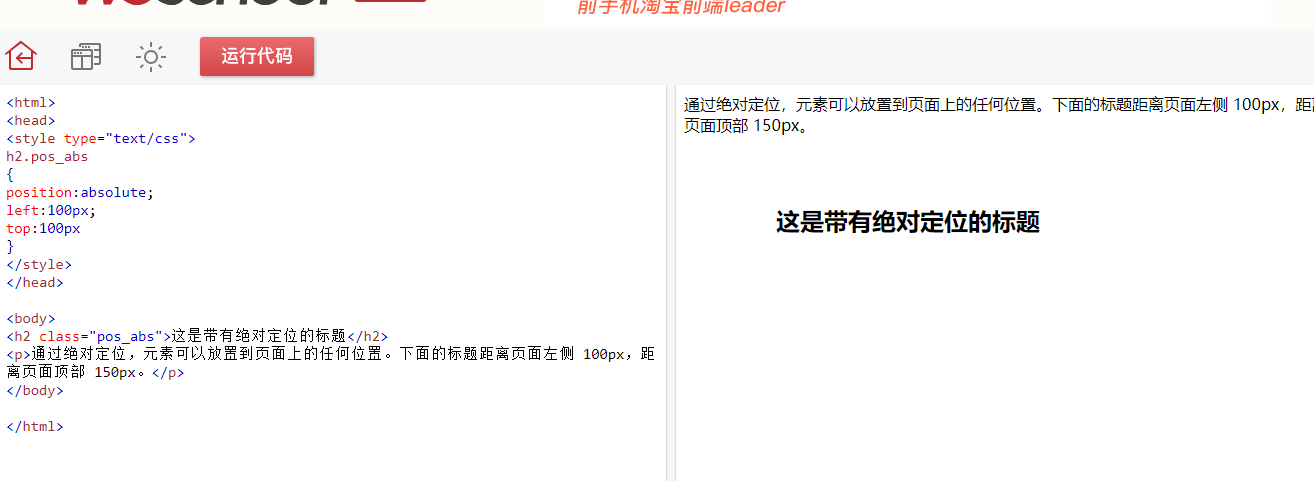
块级元素的文本行也会发生类似的情况。假设有一个包含三行文本的段落。每行文本形成一个无名框。无法直接对无名块或行框应用样式，因为没有可以应用样式的地方（注意，行框和行内框是两个概念）。但是，这有助于理解在屏幕上看到的所有东西都形成某种框。

？？？

CSS相对定位（相对于原来位置）



CSS绝对定位（相对左边框）

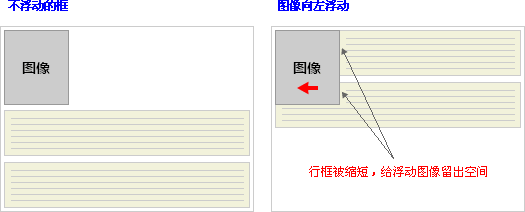


Css框

**行框和清理**

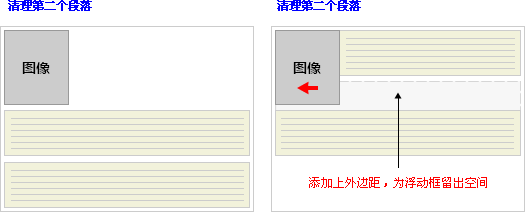
浮动框旁边的行框被缩短，从而给浮动框留出空间，行框围绕浮动框。

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](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。

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">

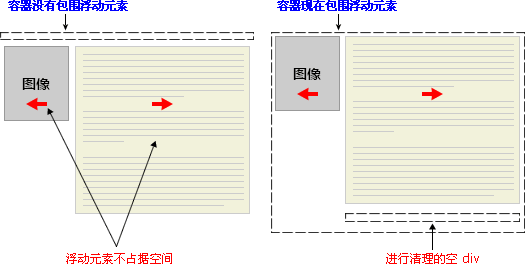
<img src="news-pic.jpg" />

<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](https://www.w3school.com.cn/cssref/pr_class_clear.asp)。

？？？

啥东西？？？

**CSS 选择器**

选择器分组：

**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;}

选择器分组：

注意，对声明分组，一定要在各个声明的最后使用分号，这很重要。浏览器会忽略样式表中的空白符。只要加了分号，就可以毫无顾忌地采用以下格式建立样式：

h1 {

font: 28px Verdana;

color: blue;

background: red;

}

代码：

Css类选择器：

<html>

<head>

<style type="text/css">

\*.important {color:red;}

</style>

</head>

<body>

<h1 class="a important">This heading is very important.</h1>

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

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

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

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

<p>...</p>

</body>

</html>

结果

**This heading is very important.**

This paragraph is very important.

This is a paragraph.

This is a paragraph.

This is a paragraph.

...

Css类选择器

\*.important {color:red;}

前加\*指的是默认带有important+空格也可以被识别。

代码：

<html>

<head>

<style type="text/css">

.important {font-weight:bold;}

.warning {font-style:italic;}

.important.warning {background:silver;}

</style>

</head>

<body>

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

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

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

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

<p>...</p>

</body>

</html>

结果：

**This paragraph is very important.**

*This is a warning.*

***This paragraph is a very important warning.***

This is a paragraph.

...

Css id

**类选择器还是 ID 选择器？**

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

**区别 1：只能在文档中使用一次**

与类不同，在一个 HTML 文档中，ID 选择器会使用一次，而且仅一次。

**区别 2：不能使用 ID 词列表**

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

**区别 3：ID 能包含更多含义**

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

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

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

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

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

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

**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。

Css id：

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

<https://www.w3school.com.cn/css/css_selector_attribute.asp>

css：

## 后代选择器：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>

Css子元素选择器：

**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>



## Css相邻兄弟选择器： 选择相邻兄弟

**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”。

代码：

<!DOCTYPE HTML>

<html>

<head>

<style type="text/css">

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

</style>

</head>

<body>

<h1>This is a heading.</h1>

<p>This is paragraph.</p>

<p>This is paragraph.</p>

<p>This is paragraph.</p>

<p>This is paragraph.</p>

<p>This is paragraph.</p>

</body>

</html>

结果：

**This is a heading.**

This is paragraph.

This is paragraph.

This is paragraph.

This is paragraph.

This is paragraph.

Css相邻兄弟选择器：

<https://www.w3school.com.cn/css/css_selector_adjacent_sibling.asp>

伪类？？？伪元素？？？