# 第1章 CSS 教程

CSS 是一种描述 HTML 文档样式的语言。

CSS 描述应该如何显示 HTML 元素。

本教程将从零起点的基础教程开始，一直到 CSS3 高级教程，为您提供全面系统地讲解。

每一章中的实例

本 CSS 教程包含数百个 CSS 实例。

通过使用我们的在线编辑器（W3Schoo TIY），您可以编辑 CSS，然后单击运行按钮来查看结果。

CSS 实例

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: lightblue;

}

h1 {

color: white;

text-align: center;

}

p {

font-family: verdana;

font-size: 20px;

}

</style>

</head>

<body>

<h1>我的第一个 CSS 实例</h1>

<p>这是一个段落。</p>

</body>

</html>

# 第2章 CSS 简介

## 2.1 什么是 CSS？

CSS 指的是层叠样式表\* (Cascading Style Sheets)

CSS 描述了如何在屏幕、纸张或其他媒体上显示 HTML 元素

CSS 节省了大量工作。它可以同时控制多张网页的布局

外部样式表存储在 CSS 文件中

\*：也称级联样式表。

## 2.2 为何使用 CSS？

CSS 用于定义网页的样式，包括针对不同设备和屏幕尺寸的设计和布局。

body {

background-color: lightblue;

}

h1 {

color: white;

text-align: center;

}

p {

font-family: verdana;

font-size: 20px;

}

## 2.3 CSS 解决了一个大问题

HTML 从未打算包含用于格式化网页的标签！

创建 HTML 的目的是描述网页的内容，例如：

<h1>这是一个标题。</h1>

<p>这是一个段落。</p>

将 <font> 之类的标签和 color 属性添加到 HTML 3.2 规范后，Web 开发人员的噩梦开始了。大型网站的开发将字体和颜色信息添加到每个页面中，这演变为一个漫长而昂贵的过程。

为了解决这个问题，万维网联盟（W3C）创建了 CSS。

CSS 从 HTML 页面中删除了样式格式！

如果您不知道 HTML 是什么，建议您阅读 HTML 教程。

## 2.4 CSS 节省了大量工作！

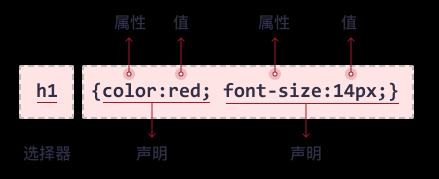
样式定义通常保存在外部 .css 文件中。

通过使用外部样式表文件，您只需更改一个文件即可更改整个网站的外观！

# 第3章 CSS 语法

## 3.1 CSS 语法

CSS 规则集（rule-set）由选择器和声明块组成：



选择器指向您需要设置样式的 HTML 元素。

声明块包含一条或多条用分号分隔的声明。

每条声明都包含一个 CSS 属性名称和一个值，以冒号分隔。

多条 CSS 声明用分号分隔，声明块用花括号括起来。

实例

在此例中，所有 <p> 元素都将居中对齐，并带有红色文本颜色：

p {

color: red;

text-align: center;

}

## 3.2 例子解释

p 是 CSS 中的选择器（它指向要设置样式的 HTML 元素：<p>）。

color 是属性，red 是属性值

text-align 是属性，center 是属性值

在下一章中，您将学到更多关于 CSS 选择器和 CSS 属性的知识。

# 第4章 CSS 选择器

CSS 选择器用于“查找”（或选取）要设置样式的 HTML 元素。

我们可以将 CSS 选择器分为五类：

1.简单选择器（根据名称、id、类来选取元素）

2.组合器选择器（根据它们之间的特定关系来选取元素）

3.伪类选择器（根据特定状态选取元素）

4.伪元素选择器（选取元素的一部分并设置其样式）

5.属性选择器（根据属性或属性值来选取元素）

此页会讲解最基本的 CSS 选择器。

## 4.1 CSS 元素选择器

元素选择器根据元素名称来选择 HTML 元素。

实例

在这里，页面上的所有 <p> 元素都将居中对齐，并带有红色文本颜色：

p {

text-align: center;

color: red;

}

## 4.2 CSS id 选择器

id 选择器使用 HTML 元素的 id 属性来选择特定元素。

元素的 id 在页面中是唯一的，因此 id 选择器用于选择一个唯一的元素！

要选择具有特定 id 的元素，请写一个井号（＃），后跟该元素的 id。

实例

这条 CSS 规则将应用于 id="para1" 的 HTML 元素：

#para1 {

text-align: center;

color: red;

}

注意：id 名称不能以数字开头。

## 4.3 CSS 类选择器

类选择器选择有特定 class 属性的 HTML 元素。

如需选择拥有特定 class 的元素，请写一个句点（.）字符，后面跟类名。

实例

在此例中，所有带有 class="center" 的 HTML 元素将为红色且居中对齐：

.center {

text-align: center;

color: red;

}

您还可以指定只有特定的 HTML 元素会受类的影响。

实例

在这个例子中，只有具有 class="center" 的 <p> 元素会居中对齐：

p.center {

text-align: center;

color: red;

}

HTML 元素也可以引用多个类。

实例

在这个例子中，<p> 元素将根据 class="center" 和 class="large" 进行样式设置：

<p class="center large">这个段落引用两个类。</p>

注意：类名不能以数字开头！

## 4.4 CSS 通用选择器

通用选择器（\*）选择页面上的所有的 HTML 元素。

实例

下面的 CSS 规则会影响页面上的每个 HTML 元素：

\* {

text-align: center;

color: blue;

}

## 4.5 CSS 分组选择器

分组选择器选取所有具有相同样式定义的 HTML 元素。

请看下面的 CSS 代码（h1、h2 和 p 元素具有相同的样式定义）：

h1 {

text-align: center;

color: red;

}

h2 {

text-align: center;

color: red;

}

p {

text-align: center;

color: red;

}

最好对选择器进行分组，以最大程度地缩减代码。

如需对选择器进行分组，请用逗号来分隔每个选择器。

实例

在这个例子中，我们对上述代码中的选择器进行分组：

h1, h2, p {

text-align: center;

color: red;

}

## 4.6 所有简单的 CSS 选择器

|  |  |  |
| --- | --- | --- |
| 选择器 | 例子 | 例子描述 |
| .class | .intro | 选取所有 class="intro" 的元素。 |
| #id | #firstname | 选取 id="firstname" 的那个元素。 |
| \* | \* | 选取所有元素。 |
| element | p | 选取所有 <p> 元素。 |
| element,element,.. | div, p | 选取所有 <div> 元素和所有 <p> 元素。 |

# 第5章 如何添加 CSS

当浏览器读到样式表时，它将根据样式表中的信息来格式化 HTML 文档。

## 5.1 三种使用 CSS 的方法

有三种插入样式表的方法：

外部 CSS

内部 CSS

行内 CSS

## 5.2 外部 CSS

通过使用外部样式表，您只需修改一个文件即可改变整个网站的外观！

每张 HTML 页面必须在 head 部分的 <link> 元素内包含对外部样式表文件的引用。

实例

外部样式在 HTML 页面 <head> 部分内的 <link> 元素中进行定义：

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

外部样式表可以在任何文本编辑器中编写，并且必须以 .css 扩展名保存。

外部 .css 文件不应包含任何 HTML 标签。

"mystyle.css" 是这样的：

"mystyle.css"

body {

background-color: lightblue;

}

h1 {

color: navy;

margin-left: 20px;

}

注意：请勿在属性值和单位之间添加空格（例如 margin-left: 20 px;）。正确的写法是：margin-left: 20px;

## 5.3 内部 CSS

如果一张 HTML 页面拥有唯一的样式，那么可以使用内部样式表。

内部样式是在 head 部分的 <style> 元素中进行定义。

实例

内部样式在 HTML 页面的 <head> 部分内的 <style> 元素中进行定义：

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: linen;

}

h1 {

color: maroon;

margin-left: 40px;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

## 5.4 行内 CSS

行内样式（也称内联样式）可用于为单个元素应用唯一的样式。

如需使用行内样式，请将 style 属性添加到相关元素。style 属性可包含任何 CSS 属性。

实例

行内样式在相关元素的 "style" 属性中定义：

<!DOCTYPE html>

<html>

<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>

<p style="color:red;">This is a paragraph.</p>

</body>

</html>

提示：行内样式失去了样式表的许多优点（通过将内容与呈现混合在一起）。请谨慎使用此方法。

## 5.5 多个样式表

如果在不同样式表中为同一选择器（元素）定义了一些属性，则将使用最后读取的样式表中的值。

假设某个外部样式表为 <h1> 元素设定的如下样式：

h1 {

color: navy;

}

然后，假设某个内部样式表也为 <h1> 元素设置了如下样式：

h1 {

color: orange;

}

实例

如果内部样式是在链接到外部样式表之后定义的，则 <h1> 元素将是橙色：

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

<style>

h1 {

color: orange;

}

</style>

</head>

实例

不过，如果在链接到外部样式表之前定义了内部样式，则 <h1> 元素将是深蓝色：

<head>

<style>

h1 {

color: orange;

}

</style>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

## 5.6 层叠顺序

当为某个 HTML 元素指定了多个样式时，会使用哪种样式呢？

页面中的所有样式将按照以下规则“层叠”为新的“虚拟”样式表，其中第一优先级最高：

行内样式（在 HTML 元素中）

外部和内部样式表（在 head 部分）

浏览器默认样式

因此，行内样式具有最高优先级，并且将覆盖外部和内部样式以及浏览器默认样式。

# 第6章 CSS 注释

## 6.1 CSS 注释

注释用于解释代码，以后在您编辑源代码时可能会有所帮助。

浏览器会忽略注释。

位于 <style> 元素内的 CSS 注释，以 /\* 开始，以 \*/ 结束：

实例

/\* 这是一条单行注释 \*/

p {

color: red;

}

您可以在代码中的任何位置添加注释：

实例

p {

color: red; /\* 把文本设置为红色 \*/

}

注释能横跨多行：

实例

/\* 这是

一条多行的

注释 \*/

p {

color: red;

}

## 6.2 HTML 和 CSS 注释

从 HTML 教程中，您学习到可以使用 <!--...--> 语法在 HTML 源代码中添加注释。

在下面的例子中，我们结合使用了 HTML 和 CSS 注释：

实例

<!DOCTYPE html>

<html>

<head>

<style>

p {

color: red; /\* 将文字颜色设置为红色 \*/

}

</style>

</head>

<body>

<h2>My Heading</h2>

<!-- 这些段落将是红色的 -->

<p>Hello World!</p>

<p>这段文本由 CSS 设置样式。</p>

<p>CSS 注释不会在输出中显示。</p>

</body>

</html>

# 第7章 CSS 颜色

指定颜色是通过使用预定义的颜色名称，或 RGB、HEX、HSL、RGBA、HSLA 值。

## 7.1 CSS 背景色

您可以为 HTML 元素设置背景色：

实例

<h1 style="background-color:DodgerBlue;">China</h1>

<p style="background-color:Tomato;">China is a great country!</p>

## 7.2 CSS 文本颜色

您可以设置文本的颜色：

China

China is a great country!

China, officially the People's Republic of China, is a country in East Asia.

实例

<h1 style="color:Tomato;">China</h1>

<p style="color:DodgerBlue;">China is a great country!</p>

<p style="color:MediumSeaGreen;">China, officially the People's Republic of China...</p>

## 7.3 CSS 边框颜色

您可以设置边框的颜色：

实例

<h1 style="border:2px solid Tomato;">Hello World</h1>

<h1 style="border:2px solid DodgerBlue;">Hello World</h1>

<h1 style="border:2px solid Violet;">Hello World</h1>

## 7.4 CSS 颜色值

在 CSS 中，还可以使用 RGB 值、HEX 值、HSL 值、RGBA 值或者 HSLA 值来指定颜色：

与颜色名 "Tomato" 等效：

<h3>实例</h3>

<h1 style="background-color:rgb(255, 99, 71);">...</h1>

<h1 style="background-color:#ff6347;">...</h1>

<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>

# 第8章 CSS RGB 颜色

## 8.1 RGB 值

在 CSS 中，可以使用下面的公式将颜色指定为 RGB 值：

rgb(red, green, blue)

每个参数 (red、green 以及 blue) 定义了 0 到 255 之间的颜色强度。

例如，rgb(255, 0, 0) 显示为红色，因为红色设置为最大值（255），其他设置为 0。

要显示黑色，请将所有颜色参数设置为 0，如下所示：rgb(0, 0, 0)。

要显示白色，请将所有颜色参数设置为 255，如下所示：rgb(255, 255, 255)。

请通过混合以下 RGB 值来进行实验：

## 8.2 RGBA 值

RGBA 颜色值是具有 alpha 通道的 RGB 颜色值的扩展 - 它指定了颜色的不透明度。

RGBA 颜色值指定为：

rgba(red, green, blue, alpha)

alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字：

请通过混合以下 RGBA 值来进行实验：

# 第9章 HEX 值

## 9.1 HEX 值

在 CSS 中，可以使用以下格式的十六进制值指定颜色：

#rrggbb

其中 rr（红色）、gg（绿色）和 bb（蓝色）是介于 00 和 ff 之间的十六进制值（与十进制 0-255 相同）。

例如，#ff0000 显示为红色，因为红色设置为最大值（ff），其他设置为最小值（00）。

请通过混合以下十六进制值来进行实验：

# 第10章 CSS HSL 颜色

## 10.1 HSL 值

在 CSS 中，可以使用色相、饱和度和明度（HSL）来指定颜色，格式如下：

hsla(hue, saturation, lightness)

色相（hue）是色轮上从 0 到 360 的度数。0 是红色，120 是绿色，240 是蓝色。

饱和度（saturation）是一个百分比值，0％ 表示灰色阴影，而 100％ 是全色。

亮度（lightness）也是百分比，0％ 是黑色，50％ 是既不明也不暗，100％是白色。

请通过混合以下 HSL 值来进行实验：

## 10.2 饱和度

饱和度可以描述为颜色的强度。

100％ 是纯色，没有灰色阴影

50％ 是 50％ 灰色，但是您仍然可以看到颜色。

0％ 是完全灰色，您无法再看到颜色。

## 10.3 亮度

颜色的明暗度可以描述为要赋予颜色多少光，其中 0％ 表示不亮（黑色），50％ 表示 50％ 亮（既不暗也不亮），100％ 表示全明（白）。

实例

<h1 style="background-color:hsl(0, 100%, 0%);">hsl(0, 100%, 0%)</h1>

<h1 style="background-color:hsl(0, 100%, 25%);">hsl(0, 100%, 25%)</h1>

<h1 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h1>

<h1 style="background-color:hsl(0, 100%, 75%);">hsl(0, 100%, 75%)</h1>

<h1 style="background-color:hsl(0, 100%, 90%);">hsl(0, 100%, 90%)</h1>

<h1 style="background-color:hsl(0, 100%, 100%);">hsl(0, 100%, 100%)</h1>

<p>通过 HSL 颜色，亮度 0% 显示黑色，亮度 100 显示白色。</p>

</body>

</html>

## 10.4 HSLA 值

HSLA 颜色值是带有 Alpha 通道的 HSL 颜色值的扩展 - 它指定了颜色的不透明度。

HSLA 颜色值指定为：

hsla(hue, saturation, lightness, alpha)

alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字：

请通过混合以下 HSLA 值进行实验：

<!DOCTYPE html>

<html>

<body>

<h1 style="background-color:hsla(9, 100%, 64%, 0);">hsla(9, 100%, 64%, 0)</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 0.2);">hsla(9, 100%, 64%, 0.2)</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 0.4);">hsla(9, 100%, 64%, 0.4)</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 0.6);">hsla(9, 100%, 64%, 0.6)</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 0.8);">hsla(9, 100%, 64%, 0.8)</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 1);">hsla(9, 100%, 64%, 1)</h1>

<p>您可使用 HSLA 颜色值生成透明色。</p>

</body>

</html>

# 第11章 CSS 背景

CSS 背景属性用于定义元素的背景效果。

在这些章节中，您将学习如下 CSS 背景属性：

background-color

background-image

background-repeat

background-attachment

background-position

## 11.1 CSS background-color

background-color 属性指定元素的背景色。

实例

页面的背景色设置如下：

body {

background-color: lightblue;

}

通过 CSS，颜色通常由以下方式指定：

有效的颜色名称 - 比如 "red"

十六进制值 - 比如 "#ff0000"

RGB 值 - 比如 "rgb(255,0,0)"

## 11.2 其他元素

您可以为任何 HTML 元素设置背景颜色：

实例

在这里，<h1>、<p> 和 <div> 元素将拥有不同的背景色：

h1 {

background-color: green;

}

div {

background-color: lightblue;

}

p {

background-color: yellow;

}

## 11.3 不透明度 / 透明度

opacity 属性指定元素的不透明度/透明度。取值范围为 0.0 - 1.0。值越低，越透明：

div {

background-color: green;

opacity: 0.3;

}

注意：使用 opacity 属性为元素的背景添加透明度时，其所有子元素都继承相同的透明度。这可能会使完全透明的元素内的文本难以阅读。

## 11.4 使用 RGBA 的透明度

如果您不希望对子元素应用不透明度，例如上面的例子，请使用 RGBA 颜色值。下面的例子设置背景色而不是文本的不透明度：

您从我们的 CSS 颜色 章节中学到了可以将 RGB 用作颜色值。除 RGB 外，还可以将 RGB 颜色值与 alpha 通道一起使用（RGBA） - 该通道指定颜色的不透明度。

RGBA 颜色值指定为：rgba(red, green, blue, alpha)。alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字。

提示：您可在我们的 CSS 颜色 一章中学到有关 RGBA 颜色的更多知识。

实例

div {

background: rgba(0, 128, 0, 0.3) /\* 30% 不透明度的绿色背景 \*/

}

# 第12章 CSS 背景图像

## 12.1 CSS 背景图像

background-image 属性指定用作元素背景的图像。

默认情况下，图像会重复，以覆盖整个元素。

实例

页面的背景图像可以像这样设置：

body {

background-image: url("paper.gif");

}

实例

本例展示了文本和背景图像的错误组合。文字难以阅读：

body {

background-image: url("bgdesert.jpg");

}

注意：使用背景图像时，请使用不会干扰文本的图像。

还可以为特定元素设置背景图像，例如 <p> 元素：

实例

p {

background-image: url("paper.gif");

}

# 第13章 CSS 背景重复

## 13.1 CSS background-repeat

默认情况下，background-image 属性在水平和垂直方向上都重复图像。

某些图像应只适合水平或垂直方向上重复，否则它们看起来会很奇怪，如下所示：

实例

body {

background-image: url("gradient\_bg.png");

}

如果上面的图像仅在水平方向重复 (background-repeat: repeat-x;)，则背景看起来会更好：

实例

body {

background-image: url("gradient\_bg.png");

background-repeat: repeat-x;

}

提示：如需垂直重复图像，请设置 background-repeat: repeat-y;。

## 13.2 CSS background-repeat: no-repeat

background-repeat 属性还可指定只显示一次背景图像：

实例

背景图像仅显示一次：

body {

background-image: url("tree.png");

background-repeat: no-repeat;

}

在上例中，背景图像与文本放置在同一位置。我们想要更改图像的位置，以免图像过多干扰文本。

## 13.3 CSS background-position

background-position 属性用于指定背景图像的位置。

实例

把背景图片放在右上角：

body {

background-image: url("tree.png");

background-repeat: no-repeat;

background-position: right top;

}

# 第14章 CSS 背景附着

## 14.1 CSS background-attachment

background-attachment 属性指定背景图像是应该滚动还是固定的（不会随页面的其余部分一起滚动）：

实例

指定应该固定背景图像：

body {

background-image: url("tree.png");

background-repeat: no-repeat;

background-position: right top;

background-attachment: fixed;

}

实例

指定背景图像应随页面的其余部分一起滚动：

body {

background-image: url("tree.png");

background-repeat: no-repeat;

background-position: right top;

background-attachment: scroll;

}

# 第15章 CSS 背景简写

## 15.1 CSS background - 简写属性

如需缩短代码，也可以在一个属性中指定所有背景属性。它被称为简写属性。

而不是这样写：

body {

background-color: #ffffff;

background-image: url("tree.png");

background-repeat: no-repeat;

background-position: right top;

}

您能够使用简写属性 background：

实例

使用简写属性在一条声明中设置背景属性：

body {

background: #ffffff url("tree.png") no-repeat right top;

}

在使用简写属性时，属性值的顺序为：

background-color

background-image

background-repeat

background-attachment

background-position

属性值之一缺失并不要紧，只要按照此顺序设置其他值即可。请注意，在上面的例子中，我们没有使用 background-attachment 属性，因为它没有值。

## 15.2 所有 CSS 背景属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| background | 在一条声明中设置所有背景属性的简写属性。 |
| background-attachment | 设置背景图像是固定的还是与页面的其余部分一起滚动。 |
| background-clip | 规定背景的绘制区域。 |
| background-color | 设置元素的背景色。 |
| background-image | 设置元素的背景图像。 |
| background-origin | 规定在何处放置背景图像。 |
| background-position | 设置背景图像的开始位置。 |
| background-repeat | 设置背景图像是否及如何重复。 |
| background-size | 规定背景图像的尺寸。 |

# 第16章 CSS 边框

## 16.1 CSS 边框属性

CSS border 属性允许您指定元素边框的样式、宽度和颜色。

我的所有边都有边框。

我有一条红色的下边框。

我有圆角边框。

我有一条蓝色的左边框。

## 16.2 CSS 边框样式

border-style 属性指定要显示的边框类型。

允许以下值：

dotted - 定义点线边框

dashed - 定义虚线边框

solid - 定义实线边框

double - 定义双边框

groove - 定义 3D 坡口边框。效果取决于 border-color 值

ridge - 定义 3D 脊线边框。效果取决于 border-color 值

inset - 定义 3D inset 边框。效果取决于 border-color 值

outset - 定义 3D outset 边框。效果取决于 border-color 值

none - 定义无边框

hidden - 定义隐藏边框

border-style 属性可以设置一到四个值（用于上边框、右边框、下边框和左边框）。

实例

演示不同的边框样式：

p.dotted {border-style: dotted;}

p.dashed {border-style: dashed;}

p.solid {border-style: solid;}

p.double {border-style: double;}

p.groove {border-style: groove;}

p.ridge {border-style: ridge;}

p.inset {border-style: inset;}

p.outset {border-style: outset;}

p.none {border-style: none;}

p.hidden {border-style: hidden;}

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

注意：除非设置了 border-style 属性，否则其他 CSS 边框属性（将在下一章中详细讲解）都不会有任何作用！

# 第17章 CSS 边框宽度

## 17.1 CSS 边框宽度

border-width 属性指定四个边框的宽度。

可以将宽度设置为特定大小（以 px、pt、cm、em 计），也可以使用以下三个预定义值之一：thin、medium 或 thick：

实例

演示不同的边框宽度：

p.one {

border-style: solid;

border-width: 5px;

}

p.two {

border-style: solid;

border-width: medium;

}

p.three {

border-style: dotted;

border-width: 2px;

}

p.four {

border-style: dotted;

border-width: thick;

}

## 17.2 特定边的宽度

border-width 属性可以设置一到四个值（用于上边框、右边框、下边框和左边框）：

实例

p.one {

border-style: solid;

border-width: 5px 20px; /\* 上边框和下边框为 5px，其他边为 20px \*/

}

p.two {

border-style: solid;

border-width: 20px 5px; /\* 上边框和下边框为 20px，其他边为 5px \*/

}

p.three {

border-style: solid;

border-width: 25px 10px 4px 35px; /\* 上边框 25px，右边框 10px，下边框 4px，左边框 35px \*/

}

# 第18章 CSS 边框颜色

## 18.1 CSS 边框颜色

border-color 属性用于设置四个边框的颜色。

可以通过以下方式设置颜色：

name - 指定颜色名，比如 "red"

HEX - 指定十六进制值，比如 "#ff0000"

RGB - 指定 RGB 值，比如 "rgb(255,0,0)"

HSL - 指定 HSL 值，比如 "hsl(0, 100%, 50%)"

transparent

注释：如果未设置 border-color，则它将继承元素的颜色。

实例

演示不同的边框颜色：

p.one {

border-style: solid;

border-color: red;

}

p.two {

border-style: solid;

border-color: green;

}

p.three {

border-style: dotted;

border-color: blue;

}

## 18.2 特定边框的颜色

border-color 属性可以设置一到四个值（用于上边框、右边框、下边框和左边框）。

实例

p.one {

border-style: solid;

border-color: red green blue yellow; /\* 上红、右绿、下蓝、左黄 \*/

}

## 18.3 HEX 值

边框的颜色也可以使用十六进制值（HEX）来指定：

实例

p.one {

border-style: solid;

border-color: #ff0000; /\* 红色 \*/

}

## 18.4 RGB 值

或者使用 RGB 值：

实例

p.one {

border-style: solid;

border-color: rgb(255, 0, 0); /\* 红色 \*/

}

## 18.5 HSL 值

也可以使用 HSL 值：

实例

p.one {

border-style: solid;

border-color: hsl(0, 100%, 50%); /\* 红色 \*/

}

# 第19章 CSS 边框各边

## 19.1 CSS 边框 - 单独的边

从上一章的例子中，您已经看到可以为每一侧指定不同的边框。

在 CSS 中，还有一些属性可用于指定每个边框（顶部、右侧、底部和左侧）：

实例

p {

border-top-style: dotted;

border-right-style: solid;

border-bottom-style: dotted;

border-left-style: solid;

}

## 19.2 不同的边框样式

上例的结果与此相同：

实例

p {

border-style: dotted solid;

}

它的工作原理是这样的：

如果 border-style 属性设置四个值：

border-style: dotted solid double dashed;

上边框是虚线

右边框是实线

下边框是双线

左边框是虚线

如果 border-style 属性设置三个值：

border-style: dotted solid double;

上边框是虚线

右和左边框是实线

下边框是双线

如果 border-style 属性设置两个值：

border-style: dotted solid;

上和下边框是虚线

右和左边框是实线

如果 border-style 属性设置一个值：

border-style: dotted;

四条边均为虚线

实例

/\* 四个值 \*/

p {

border-style: dotted solid double dashed;

}

/\* 三个值 \*/

p {

border-style: dotted solid double;

}

/\* 两个值 \*/

p {

border-style: dotted solid;

}

/\* 一个值 \*/

p {

border-style: dotted;

}

上例中使用的是 border-style 属性。但 border-width 和 border-color 也同样适用。

# 第20章 CSS 简写边框属性

## 20.1 CSS Border - 简写属性

就像您在上一章中所见，处理边框时要考虑许多属性。

为了缩减代码，也可以在一个属性中指定所有单独的边框属性。

border 属性是以下各个边框属性的简写属性：

border-width

border-style（必需）

border-color

实例

p {

border: 5px solid red;

}

您还可以只为一个边指定所有单个边框属性：

左边框

p {

border-left: 6px solid red;

background-color: lightgrey;

}

下边框

p {

border-bottom: 6px solid red;

background-color: lightgrey;

}

# 第21章 CSS 圆角边框

## 21.1 CSS 圆角边框

border-radius 属性用于向元素添加圆角边框：

实例

p {

border: 2px solid red;

border-radius: 5px;

}

## 21.2 一个声明中的所有上边框属性

<!DOCTYPE html>

<html>

<head>

<style>

p {

border-style: solid;

border-top: thick double #ff0000;

}

</style>

</head>

<body>

<p>这是段落中的一些文本。</p>

</body>

</html>

## 21.3 设置下边框的样式

<!DOCTYPE html>

<html>

<head>

<style>

p {border-style: solid;}

p.none {border-bottom-style: none;}

p.dotted {border-bottom-style: dotted;}

p.dashed {border-bottom-style: dashed;}

p.solid {border-bottom-style: solid;}

p.double {border-bottom-style: double;}

p.groove {border-bottom-style: groove;}

p.ridge {border-bottom-style: ridge;}

p.inset {border-bottom-style: inset;}

p.outset {border-bottom-style: outset;}

</style>

</head>

<body>

<p class="none">无下边框。</p>

<p class="dotted">点状下边框。</p>

<p class="dashed">虚线下边框。</p>

<p class="solid">实线下边框。</p>

<p class="double">双线下边框。</p>

<p class="groove">凹槽下边框。</p>

<p class="ridge">垄状下边框。</p>

<p class="inset">内陷下边框。</p>

<p class="outset">凸出下边框。</p>

</body>

</html>

## 21.4 设置左边框的宽度

<!DOCTYPE html>

<html>

<head>

<style>

p {

border-style: solid;

border-left-width: 15px;

}

</style>

</head>

<body>

<p><b>注释：</b>如果单独使用，"border-left-width" 属性不起作用。请首先使用 "border-style" 属性来设置边框。</p>

</body>

</html>

## 21.5 设置四条边框的颜色

<!DOCTYPE html>

<html>

<head>

<style>

p.one {

border-style: solid;

border-color: #0000ff;

}

p.two {

border-style: solid;

border-color: #ff0000 #0000ff;

}

p.three {

border-style: solid;

border-color: #ff0000 #00ff00 #0000ff;

}

p.four {

border-style: solid;

border-color: #ff0000 #00ff00 #0000ff rgb(250,0,255);

}

</style>

</head>

<body>

<p class="one">一种颜色的边框！</p>

<p class="two">两种颜色的边框！</p>

<p class="three">三种颜色的边框！</p>

<p class="four">四种颜色的边框！</p>

<p><b>注释：</b>如果单独使用，"border-color" 属性不起作用。请首先使用 "border-style" 属性来设置边框。</p>

</body>

</html>

## 21.6 设置右边框的颜色

<!DOCTYPE html>

<html>

<head>

<style>

p {

border-style: solid;

border-right-color: #ff0000;

}

</style>

</head>

<body>

<p>这是段落中的一些文本。</p>

</body>

</html>

## 21.7 所有 CSS 边框属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| border | 简写属性，在一条声明中设置所有边框属性。 |
| border-color | 简写属性，设置四条边框的颜色。 |
| border-radius | 简写属性，可设置圆角的所有四个 border-\*-radius 属性。 |
| border-style | 简写属性，设置四条边框的样式。 |
| border-width | 简写属性，设置四条边框的宽度。 |
| border-bottom | 简写属性，在一条声明中设置所有下边框属性。 |
| border-bottom-color | 设置下边框的颜色。 |
| border-bottom-style | 设置下边框的样式。 |
| border-bottom-width | 设置下边框的宽度。 |
| border-left | 简写属性，在一条声明中设置所有左边框属性。 |
| border-left-color | 设置左边框的颜色。 |
| border-left-style | 设置左边框的样式。 |
| border-left-width | 设置左边框的宽度。 |
| border-right | 简写属性，在一条声明中设置所有右边框属性。 |
| border-right-color | 设置右边框的颜色。 |
| border-right-style | 设置右边框的样式。 |
| border-right-width | 设置右边框的宽度。 |
| border-top | 简写属性，在一条声明中设置所有上边框属性。 |
| border-top-color | 设置上边框的颜色。 |
| border-top-style | 设置上边框的样式。 |
| border-top-width | 设置上边框的宽度。 |

# 第22章 CSS 外边距

## 22.1 CSS 外边距

CSS margin 属性用于在任何定义的边框之外，为元素周围创建空间。

通过 CSS，您可以完全控制外边距。有一些属性可用于设置元素每侧（上、右、下和左）的外边距。

## 22.2 Margin - 单独的边

CSS 拥有用于为元素的每一侧指定外边距的属性：

margin-top

margin-right

margin-bottom

margin-left

所有外边距属性都可以设置以下值：

auto - 浏览器来计算外边距

length - 以 px、pt、cm 等单位指定外边距

% - 指定以包含元素宽度的百分比计的外边距

inherit - 指定应从父元素继承外边距

提示：允许负值。

实例

为 <p> 元素的所有四个边设置不同的外边距：

p {

margin-top: 100px;

margin-bottom: 100px;

margin-right: 150px;

margin-left: 80px;

}

## 22.3 Margin - 简写属性

为了缩减代码，可以在一个属性中指定所有外边距属性。

margin 属性是以下各外边距属性的简写属性：

margin-top

margin-right

margin-bottom

margin-left

工作原理是这样的：

如果 margin 属性有四个值：

margin: 25px 50px 75px 100px;

上外边距是 25px

右外边距是 50px

下外边距是 75px

左外边距是 100px

实例

margin 简写属性设置四个值：

p {

margin: 25px 50px 75px 100px;

}

如果 margin 属性设置三个值：

margin: 25px 50px 75px;

上外边距是 25px

右和左外边距是 50px

下外边距是 75px

实例

使用已设置三个值的 margin 简写属性：

p {

margin: 25px 50px 75px;

}

如果 margin 属性设置两个值：

margin: 25px 50px;

上和下外边距是 25px

右和左外边距是 50px

实例

使用设置了两个值的 margin 简写属性：

p {

margin: 25px 50px;

}

如果 margin 属性设置了一个值：

margin: 25px;

所有四个外边距都是 25px

实例

使用设置一个值的 margin 简写属性：

p {

margin: 25px;

}

## 22.4 auto 值

您可以将 margin 属性设置为 auto，以使元素在其容器中水平居中。

然后，该元素将占据指定的宽度，并且剩余空间将在左右边界之间平均分配。

实例

使用 margin: auto：

div {

width: 300px;

margin: auto;

border: 1px solid red;

}

## 22.5 inherit 值

本例使 <p class="ex1"> 元素的左外边距继承自父元素（<div>）：

实例

使用 inherit 值：

div {

border: 1px solid red;

margin-left: 100px;

}

p.ex1 {

margin-left: inherit;

}

# 第23章 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。

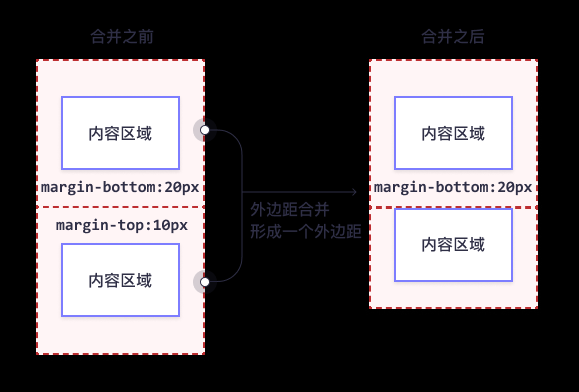
合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

## 23.1 外边距合并

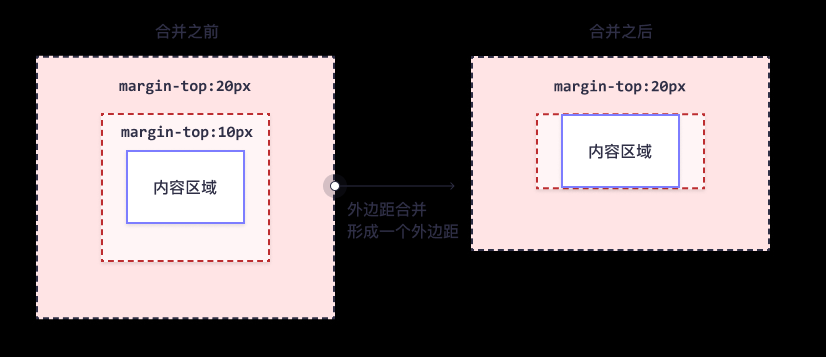
外边距合并（叠加）是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。

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：

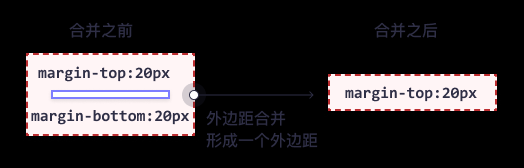


当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并。请看下图：

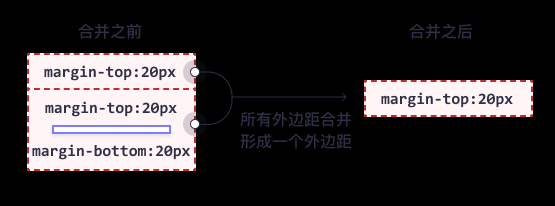


尽管看上去有些奇怪，但是外边距甚至可以与自身发生合并。

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：

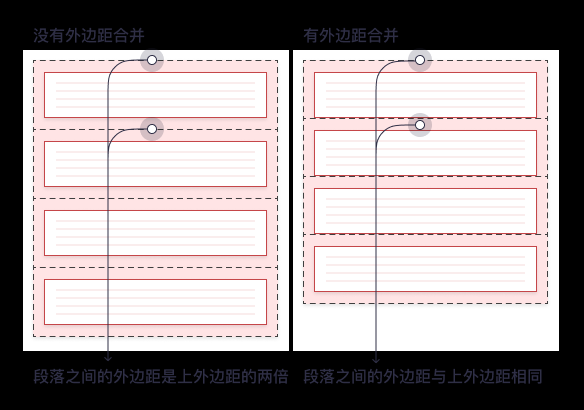


如果这个外边距遇到另一个元素的外边距，它还会发生合并：



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。

## 23.2 所有 CSS 外边距属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| margin | 用于在一条声明中设置外边距属性的简写属性。 |
| 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。 |

# 第24章 CSS 内边距

## 24.1 CSS 内边距

CSS padding 属性用于在任何定义的边界内的元素内容周围生成空间。

通过 CSS，您可以完全控制内边距（填充）。有一些属性可以为元素的每一侧（上、右、下和左侧）设置内边距。

## 24.2 Padding - 单独的边

CSS 拥有用于为元素的每一侧指定内边距的属性：

padding-top

padding-right

padding-bottom

padding-left

所有内边距属性都可以设置以下值：

length - 以 px、pt、cm 等单位指定内边距

% - 指定以包含元素宽度的百分比计的内边距

inherit - 指定应从父元素继承内边距

提示：不允许负值。

实例

为 <div> 元素的所有四个边设置不同的内边距：

div {

padding-top: 50px;

padding-right: 30px;

padding-bottom: 50px;

padding-left: 80px;

}

## 24.3 Padding - 简写属性

为了缩减代码，可以在一个属性中指定所有内边距属性。

padding 属性是以下各内边距属性的简写属性：

padding-top

padding-right

padding-bottom

padding-left

工作原理是这样的：

如果 padding 属性有四个值：

padding: 25px 50px 75px 100px;

上内边距是 25px

右内边距是 50px

下内边距是 75px

左内边距是 100px

实例

使用设置了四个值的 padding 简写属性：

div {

padding: 25px 50px 75px 100px;

}

如果 padding 属性设置了三个值：

padding: 25px 50px 75px;

上内边距是 25px

右和左内边距是 50px

下内边距是 75px

实例

使用设置了三个值的 padding 简写属性：

div {

padding: 25px 50px 75px;

}

如果 padding 属性设置了两个值：

padding: 25px 50px;

上和下内边距是 25px

右和左内边距是 50px

实例

使用设置了两个值的 padding 简写属性：

div {

padding: 25px 50px;

}

如果 padding 属性设置了一个值：

padding: 25px;

所有四个内边距都是 25px

实例

使用设置了一个值的 padding 简写属性：

div {

padding: 25px;

}

## 24.4 内边距和元素宽度

CSS width 属性指定元素内容区域的宽度。内容区域是元素（盒模型）的内边距、边框和外边距内的部分。

因此，如果元素拥有指定的宽度，则添加到该元素的内边距会添加到元素的总宽度中。这通常是不希望的结果。

实例

在这里，<div> 元素的宽度为 300px。但是，<div> 元素的实际宽度将是 350px（300px + 左内边距 25px + 右内边距 25px）：

div {

width: 300px;

padding: 25px;

}

若要将宽度保持为 300px，无论填充量如何，那么您可以使用 box-sizing 属性。这将导致元素保持其宽度。如果增加内边距，则可用的内容空间会减少。

实例

使用 box-sizing 属性将宽度保持为 300px，无论填充量如何：

div {

width: 300px;

padding: 25px;

box-sizing: border-box;

}

## 24.5 所有 CSS 内边距属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| padding | 用于在一条声明中设置所有内边距属性的简写属性。 |
| 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。 |

# 第25章 CSS 高度和宽度

## 25.1 CSS 设置高度和宽度

height 和 width 属性用于设置元素的高度和宽度。

height 和 width 属性不包括内边距、边框或外边距。它设置的是元素内边距、边框以及外边距内的区域的高度或宽度。

## 25.2 CSS 高度和宽度值

height 和 width 属性可设置如下值：

auto - 默认。浏览器计算高度和宽度。

length - 以 px、cm 等定义高度/宽度。

% - 以包含块的百分比定义高度/宽度。

initial - 将高度/宽度设置为默认值。

inherit - 从其父值继承高度/宽度。

## 25.3 CSS 高度和宽度实例

实例

设置 <div> 元素的高度和宽度：

div {

height: 200px;

width: 50%;

background-color: powderblue;

}

实例

设置另一个 <div> 元素的高度和宽度：

div {

height: 100px;

width: 500px;

background-color: powderblue;

}

注意：请记住，height 和 width 属性不包括内边距、边框或外边距！它们设置的是元素的内边距、边框和外边距内的区域的高度/宽度！

## 25.4 设置 max-width

max-width 属性用于设置元素的最大宽度。

可以用长度值（例如 px、cm 等）或包含块的百分比（％）来指定 max-width（最大宽度），也可以将其设置为 none（默认值。意味着没有最大宽度）。

当浏览器窗口小于元素的宽度（500px）时，会发生之前那个 <div> 的问题。然后，浏览器会将水平滚动条添加到页面。

在这种情况下，使用 max-width 能够改善浏览器对小窗口的处理。

提示：将浏览器窗口拖动到小于500px的宽度，以查看两个 div 之间的区别！

此元素的高度为 100 像素，最大宽度为 500 像素。

此元素的高度为 100 像素，最大宽度为 500 像素。

注释：max-width 属性的值将覆盖 width（宽度）。

实例

这个 <div> 元素的高度为 100 像素，最大宽度为 500 像素：

div {

max-width: 500px;

height: 100px;

background-color: powderblue;

}

## 25.5 设置 CSS 尺寸属性

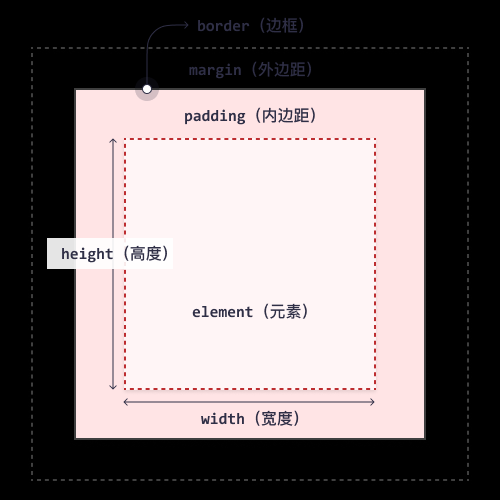
|  |  |
| --- | --- |
| 属性 | 描述 |
| height | 设置元素的高度。 |
| max-height | 设置元素的最大高度。 |
| max-width | 设置元素的最大宽度。 |
| min-height | 设置元素的最小高度。 |
| min-width | 设置元素的最小宽度。 |
| width | 设置元素的宽度。 |

# 第26章 CSS 框模型

## 26.1 CSS 框模型

所有 HTML 元素都可以视为方框。在 CSS 中，在谈论设计和布局时，会使用术语“盒模型”或“框模型”。

CSS 框模型实质上是一个包围每个 HTML 元素的框。它包括：外边距、边框、内边距以及实际的内容。下图展示了框模型：



对不同部分的说明：

1.内容 - 框的内容，其中显示文本和图像。

2.内边距 - 清除内容周围的区域。内边距是透明的。

3.边框 - 围绕内边距和内容的边框。

4.外边距 - 清除边界外的区域。外边距是透明的。

框模型允许我们在元素周围添加边框，并定义元素之间的空间。

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：

\* {

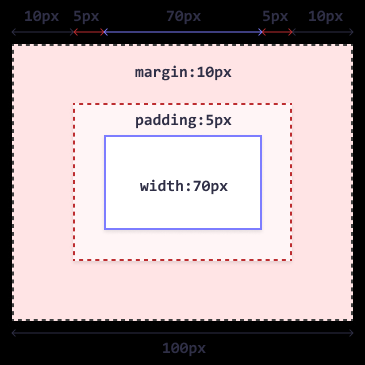
margin: 0;

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。

实例

演示框模型：

div {

width: 300px;

border: 15px solid green;

padding: 50px;

margin: 20px;

}

## 26.2 元素的宽度和高度

为了在所有浏览器中正确设置元素的宽度和高度，您需要了解框模型如何工作。

重要提示：使用 CSS 设置元素的 width 和 height 属性时，只需设置内容区域的宽度和高度。要计算元素的完整大小，还必须把内边距、边框和外边距加起来。

实例

<div> 元素的总宽度将是 350px：

div {

width: 320px;

padding: 10px;

border: 5px solid gray;

margin: 0;

}

## 26.3 计算如下：

320px(宽度)

+ 20px（左+右内边距）

+ 10px（左+右边框）

+ 0px（左+右外边距）

= 350px

元素的总宽度应该这样计算：

元素总宽度 = 宽度 + 左内边距 + 右内边距 + 左边框 + 右边框 + 左外边距 + 右外边距

元素的总高度应该这样计算：

元素总高度 = 高度 + 上内边距 + 下内边距 + 上边框 + 下边框 + 上外边距 + 下外边距

# 第27章 CSS 轮廓

## 27.1 CSS 轮廓

轮廓是在元素周围绘制的一条线，在边框之外，以凸显元素。

CSS 拥有如下轮廓属性：

outline-style

outline-color

outline-width

outline-offset

outline

注意：轮廓与边框不同！不同之处在于：轮廓是在元素边框之外绘制的，并且可能与其他内容重叠。同样，轮廓也不是元素尺寸的一部分；元素的总宽度和高度不受轮廓线宽度的影响。

## 27.2 CSS 轮廓样式

outline-style 属性指定轮廓的样式，并可设置如下值：

dotted - 定义点状的轮廓。

dashed - 定义虚线的轮廓。

solid - 定义实线的轮廓。

double - 定义双线的轮廓。

groove - 定义 3D 凹槽轮廓。

ridge - 定义 3D 凸槽轮廓。

inset - 定义 3D 凹边轮廓。

outset - 定义 3D 凸边轮廓。

none - 定义无轮廓。

hidden - 定义隐藏的轮廓。

下例展示了不同的 outline-style 值：

实例

演示不同的轮廓样式：

p.dotted {outline-style: dotted;}

p.dashed {outline-style: dashed;}

p.solid {outline-style: solid;}

p.double {outline-style: double;}

p.groove {outline-style: groove;}

p.ridge {outline-style: ridge;}

p.inset {outline-style: inset;}

p.outset {outline-style: outset;}

注意：除非设置了 outline-style 属性，否则其他轮廓属性（在下一章中将详细介绍）都不会有任何作用！

# 第28章 CSS 轮廓宽度

## 28.1 CSS 轮廓宽度

outline-width 属性指定轮廓的宽度，并可设置如下值之一：

thin（通常为 1px）

medium（通常为 3px）

thick （通常为 5px）

特定尺寸（以 px、pt、cm、em 计）

下例展示了一些不同宽度的轮廓：

实例

p.ex1 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: thin;

}

p.ex2 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: medium;

}

p.ex3 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: thick;

}

p.ex4 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: 4px;

}

# 第29章 CSS 轮廓颜色

## 29.1 CSS 轮廓颜色

outline-color 属性用于设置轮廓的颜色。

可以通过以下方式设置颜色：

name - 指定颜色名，比如 "red"

HEX - 指定十六进制值，比如 "#ff0000"

RGB - 指定 RGB 值，比如 "rgb(255,0,0)"

HSL - 指定 HSL 值，比如 "hsl(0, 100%, 50%)"

invert - 执行颜色反转（确保轮廓可见，无论是什么颜色背景）

下例展示了一些不同颜色的不同轮廓样式。请注意，这些元素在轮廓内还有黑色细边框：

实例

p.ex1 {

border: 2px solid black;

outline-style: solid;

outline-color: red;

}

p.ex2 {

border: 2px solid black;

outline-style: dotted;

outline-color: blue;

}

p.ex3 {

border: 2px solid black;

outline-style: outset;

outline-color: grey;

}

HEX 值

您也可以使用十六进制值（HEX）指定轮廓颜色：

实例

p.ex1 {

outline-style: solid;

outline-color: #ff0000; /\* 红色 \*/

}

RGB 值

或者通过使用 RGB 值：

实例

p.ex1 {

outline-style: solid;

outline-color: rgb(255, 0, 0); /\* 红色 \*/

}

HSL 值

您还可以使用 HSL 值：

实例

p.ex1 {

outline-style: solid;

outline-color: hsl(0, 100%, 50%); /\* 红色 \*/

}

## 29.2 反转颜色

下例使用 outline-color: invert，执行了颜色反转。这样可以确保无论颜色背景如何，轮廓都是可见的：

p.ex1 {

border: 1px solid yellow;

outline-style: solid;

outline-color: invert;

}

# 第30章 CSS 轮廓简写

## 30.1 CSS Outline - 简写属性

outline 属性是用于设置以下各个轮廓属性的简写属性：

outline-width

outline-style（必需）

outline-color

从上面的列表中，outline 属性可指定一个、两个或三个值。值的顺序无关紧要。

下例展示了用简写的 outline 属性指定的一些轮廓：

实例

p.ex1 {outline: dashed;}

p.ex2 {outline: dotted red;}

p.ex3 {outline: 5px solid yellow;}

p.ex4 {outline: thick ridge pink;}

# 第31章 CSS 轮廓偏移

## 31.1 CSS 轮廓偏移

outline-offset 属性在元素的轮廓与边框之间添加空间。元素及其轮廓之间的空间是透明的。

下例指定边框边缘外 25px 的轮廓：

实例

p {

margin: 50px;

border: 1px solid black;

outline: 1px solid red;

outline-offset: 25px;

}

下例显示元素与其轮廓之间的空间是透明的：

实例

p {

margin: 30px;

background: yellow;

border: 1px solid black;

outline: 1px solid red;

outline-offset: 25px;

}

## 31.2 所有 CSS 轮廓属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| outline | 简写属性，在一条声明中设置 outline-width、outline-style 以及 outline-color。 |
| outline-color | 设置轮廓的颜色。 |
| outline-offset | 指定轮廓与元素的边缘或边框之间的空间。 |
| outline-style | 设置轮廓的样式。 |
| outline-width | 设置轮廓的宽度。 |

# 第32章 CSS 文本

## 32.1 文本颜色

color 属性用于设置文本的颜色。颜色由以下值指定：

颜色名 - 比如 "red"

十六进制值 - 比如 "#ff0000"

RGB 值 - 比如 "rgb(255,0,0)"

查看 CSS 颜色值，以获取可能颜色值的完整列表。

页面的默认文本颜色是在 body 选择器中定义的。

实例

body {

color: blue;

}

h1 {

color: green;

}

提示：对于 W3C compliant CSS：如果您定义了 color 属性，则还必须定义 background-color 属性。

## 32.2 文本颜色和背景色

在本例中，我们定义了 background-color 属性和 color 属性：

实例

body {

background-color: lightgrey;

color: blue;

}

h1 {

background-color: black;

color: white;

}

# 第33章 CSS 文本对齐

## 33.1 文本对齐

text-align 属性用于设置文本的水平对齐方式。

文本可以左对齐或右对齐，或居中对齐。

下例展示了居中对齐以及左右对齐的文本（如果文本方向是从左到右，则默认为左对齐；如果文本方向是从右到左，则默认是右对齐）：

实例

h1 {

text-align: center;

}

h2 {

text-align: left;

}

h3 {

text-align: right;

}

当 text-align 属性设置为 "justify" 后，将拉伸每一行，以使每一行具有相等的宽度，并且左右边距是直的（就像在杂志和报纸中）：

实例

div {

text-align: justify;

}

## 33.2 文本方向

direction 和 unicode-bidi 属性可用于更改元素的文本方向：

实例

p {

direction: rtl;

unicode-bidi: bidi-override;

}

## 33.3 垂直对齐

vertical-align 属性设置元素的垂直对齐方式。

本例演示如何设置文本中图像的垂直对齐方式：

实例

img.top {

vertical-align: top;

}

img.middle {

vertical-align: middle;

}

img.bottom {

vertical-align: bottom;

}

# 第34章 CSS 文字装饰

## 34.1 文字装饰

text-decoration 属性用于设置或删除文本装饰。

text-decoration: none; 通常用于从链接上删除下划线：

实例

a {

text-decoration: none;

}

其他 text-decoration 值用于装饰文本：

实例

h1 {

text-decoration: overline;

}

h2 {

text-decoration: line-through;

}

h3 {

text-decoration: underline;

}

注释：建议不要在非链接文本加下划线，因为这经常会使读者感到困惑。

# 第35章 CSS 文本转换

文本转换

text-transform 属性用于指定文本中的大写和小写字母。

它可用于将所有内容转换为大写或小写字母，或将每个单词的首字母大写：

实例

p.uppercase {

text-transform: uppercase;

}

p.lowercase {

text-transform: lowercase;

}

p.capitalize {

text-transform: capitalize;

}

# 第36章 CSS 文字间距

## 36.1 文字缩进

text-indent 属性用于指定文本第一行的缩进：

实例

p {

text-indent: 50px;

}

## 36.2 字母间距

letter-spacing 属性用于指定文本中字符之间的间距。

下例演示如何增加或减少字符之间的间距：

实例

h1 {

letter-spacing: 3px;

}

h2 {

letter-spacing: -3px;

}

## 36.3 行高

line-height 属性用于指定行之间的间距：

实例

p.small {

line-height: 0.8;

}

p.big {

line-height: 1.8;

}

## 36.4 字间距

word-spacing 属性用于指定文本中单词之间的间距。

下例演示如何增加或减少单词之间的间距：

实例

h1 {

word-spacing: 10px;

}

h2 {

word-spacing: -5px;

}

## 36.5 空白

white-space 属性指定元素内部空白的处理方式。

此例演示如何禁用元素内的文本换行：

实例

p {

white-space: nowrap;

}

# 第37章 CSS 文本阴影

## 37.1 文本阴影

text-shadow 属性为文本添加阴影。

最简单的用法是只指定水平阴影（2px）和垂直阴影（2px）：

文字阴影效果！

实例

h1 {

text-shadow: 2px 2px;

}

实例

h1 {

text-shadow: 2px 2px red;

}

实例

h1 {

text-shadow: 2px 2px 5px red;

}

## 37.2 所有 CSS 文本属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| color | 设置文本颜色。 |
| direction | 指定文本的方向 / 书写方向。 |
| letter-spacing | 设置字符间距。 |
| line-height | 设置行高。 |
| text-align | 指定文本的水平对齐方式。 |
| text-decoration | 指定添加到文本的装饰效果。 |
| text-indent | 指定文本块中首行的缩进。 |
| text-shadow | 指定添加到文本的阴影效果。 |
| text-transform | 控制文本的大小写。 |
| text-overflow | 指定应如何向用户示意未显示的溢出内容。 |
| unicode-bidi | 与 direction 属性一起使用，设置或返回是否应重写文本来支持同一文档中的多种语言。 |
| vertical-align | 指定文本的垂直对齐方式。 |
| white-space | 指定如何处理元素内的空白。 |
| word-spacing | 设置单词间距。 |

# 第38章 CSS 字体

## 38.1 字体选择很重要

选择正确的字体会对网站的用户体验产生巨大影响。

正确的字体可以为您的品牌创造强有力的形象。

使用易于阅读的字体很重要。字体为您的文本增加了价值。为字体选择正确的颜色和文本大小也很重要。

## 38.2 通用字体族

在 CSS 中，有五个通用字体族：

衬线字体（Serif）- 在每个字母的边缘都有一个小的笔触。它们营造出一种形式感和优雅感。

无衬线字体（Sans-serif）- 字体线条简洁（没有小笔画）。它们营造出现代而简约的外观。

等宽字体（Monospace）- 这里所有字母都有相同的固定宽度。它们创造出机械式的外观。

草书字体（Cursive）- 模仿了人类的笔迹。

幻想字体（Fantasy）- 是装饰性/俏皮的字体。

所有不同的字体名称都属于这五个通用字体系列之一。

提示：在计算机屏幕上，无衬线字体被认为比衬线字体更易于阅读。

## 38.3 CSS font-family 属性

在 CSS 中，我们使用 font-family 属性规定文本的字体。

font-family 属性应包含多个字体名称作为“后备”系统，以确保浏览器/操作系统之间的最大兼容性。请以您需要的字体开始，并以通用系列结束（如果没有其他可用字体，则让浏览器选择通用系列中的相似字体）。字体名称应以逗号分隔。

注释：如果字体名称不止一个单词，则必须用引号引起来，例如："Times New Roman"。

实例

为三个段落规定不同的字体：

.p1 {

font-family: "Times New Roman", Times, serif;

}

.p2 {

font-family: Arial, Helvetica, sans-serif;

}

.p3 {

font-family: "Lucida Console", "Courier New", monospace;

}

# 第39章 CSS 字体样式

## 39.1 字体样式

font-style 属性主要用于指定斜体文本。

此属性可设置三个值：

normal - 文字正常显示

italic - 文本以斜体显示

oblique - 文本为“倾斜”（倾斜与斜体非常相似，但支持较少）

实例

p.normal {

font-style: normal;

}

p.italic {

font-style: italic;

}

p.oblique {

font-style: oblique;

}

## 39.2 字体粗细

font-weight 属性指定字体的粗细：

实例

p.normal {

font-weight: normal;

}

p.thick {

font-weight: bold;

}

## 39.3 字体变体

font-variant 属性指定是否以 small-caps 字体（小型大写字母）显示文本。

在 small-caps 字体中，所有小写字母都将转换为大写字母。但是，转换后的大写字母的字体大小小于文本中原始大写字母的字体大小。

实例

p.normal {

font-variant: normal;

}

p.small {

font-variant: small-caps;

}

# 第40章 CSS 字体大小

## 40.1 字体大小

font-size 属性设置文本的大小。

在网页设计中，能够管理文本大小很重要。但是，不应使用调整字体大小来使段落看起来像标题，或是使标题看起来像段落。

请始终使用正确的 HTML 标签，例如 <h1> - <h6> 用于标题，而 <p> 仅用于段落。

font-size 值可以是绝对或相对大小。

绝对尺寸：

将文本设置为指定大小

不允许用户在所有浏览器中更改文本大小（可访问性不佳）

当输出的物理尺寸已知时，绝对尺寸很有用

相对尺寸：

设置相对于周围元素的大小

允许用户在浏览器中更改文本大小

注释：如果您没有指定字体大小，则普通文本（如段落）的默认大小为 16px（16px = 1em）。

## 40.2 以像素设置字体大小

使用像素设置文本大小可以完全控制文本大小：

实例

h1 {

font-size: 40px;

}

h2 {

font-size: 30px;

}

p {

font-size: 14px;

}

提示：如果您使用了像素，则仍然可以使用缩放工具来调整整个页面的大小。

## 40.3 用 em 设置字体大小

为了允许用户调整文本大小（在浏览器菜单中），许多开发人员使用 em 而不是像素。

W3C 建议使用 em 尺寸单位。

1em 等于当前字体大小。浏览器中的默认文本大小为 16px。因此，默认大小 1em 为 16px。

可以使用这个公式从像素到 em 来计算大小：pixels/16=em。

实例

h1 {

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

}

h2 {

font-size: 1.875em; /\* 30px/16=1.875em \*/

}

p {

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

}

在上例中，em 单位的文本大小与上一个例子中的像素大小相同。但是，若使用 em 尺寸，则可以在所有浏览器中调整文本大小。

不幸的是，旧版本的 Internet Explorer 仍然存在问题。放大文本时它比应该大的尺寸更大，缩小文本时会更小。

## 40.4 使用百分比和 Em 的组合

实例

body {

font-size: 100%;

}

h1 {

font-size: 2.5em;

}

h2 {

font-size: 1.875em;

}

p {

font-size: 0.875em;

}

我们的代码目前运行良好！它在所有浏览器中显示相同的文本大小，并允许所有浏览器缩放或调整文本大小！

## 40.5 响应式字体大小

可以使用 vw 单位设置文本大小，它的意思是“视口宽度”（"viewport width"）。

这样，文本大小将遵循浏览器窗口的大小，请调整浏览器窗口的大小，以查看字体大小如何缩放：

实例

<h1 style="font-size:10vw">Hello World</h1>

视口（Viewport）是浏览器窗口的大小。 1vw = 视口宽度的 1％。如果视口为 50 厘米宽，则 1vw 为 0.5 厘米。

# 第41章 CSS 谷歌字体

## 41.1 谷歌字体

如果您不想使用 HTML 中的任何标准字体，则可以使用 Google Fonts API 向页面添加数百种其他字体。

只需添加一个样式表链接并引用您选择的字体系列：

实例

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">

<style>

body {

font-family: "Sofia";

font-size: 22px;

}

</style>

</head>

<body>

<h1>Sofia Font</h1>

<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>

</body>

</html>

# 第42章 CSS 字体属性

## 42.1 字体属性

为了缩短代码，也可以在一个属性中指定所有单个字体属性。

font 属性是以下属性的简写属性：

font-style

font-variant

font-weight

font-size/line-height

font-family

实例

使用简写声明设置一些字体属性：

p.a {

font: 20px Arial, sans-serif;

}

p.b {

font: italic small-caps bold 12px/30px Georgia, serif;

}

注意：font-size 和 font-family 的值是必需的。如果缺少其他值之一，则会使用其默认值。

## 42.2 所有 CSS 字体属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| font | 简写属性。在一条声明中设置所有字体属性。 |
| font-family | 规定文本的字体系列（字体族）。 |
| font-size | 规定文本的字体大小。 |
| font-style | 规定文本的字体样式。 |
| font-variant | 规定是否以小型大写字母的字体显示文本。 |
| font-weight | 规定字体的粗细。 |

# 第43章 CSS 图标

## 43.1 如何添加图标

向 HTML 页面添加图标的最简单方法是使用图标库，比如 Font Awesome。

将指定的图标类的名称添加到任何行内 HTML 元素（如 <i> 或 <span>）。

下面的图标库中的所有图标都是可缩放矢量，可以使用 CSS进行自定义（大小、颜色、阴影等）。

## 43.2 Font Awesome 图标

如需使用 Font Awesome 图标，请访问 fontawesome.com，登录并获取代码添加到 HTML 页面的 <head> 部分：

<script src="https://kit.fontawesome.com/yourcode.js"></script>

请在我们的 Font Awesome 5 教程中，阅读有关如何开始使用 Font Awesome 的更多内容。

提示：无需下载或安装！

实例

<!DOCTYPE html>

<html>

<head>

<script src="https://kit.fontawesome.com/a076d05399.js"></script>

</head>

<body>

<i class="fas fa-cloud"></i>

<i class="fas fa-heart"></i>

<i class="fas fa-car"></i>

<i class="fas fa-file"></i>

<i class="fas fa-bars"></i>

</body>

</html>

## 43.3 Bootstrap 图标

如需使用 Bootstrap glyphicons，请在 HTML 页面的 <head> 部分内添加这行：

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

提示：无需下载或安装！

实例

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

</head>

<body>

<i class="glyphicon glyphicon-cloud"></i>

<i class="glyphicon glyphicon-remove"></i>

<i class="glyphicon glyphicon-user"></i>

<i class="glyphicon glyphicon-envelope"></i>

<i class="glyphicon glyphicon-thumbs-up"></i>

</body>

</html>

## 43.4 Google 图标

如需使用 Google 图标，请在HTML页面的 <head> 部分中添加以下行：

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

提示：无需下载或安装！

实例

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

</head>

<body>

<i class="material-icons">cloud</i>

<i class="material-icons">favorite</i>

<i class="material-icons">attachment</i>

<i class="material-icons">computer</i>

<i class="material-icons">traffic</i>

</body>

</html>

# 第44章 CSS 链接

## 44.1 设置链接样式

链接可以使用任何 CSS 属性（例如 color、font-family、background 等）来设置样式。

实例

a {

color: hotpink;

}

此外，可以根据链接处于什么状态来设置链接的不同样式。

四种链接状态分别是：

a:link - 正常的，未访问的链接

a:visited - 用户访问过的链接

a:hover - 用户将鼠标悬停在链接上时

a:active - 链接被点击时

实例

/\* 未被访问的链接 \*/

a:link {

color: red;

}

/\* 已被访问的链接 \*/

a:visited {

color: green;

}

/\* 将鼠标悬停在链接上 \*/

a:hover {

color: hotpink;

}

/\* 被选择的链接 \*/

a:active {

color: blue;

}

如果为多个链接状态设置样式，请遵循如下顺序规则：

a:hover 必须 a:link 和 a:visited 之后

a:active 必须在 a:hover 之后

## 44.2 文本装饰

text-decoration 属性主要用于从链接中删除下划线：

实例

a:link {

text-decoration: none;

}

a:visited {

text-decoration: none;

}

a:hover {

text-decoration: underline;

}

a:active {

text-decoration: underline;

}

## 44.3 背景色

background-color 属性可用于指定链接的背景色：

实例

a:link {

background-color: yellow;

}

a:visited {

background-color: cyan;

}

a:hover {

background-color: lightgreen;

}

a:active {

background-color: hotpink;

}

## 44.4 链接按钮

本例演示了一个更高级的例子，其中我们组合了多个 CSS 属性，将链接显示为框/按钮：

实例

a:link, a:visited {

background-color: #f44336;

color: white;

padding: 14px 25px;

text-align: center;

text-decoration: none;

display: inline-block;

}

a:hover, a:active {

background-color: red;

}

## 44.5 为超链接添加不同的样式

<!DOCTYPE html>

<html>

<head>

<style>

a.one:link {color:#ff0000;}

a.one:visited {color:#0000ff;}

a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}

a.two:visited {color:#0000ff;}

a.two:hover {font-size:150%;}

a.three:link {color:#ff0000;}

a.three:visited {color:#0000ff;}

a.three:hover {background:#66ff66;}

a.four:link {color:#ff0000;}

a.four:visited {color:#0000ff;}

a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}

a.five:visited {color:#0000ff;text-decoration:none;}

a.five:hover {text-decoration:underline;}

</style>

</head>

<body>

<p>请把鼠标移到链接上并观察样式的变化：</p>

<p><b><a class="one" href="default.asp" target="\_blank">此链接改变颜色</a></b></p>

<p><b><a class="two" href="default.asp" target="\_blank">此链接改变字体大小</a></b></p>

<p><b><a class="three" href="default.asp" target="\_blank">此链接改变背景色</a></b></p>

<p><b><a class="four" href="default.asp" target="\_blank">此链接改变字体族</a></b></p>

<p><b><a class="five" href="default.asp" target="\_blank">此链接改变文本装饰</a></b></p>

</body>

</html>

## 44.6 高级 - 创建带边框的链接按钮

<!DOCTYPE html>

<html>

<head>

<style>

a:link, a:visited {

background-color: white;

color: black;

border: 2px solid green;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

}

a:hover, a:active {

background-color: green;

color: white;

}

</style>

</head>

<body>

<a href="default.asp" target="\_blank">这是一个链接</a>

</body>

</html>

## 44.7 改变光标

<!DOCTYPE html>

<html>

<body>

<p>请把鼠标移动到单词上，以查看指针效果：</p>

<span style="cursor:auto">auto</span><br>

<span style="cursor:crosshair">crosshair</span><br>

<span style="cursor:default">default</span><br>

<span style="cursor:e-resize">e-resize</span><br>

<span style="cursor:help">help</span><br>

<span style="cursor:move">move</span><br>

<span style="cursor:n-resize">n-resize</span><br>

<span style="cursor:ne-resize">ne-resize</span><br>

<span style="cursor:nw-resize">nw-resize</span><br>

<span style="cursor:pointer">pointer</span><br>

<span style="cursor:progress">progress</span><br>

<span style="cursor:s-resize">s-resize</span><br>

<span style="cursor:se-resize">se-resize</span><br>

<span style="cursor:sw-resize">sw-resize</span><br>

<span style="cursor:text">text</span><br>

<span style="cursor:w-resize">w-resize</span><br>

<span style="cursor:wait">wait</span><br>

</body>

</html>

# 第45章 CSS 列表

## 45.1 HTML 列表和 CSS 列表属性

在 HTML 中，列表主要有两种类型：

无序列表（<ul>）- 列表项用的是项目符号标记

有序列表（<ol>）- 列表项用的是数字或字母标记

CSS 列表属性使您可以：

为有序列表设置不同的列表项标记

为无序列表设置不同的列表项标记

将图像设置为列表项标记

为列表和列表项添加背景色

## 45.2 不同的列表项目标记

list-style-type 属性指定列表项标记的类型。

下例显示了一些可用的列表项标记：

实例

ul.a {

list-style-type: circle;

}

ul.b {

list-style-type: square;

}

ol.c {

list-style-type: upper-roman;

}

ol.d {

list-style-type: lower-alpha;

}

注释：有些值用于无序列表，而有些值用于有序列表。

## 45.3 图像作为列表项标记

list-style-image 属性将图像指定为列表项标记：

实例

ul {

list-style-image: url('sqpurple.gif');

}

## 45.4 定位列表项标记

list-style-position 属性指定列表项标记（项目符号）的位置。

"list-style-position: outside;" 表示项目符号点将在列表项之外。列表项每行的开头将垂直对齐。这是默认的：

"list-style-position: inside;" 表示项目符号将在列表项内。由于它是列表项的一部分，因此它将成为文本的一部分，并在开头推开文本：

ul.a {

list-style-position: outside;

}

ul.b {

list-style-position: inside;

}

## 45.5 删除默认设置

list-style-type:none 属性也可以用于删除标记/项目符号。请注意，列表还拥有默认的外边距和内边距。要删除此内容，请在 <ul> 或 <ol> 中添加 margin:0 和 padding:0 ：

实例

ul {

list-style-type: none;

margin: 0;

padding: 0;

}

## 45.6 列表 - 简写属性

list-style 属性是一种简写属性。它用于在一条声明中设置所有列表属性：

实例

ul {

list-style: square inside url("sqpurple.gif");

}

## 45.7 在使用简写属性时，属性值的顺序为：

list-style-type（如果指定了 list-style-image，那么在由于某种原因而无法显示图像时，会显示这个属性的值）

list-style-position（指定列表项标记应显示在内容流的内部还是外部）

list-style-image（将图像指定为列表项标记）

如果缺少上述属性值之一，则将插入缺失属性的默认值（如果有）。

## 45.8 设置列表的颜色样式

我们还可以使用颜色设置列表样式，使它们看起来更有趣。

添加到 <ol> 或 <ul> 标记的任何样式都会影响整个列表，而添加到 <li> 标记的属性将影响各个列表项：

实例

ol {

background: #ff9999;

padding: 20px;

}

ul {

background: #3399ff;

padding: 20px;

}

ol li {

background: #ffe5e5;

padding: 5px;

margin-left: 35px;

}

ul li {

background: #cce5ff;

margin: 5px;

}

## 45.9 带红色左边框的自定义列表

<!DOCTYPE html>

<html>

<head>

<style>

ul {

border-left: 5px solid red;

background-color: #f1f1f1;

list-style-type: none;

padding: 10px 20px;

}

</style>

</head>

<body>

<p>左边框为红色的列表：</p>

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

</body>

</html>

## 45.10 全屏宽度的边框列表

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

padding: 0;

border: 1px solid #ddd;

}

ul li {

padding: 8px 16px;

border-bottom: 1px solid #ddd;

}

ul li:last-child {

border-bottom: none

}

</style>

</head>

<body>

<p>全宽的带边框的列表：</p>

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

</body>

</html>

## 45.11 列表的所有不同列表项标记

<!DOCTYPE html>

<html>

<head>

<style>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

</style>

</head>

<body>

<ul class="a">

<li>Circle type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<ul class="b">

<li>Disc type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<ul class="c">

<li>Square type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<ol class="d">

<li>Armenian type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="e">

<li>Cjk-ideographic type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="f">

<li>Decimal type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="g">

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

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="h">

<li>Georgian type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="i">

<li>Hebrew type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="j">

<li>Hiragana type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="k">

<li>Hiragana-iroha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="l">

<li>Katakana type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="m">

<li>Katakana-iroha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="n">

<li>Lower-alpha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="o">

<li>Lower-greek type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="p">

<li>Lower-latin type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="q">

<li>Lower-roman type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="r">

<li>Upper-alpha type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="s">

<li>Upper-latin type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="t">

<li>Upper-roman type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="u">

<li>None type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="v">

<li>inherit type</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

</body>

</html>

## 45.12 所有 CSS 列表属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| list-style | 简写属性。在一条声明中设置列表的所有属性。 |
| list-style-image | 指定图像作为列表项标记。 |
| list-style-position | 规定列表项标记（项目符号）的位置。 |
| list-style-type | 规定列表项标记的类型。 |

# 第46章 CSS 表格

使用 CSS 可以极大地改善 HTML 表格的外观：

## 46.1 表格边框

如需在 CSS 中设置表格边框，请使用 border 属性。

下例为 <table>、<th> 和 <td> 元素规定了黑色边框：

table, th, td {

border: 1px solid black;

}

注意：上例中的表格拥有双边框。这是因为 table 和 <th> 和 <td> 元素都有单独的边框。

## 46.2 全宽表格

在某些情况下，上表似乎很小。如果您需要一个可以覆盖整个屏幕（全宽）的表格，请为 <table> 元素添加 width: 100%：

实例

table {

width: 100%;

}

## 46.3 双边框

请注意上面的表格有双边框。这是因为表格和 th、td 元素都有单独的边框。

如需删除双边框，请看下面的例子。

## 46.4 合并表格边框

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;

}

如果只希望表格周围有边框，则仅需为 <table> 指定 border 属性：

table {

border: 1px solid black;

}

## 46.5 表格宽度和高度

表格的宽度和高度由 width 和 height 属性定义。

下例将表的宽度设置为 100％，将 <th> 元素的高度设置为 50px：

table {

width: 100%;

}

th {

height: 50px;

}

要创建仅占页面一半的表，请使用 width: 50%：

实例

table {

width: 50%;

}

th {

height: 70px;

}

## 46.6 水平对齐

text-align 属性设置 <th> 或 <td> 中内容的水平对齐方式（左、右或居中）。

默认情况下，<th> 元素的内容居中对齐，而 <td> 元素的内容左对齐。

要使 <td> 元素的内容也居中对齐，请使用 text-align: center：

th {

text-align: center;

}

下例使 <th> 元素中的文本左对齐：

th {

text-align: left;

}

## 46.7 垂直对齐

vertical-align 属性设置 <th> 或 <td> 中内容的垂直对齐方式（上、下或居中）。

默认情况下，表中内容的垂直对齐是居中（<th> 和 <td> 元素都是）。

下例将 <td> 元素的垂直文本对齐方式设置为下对齐：

td {

height: 50px;

vertical-align: bottom;

}

## 46.8 表格内边距

如需控制边框和表格内容之间的间距，请在 <td> 和 <th> 元素上使用 padding 属性：

th, td {

padding: 15px;

text-align: left;

}

## 46.9 水平分隔线

向 <th> 和 <td> 添加 border-bottom 属性，以实现水平分隔线：

实例

th, td {

border-bottom: 1px solid #ddd;

}

## 46.10 可悬停表格

在 <tr> 元素上使用 :hover 选择器，以突出显示鼠标悬停时的表格行：

tr:hover {background-color: #f5f5f5;}

## 46.11 条状表格

为了实现斑马纹表格效果，请使用 nth-child() 选择器，并为所有偶数（或奇数）表行添加 background-color：

实例

tr:nth-child(even) {background-color: #f2f2f2;}

## 46.12 表格颜色

下例指定了 <th> 元素的背景颜色和文本颜色：

th {

background-color: #4CAF50;

color: white;

}

## 46.13 响应式表格

如果屏幕太小而无法显示全部内容，则响应式表格会显示水平滚动条：

在 <table> 元素周围添加带有 overflow-x:auto 的容器元素（例如 <div>），以实现响应式效果：

实例

<!DOCTYPE html>

<html>

<head>

<style>

table {

border-collapse: collapse;

width: 100%;

}

th, td {

text-align: left;

padding: 8px;

}

tr:nth-child(even) {background-color: #f2f2f2;}

</style>

</head>

<body>

<h1>响应式表格</h1>

<p>如果屏幕太小无法显示全部内容，响应表将显示水平滚动条。请调整浏览器窗口的大小以查看效果：</p>

<p>如需创建响应式表格，请用 <strong>overflow-x:auto</strong> 的容器元素（比如 div）包围表格元素：</p>

<div style="overflow-x:auto;">

<table>

<tr>

<th>First Name</th>

<th>Last Name</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

<th>Points</th>

</tr>

<tr>

<td>Bill</td>

<td>Gates</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

</tr>

<tr>

<td>Steve</td>

<td>Jobs</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

</tr>

<tr>

<td>Elon</td>

<td>Musk</td>

<td>67</td>

<td>67</td>

<td>67</td>

<td>67</td>

<td>67</td>

<td>67</td>

<td>67</td>

<td>67</td>

<td>67</td>

<td>67</td>

</tr>

</table>

</div>

</body>

</html>

注释：在 OS X Lion（在 Mac 上）中，滚动条默认情况下是隐藏的，并且仅在使用时显示（即使设置了 "overflow:scroll"）。

## 46.14 CSS 表格属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| border | 简写属性。在一条声明中设置所有边框属性。 |
| border-collapse | 规定是否应折叠表格边框。 |
| border-spacing | 规定相邻单元格之间的边框的距离。 |
| caption-side | 规定表格标题的位置。 |
| empty-cells | 规定是否在表格中的空白单元格上显示边框和背景。 |
| table-layout | 设置用于表格的布局算法。 |

# 第47章 CSS 布局 - display 属性

display 属性是用于控制布局的最重要的 CSS 属性。

## 47.1 display 属性

display 属性规定是否/如何显示元素。

每个 HTML 元素都有一个默认的 display 值，具体取决于它的元素类型。大多数元素的默认 display 值为 block 或 inline。

## 47.2 块级元素（block element）

块级元素总是从新行开始，并占据可用的全部宽度（尽可能向左和向右伸展）。

块级元素的一些例子：

<div>

<h1> - <h6>

<p>

<form>

<header>

<footer>

<section>

## 47.3 行内元素（inline element）

内联元素不从新行开始，仅占用所需的宽度。

行内元素的一些例子：

<span>

<a>

<img>

## 47.4 Display: none;

display: none; 通常与 JavaScript 一起使用，以隐藏和显示元素，而无需删除和重新创建它们。如果您想知道如何实现此目标，请查看本页面上的最后一个实例。

默认情况下，<script> 元素使用 display: none;。

## 47.5 覆盖默认的 Display 值

如前所述，每个元素都有一个默认 display 值。但是，您可以覆盖它。

将行内元素更改为块元素，反之亦然，对于使页面以特定方式显示同时仍遵循 Web 标准很有用。

一个常见的例子是为实现水平菜单而生成行内的 <li> 元素：

实例

li {

display: inline;

}

注意：设置元素的 display 属性仅会更改元素的显示方式，而不会更改元素的种类。因此，带有 display: block; 的行内元素不允许在其中包含其他块元素。

下例将 <span> 元素显示为块元素：

实例

span {

display: block;

}

下例将 <a> 元素显示为块元素：

实例

a {

display: block;

}

## 47.6 隐藏元素 - display:none 还是 visibility:hidden？

通过将 display 属性设置为 none 可以隐藏元素。该元素将被隐藏，并且页面将显示为好像该元素不在其中：

实例

h1.hidden {

display: none;

}

## 47.7 visibility:hidden; 也可以隐藏元素。

但是，该元素仍将占用与之前相同的空间。元素将被隐藏，但仍会影响布局：

实例

h1.hidden {

visibility: hidden;

}

## 47.8 display: none; 与 visibility: hidden; 之间的差异

<!DOCTYPE html>

<html>

<head>

<style>

.imgbox {

float: left;

text-align: center;

width: 185px;

border: 1px solid gray;

margin: 4px;

padding: 6px;

}

button {

width: 100%;

}

</style>

</head>

<body>

<h1>display:none 与 visiblity: hidden 的区别</h1>

<p><b>visibility:hidden</b> 隐藏元素，但仍占据布局中的空间。</p>

<p><b>display:none</b> 从文档中删除元素。它不会占据任何空间。</p>

<div class="imgbox" id="imgbox1">Box 1<br>

<img src="/i/css/imgbox-1.gif" alt="Box 1" style="width:100%">

<button onclick="removeElement()">删除</button>

</div>

<div class="imgbox" id="imgbox2">Box 2<br>

<img src="/i/css/imgbox-2.gif" alt="Box 2" style="width:100%">

<button onclick="changeVisibility()">隐藏</button>

</div>

<div class="imgbox">Box 3<br>

<img src="/i/css/imgbox-3.gif" alt="Box 3" style="width:100%">

<button onclick="resetElement()">重置所有</button>

</div>

<script>

function removeElement() {

document.getElementById("imgbox1").style.display = "none";

}

function changeVisibility() {

document.getElementById("imgbox2").style.visibility = "hidden";

}

function resetElement() {

document.getElementById("imgbox1").style.display = "block";

document.getElementById("imgbox2").style.visibility = "visible";

}

</script>

</body>

</html>

## 47.9 结合使用 CSS 和 JavaScript 来显示内容

<!DOCTYPE html>

<html>

<head>

<style>

#panel, .flip {

font-size: 16px;

padding: 10px;

text-align: center;

background-color: #4CAF50;

color: white;

border: solid 1px #a6d8a8;

margin: auto;

}

#panel {

display: none;

}

</style>

</head>

<body>

<p class="flip" onclick="myFunction()">点击这里来显示面板</p>

<div id="panel">

<p>该面板包含一个 div 元素，默认情况下该元素是隐藏的（display: none）。</p>

<p>它使用 CSS 进行样式设置，我们使用 JavaScript 来显示它（display: block）。</p>

<p>工作原理：请注意，带有 class="flip" 的 p 元素有 onclick 属性。当用户单击 p 元素时，将执行一个名为 myFunction() 的函数，该函数将 id="panel" 的 div 样式从 display:none（隐藏）更改为 display:block（可见）。</p>

<p>您将在我们的 JavaScript 教程中学到有关 JavaScript 的更多知识。</p>

</div>

<script>

function myFunction() {

document.getElementById("panel").style.display = "block";

}

</script>

</body>

</html>

## 47.10 CSS Display/Visibility 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| display | 指定应如何显示元素。 |
| visibility | 指定元素是否应该可见。 |

# 第48章 CSS 布局 - width 和 max-width

使用 width、max-width 和 margin: auto;

如上一章所述，块级元素始终占用可用的全部宽度（尽可能向左和向右伸展）。

设置块级元素的 width 将防止其延伸到其容器的边缘。然后，您可以将外边距设置为 auto，以将元素在其容器中水平居中。元素将占用指定的宽度，剩余空间将在两个外边距之间平均分配：

注意：当浏览器窗口小于元素的宽度时，上面这个 <div> 会发生问题。浏览器会将水平滚动条添加到页面。

在这种情况下，使用 max-width 可以改善浏览器对小窗口的处理。为了使网站在小型设备上可用，这一点很重要：

提示：请将浏览器窗口的大小调整为小于 500 像素，以查看两个 div 之间的区别！

这是上面两个 div 的例子：

div.ex1 {

width: 500px;

margin: auto;

border: 3px solid #73AD21;

}

div.ex2 {

max-width: 500px;

margin: auto;

border: 3px solid #73AD21;

}

# 第49章 CSS 布局 - position 属性

position 属性规定应用于元素的定位方法的类型（static、relative、fixed、absolute 或 sticky）。

## 49.1 position 属性

position 属性规定应用于元素的定位方法的类型。

有五个不同的位置值：

static

relative

fixed

absolute

sticky

元素其实是使用 top、bottom、left 和 right 属性定位的。但是，除非首先设置了 position 属性，否则这些属性将不起作用。根据不同的 position 值，它们的工作方式也不同。

## 49.2 position: static;

HTML 元素默认情况下的定位方式为 static（静态）。

静态定位的元素不受 top、bottom、left 和 right 属性的影响。

position: static; 的元素不会以任何特殊方式定位；它始终根据页面的正常流进行定位：

这是所用的 CSS：

实例

div.static {

position: static;

border: 3px solid #73AD21;

}

## 49.3 position: relative;

position: relative; 的元素相对于其正常位置进行定位。

设置相对定位的元素的 top、right、bottom 和 left 属性将导致其偏离其正常位置进行调整。不会对其余内容进行调整来适应元素留下的任何空间。

这是所用的 CSS：

实例

div.relative {

position: relative;

left: 30px;

border: 3px solid #73AD21;

}

## 49.4 position: fixed;

position: fixed; 的元素是相对于视口定位的，这意味着即使滚动页面，它也始终位于同一位置。 top、right、bottom 和 left 属性用于定位此元素。

固定定位的元素不会在页面中通常应放置的位置上留出空隙。

请注意页面右下角的这个固定元素。这是所用的 CSS：

实例

div.fixed {

position: fixed;

bottom: 0;

right: 0;

width: 300px;

border: 3px solid #73AD21;

}

## 49.5 position: absolute;

position: absolute; 的元素相对于最近的定位祖先元素进行定位（而不是相对于视口定位，如 fixed）。

然而，如果绝对定位的元素没有祖先，它将使用文档主体（body），并随页面滚动一起移动。

注意：“被定位的”元素是其位置除 static 以外的任何元素。

这是一个简单的例子：

这是所用的 CSS：

实例

div.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

## 49.6 position: sticky;

position: sticky; 的元素根据用户的滚动位置进行定位。

粘性元素根据滚动位置在相对（relative）和固定（fixed）之间切换。起先它会被相对定位，直到在视口中遇到给定的偏移位置为止 - 然后将其“粘贴”在适当的位置（比如 position:fixed）。

注意：Internet Explorer、Edge 15 以及更早的版本不支持粘性定位。 Safari 需要 -webkit- 前缀（请参见下面的实例）。您还必须至少指定 top、right、bottom 或 left 之一，以便粘性定位起作用。

在此例中，在到达其滚动位置时，sticky 元素将停留在页面顶部（top: 0）。

实例

div.sticky {

position: -webkit-sticky; /\* Safari \*/

position: sticky;

top: 0;

background-color: green;

border: 2px solid #4CAF50;

}

## 49.7 重叠元素

在对元素进行定位时，它们可以与其他元素重叠。

z-index 属性指定元素的堆栈顺序（哪个元素应放置在其他元素的前面或后面）。

元素可以设置正或负的堆叠顺序：

实例

img {

position: absolute;

left: 0px;

top: 0px;

z-index: -1;

}

具有较高堆叠顺序的元素始终位于具有较低堆叠顺序的元素之前。

注意：如果两个定位的元素重叠而未指定 z-index，则位于 HTML 代码中最后的元素将显示在顶部。

## 49.8 所有 CSS 定位属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| bottom | 设置定位框的底部外边距边缘。 |
| clip | 剪裁绝对定位的元素。 |
| left | 设置定位框的左侧外边距边缘。 |
| position | 规定元素的定位类型。 |
| right | 设置定位框的右侧外边距边缘。 |
| top | 设置定位框的顶部外边距边缘。 |
| z-index | 设置元素的堆叠顺序。 |

# 第50章 CSS 布局 - 溢出

CSS overflow 属性控制对太大而区域无法容纳的内容的处理方式。

## 50.1 CSS Overflow

overflow 属性指定在元素的内容太大而无法放入指定区域时是剪裁内容还是添加滚动条。

overflow 属性可设置以下值：

visible - 默认。溢出没有被剪裁。内容在元素框外渲染

hidden - 溢出被剪裁，其余内容将不可见

scroll - 溢出被剪裁，同时添加滚动条以查看其余内容

auto - 与 scroll 类似，但仅在必要时添加滚动条

注释：overflow 属性仅适用于具有指定高度的块元素。

注释：在 OS X Lion（在 Mac 上）中，滚动条默认情况下是隐藏的，并且仅在使用时显示（即使设置了 "overflow:scroll"）。

## 50.2 overflow: visible

默认情况下，溢出是可见的(visible)，这意味着它不会被裁剪，而是在元素框外渲染：

实例

div {

width: 200px;

height: 50px;

background-color: #eee;

overflow: visible;

}

## 50.3 overflow: hidden

如果使用 hidden 值，溢出会被裁剪，其余内容被隐藏：

实例

div {

overflow: hidden;

}

## 50.4 overflow: scroll

如果将值设置为 scroll，溢出将被裁剪，并添加滚动条以便在框内滚动。请注意，这将在水平和垂直方向上添加一个滚动条（即使您不需要它）：

实例

div {

overflow: scroll;

}

## 50.5 overflow: auto

auto 值类似于 scroll，但是它仅在必要时添加滚动条：

实例

div {

overflow: auto;

}

## 50.6 overflow-x 和 overflow-y

overflow-x 和 overflow-y 属性规定是仅水平还是垂直地（或同时）更改内容的溢出：

overflow-x 指定如何处理内容的左/右边缘。

overflow-y 指定如何处理内容的上/下边缘。

实例

div {

overflow-x: hidden; /\* 隐藏水平滚动栏 \*/

overflow-y: scroll; /\* 添加垂直滚动栏 \*/

}

## 50.7 所有 CSS Overflow 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| overflow | 规定如果内容溢出元素框会发生什么情况。 |
| overflow-x | 规定在元素的内容区域溢出时如何处理内容的左/右边缘。 |
| overflow-y | 指定在元素的内容区域溢出时如何处理内容的上/下边缘。 |

# 第51章 CSS 布局 - 浮动和清除

## 51.1 CSS 布局 - 浮动和清除

CSS float 属性规定元素如何浮动。

CSS clear 属性规定哪些元素可以在清除的元素旁边以及在哪一侧浮动。

## 51.2 float 属性

float 属性用于定位和格式化内容，例如让图像向左浮动到容器中的文本那里。

float 属性可以设置以下值之一：

left - 元素浮动到其容器的左侧

right - 元素浮动在其容器的右侧

none - 元素不会浮动（将显示在文本中刚出现的位置）。默认值。

inherit - 元素继承其父级的 float 值

最简单的用法是，float 属性可实现（报纸上）文字包围图片的效果。

## 51.3 实例 - float: right;

下例指定图像应在文本中向右浮动：

Pineapple

领先的 Web 技术教程 - 全部免费。在 W3School，你可以找到你所需要的所有的网站建设教程。从基础的 HTML 到 CSS，乃至进阶的 XML、SQL、JS、PHP。

我们的参考手册涵盖了网站技术的方方面面。其中包括W3C标准技术：HTML、CSS、XML 。以及其他技术，诸如 JavaScript、PHP、SQL 等。

在 W3School，我们提供上千个实例。通过使用我们的在线编辑器，你可以编辑这些例子，并对代码进行实验。

实例

img {

float: right;

}

## 51.4 实例 - float: left;

下例指定图像应在文本中向左浮动：

实例

img {

float: left;

}

## 51.5 实例 - No float

在下例中，图像将显示在文本中刚出现的位置（float: none;）：

实例

img {

float: none;

}

# 第52章 CSS 布局 - clear 和 clearfix

## 52.1 clear 属性

clear 属性指定哪些元素可以浮动于被清除元素的旁边以及哪一侧。

clear 属性可设置以下值之一：

none - 允许两侧都有浮动元素。默认值

left - 左侧不允许浮动元素

right- 右侧不允许浮动元素

both - 左侧或右侧均不允许浮动元素

inherit - 元素继承其父级的 clear 值

使用 clear 属性的最常见用法是在元素上使用了 float 属性之后。

在清除浮动时，应该对清除与浮动进行匹配：如果某个元素浮动到左侧，则应清除左侧。您的浮动元素会继续浮动，但是被清除的元素将显示在其下方。

下例将清除向左的浮动。表示在（div 的）左侧不允许出现浮动元素：

实例

div {

clear: left;

}

## 52.2 clearfix Hack

如果一个元素比包含它的元素高，并且它是浮动的，它将“溢出”到其容器之外：

然后我们可以向包含元素添加 overflow: auto;，来解决此问题：

实例

.clearfix {

overflow: auto;

}

只要您能够控制外边距和内边距（否则您可能会看到滚动条），overflow: auto clearfix 就会很好地工作。但是，新的现代 clearfix hack 技术使用起来更安全，以下代码被应用于多数网站：

实例

.clearfix::after {

content: "";

clear: both;

display: table;

}

您将在稍后的章节学到 ::after 伪元素。

# 第53章 CSS 布局 - 浮动实例

网格 / 等宽的框

通过使用 float 属性，可以轻松地并排浮动内容框：

实例

\* {

box-sizing: border-box;

}

.box {

float: left;

width: 33.33%; /\* 三个框（四框使用 25%，两框使用 50%，以此类推） \*/

padding: 50px; /\* 如果需要在图片间增加间距 \*/

}

什么是 box-sizing？

您可以轻松地并排创建三个浮动框。但是，当您添加一些内容来扩大每个框的宽度（例如，内边距或边框）时，这个框会损坏。 box-sizing 属性允许我们在框的总宽度（和高度）中包括内边距和边框，确保内边距留在框内而不会破裂。

这种框格（The grid of boxes）也可以用来并排显示图像：

实例

.img-container {

float: left;

width: 33.33%; /\* 三个框（四框使用 25%，两框使用 50%，以此类推） \*/

padding: 5px; /\* 如果需要在图片间增加间距 \*/

}

等宽的框

在上例中，您学习了如何以相等的宽度并排浮动框。但是，创建具有相同高度的浮动框并不容易。不过，快速解决方案是设置一个固定的高度，如下例所示：

实例

.box {

height: 500px;

}

但是，这么做就失去了弹性。如果可以保证框中始终有相同数量的内容，那是可以的。但是很多时候，内容是不一样的。如果您在手机上尝试上例，则会看到第二个框的内容将显示在框的外部。这是 CSS3 Flexbox 派上用场的地方 - 因为它可以自动拉伸框使其与最长的框一样长：

实例

使用 Flexbox 创建弹性框：

Flexbox 的唯一问题是它在 Internet Explorer 10 或更早版本中不起作用。您可以在我们的 CSS Flexbox 章节中学习有关 Flexbox 布局模块的更多知识。

导航菜单

将 float 与超链接列表一起使用，来创建水平菜单：

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: inline-block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

.active {

background-color: red;

}

</style>

</head>

<body>

<ul>

<li><a href="#home" class="active">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

Web 布局实例

使用 float 属性完成整个 Web 布局也很常见：

实例

.header, .footer {

background-color: grey;

color: white;

padding: 15px;

}

.column {

float: left;

padding: 15px;

}

.clearfix::after {

content: "";

clear: both;

display: table;

}

.menu {

width: 25%;

}

.content {

width: 75%;

}

带有边框和外边距的图像浮动到段落的右侧

<!DOCTYPE html>

<html>

<head>

<style>

img {

float: right;

border: 1px dotted black;

margin: 0px 0px 15px 20px;

}

</style>

</head>

<body>

<p>

在下面的段落中，图像将向右浮动。黑色的虚线边框被添加到图像。

我们还为图像添加了外边距以将文本推离图像：

图像的上、右外边距为 0px，下外边距为 15px，左外边距为 20px。

</p>

<p><img src="/i/logo/w3logo-1.png" width="188" height="267">

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

</body>

</html>

带标题的图像浮动到右侧

<!DOCTYPE html>

<html>

<head>

<style>

div {

float: right;

width: 120px;

margin: 0 0 15px 20px;

padding: 15px;

border: 1px solid black;

text-align: center;

}

</style>

</head>

<body>

<div>

<img src="/i/logo/w3logo-1.png" width="188" height="267"><br>CSS is fun!

</div>

<p>

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

<p>

在上面的段落中，div 元素的宽度为 120 像素，其中包含图像。

div 元素将向右浮动。外边距被添加到 div 以将文本推离 div。

将边框和内边距（填充）添加到 div 中，以凸显图片和标题。

</p>

</body>

</html>

让段落的第一个字母向左浮动

<!DOCTYPE html>

<html>

<head>

<style>

span {

float: left;

width: 0.7em;

font-size: 400%;

font-family: algerian, courier;

line-height: 80%;

}

</style>

</head>

<body>

<p>

<span>T</span>his is some text.

This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

<p>

在上面的段落中，文本的第一个字母嵌入 span 元素中。

span 元素的宽度是当前字体大小的 0.7 倍。

span 元素的字体大小为 400％（很大），行高为 80％。

span 中字母的字体将是 "Algerian"。

</p>

</body>

</html>

用浮动创建一个网站

<!DOCTYPE html>

<html>

<head>

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0;

}

.header {

background-color: #2196F3;

color: white;

text-align: center;

padding: 15px;

}

.footer {

background-color: #444;

color: white;

padding: 15px;

}

.topmenu {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #777;

}

.topmenu li {

float: left;

}

.topmenu li a {

display: inline-block;

color: white;

text-align: center;

padding: 16px;

text-decoration: none;

}

.topmenu li a:hover {

background-color: #222;

}

.topmenu li a.active {

color: white;

background-color: #4CAF50;

}

.column {

float: left;

padding: 15px;

}

.clearfix::after {

content: "";

clear: both;

display: table;

}

.sidemenu {

width: 25%;

}

.content {

width: 75%;

}

.sidemenu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.sidemenu li a {

margin-bottom: 4px;

display: block;

padding: 8px;

background-color: #eee;

text-decoration: none;

color: #666;

}

.sidemenu li a:hover {

background-color: #555;

color: white;

}

.sidemenu li a.active {

background-color: #008CBA;

color: white;

}

</style>

</head>

<body>

<ul class="topmenu">

<li><a href="#home" class="active">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

<div class="clearfix">

<div class="column sidemenu">

<ul>

<li><a href="#flight">The Flight</a></li>

<li><a href="#city" class="active">The City</a></li>

<li><a href="#island">The Island</a></li>

<li><a href="#food">The Food</a></li>

<li><a href="#people">The People</a></li>

<li><a href="#history">The History</a></li>

<li><a href="#oceans">The Oceans</a></li>

</ul>

</div>

<div class="column content">

<div class="header">

<h1>The City</h1>

</div>

<h1>Shanghai</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!

</p>

<p>You will learn more about responsive web pages in a later chapter.</p>

</div>

</div>

<div class="footer">

<p>Footer Text</p>

</div>

</body>

</html>

所有 CSS 浮动属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| box-sizing | 定义元素的宽度和高度的计算方式：它们是否应包含内边距和边框。 |
| clear | 指定哪些元素可以在被清除的元素旁边以及在哪一侧浮动。 |
| float | 指定元素应如何浮动。 |
| overflow | 指定如果内容溢出元素框会发生什么情况。 |
| overflow-x | 指定当溢出元素的内容区域时，如何处理内容的左/右边缘。 |
| overflow-y | 指定当溢出元素的内容区域时，如何处理内容的上/下边缘。 |

# 第54章 CSS 布局 - display: inline-block

## 54.1 display: inline-block

与 display: inline 相比，主要区别在于 display: inline-block 允许在元素上设置宽度和高度。

同样，如果设置了 display: inline-block，将保留上下外边距/内边距，而 display: inline 则不会。

与 display: block 相比，主要区别在于 display：inline-block 在元素之后不添加换行符，因此该元素可以位于其他元素旁边。

下例展示 display: inline、display: inline-block 以及 display: block 的不同行为：

实例

span.a {

display: inline; /\* span 的默认值 \*/

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

span.b {

display: inline-block;

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

span.c {

display: block;

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

## 54.2 使用 inline-block 来创建导航链接

display 的一种常见用法：inline-block 用于水平而不是垂直地显示列表项。下例创建了一个水平导航链接：

实例

.nav {

background-color: yellow;

list-style-type: none;

text-align: center;

padding: 0;

margin: 0;

}

.nav li {

display: inline-block;

font-size: 20px;

padding: 20px;

}

# 第55章 CSS 布局 - 水平和垂直对齐

## 55.1 居中对齐元素

要使块元素（例如 <div> ）水平居中，请使用 margin: auto;。

设置元素的宽度将防止其延伸到容器的边缘。

然后，元素将占用指定的宽度，剩余空间将在两个外边距之间平均分配：

实例

.center {

margin: auto;

width: 50%;

border: 3px solid green;

padding: 20px;

}

注意：如果未设置 width 属性（或将其设置为 100％），则居中对齐无效。

## 55.2 居中对齐文本

如果仅需在元素内居中文本，请使用 text-align: center;：

实例

.center {

text-align: center;

border: 3px solid green;

}

## 55.3 居中对齐图像

如需居中图像，请将左右外边距设置为 auto，并将其设置为块元素：

实例

img {

display: block;

margin-left: auto;

margin-right: auto;

width: 40%;

}

## 55.4 左和右对齐 - 使用 position

对齐元素的一种方法是使用 position: absolute; :

实例

.right {

position: absolute;

right: 0px;

width: 300px;

border: 3px solid #73AD21;

padding: 20px;

}

注意：绝对定位的元素将从正常流中删除，并可能出现元素重叠。

## 55.5 左和右对齐 - 使用 float

对齐元素的另一种方法是使用 float 属性：

实例

.right {

float: right;

width: 300px;

border: 3px solid #73AD21;

padding: 10px;

}

注意：如果一个元素比包含它的元素高，并且它是浮动的，它将溢出其容器。您可以使用 clearfix hack 来解决此问题（请看下面的例子）。

## 55.6 clearfix Hack

然后我们可以向包含元素添加 overflow: auto;，来解决此问题：

实例

.clearfix {

overflow: auto;

}

## 55.7 垂直对齐 - 使用 padding

有很多方法可以在 CSS 中垂直对齐元素。一个简单的解决方案是使用上下内边距：

实例

.center {

padding: 70px 0;

border: 3px solid green;

}

如需同时垂直和水平对齐，请使用 padding 和 text-align: center;：

实例

.center {

padding: 70px 0;

border: 3px solid green;

text-align: center;

}

## 55.8 垂直对齐 - 使用 line-height

另一个技巧是使用其值等于 height 属性值的 line-height 属性：

实例

.center {

line-height: 200px;

height: 200px;

border: 3px solid green;

text-align: center;

}

/\* 如果有多行文本，请添加如下代码：\*/

.center p {

line-height: 1.5;

display: inline-block;

vertical-align: middle;

}

## 55.9 垂直对齐 - 使用 position 和 transform

如果您的选择不是 padding 和 line-height，则另一种解决方案是使用 position 和 transform 属性：

实例

.center {

height: 200px;

position: relative;

border: 3px solid green;

}

.center p {

margin: 0;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

## 55.10 垂直对齐 - 使用 Flexbox

您还可以使用 flexbox 将内容居中。请注意，IE10 以及更早的版本不支持 flexbox：

实例

.center {

display: flex;

justify-content: center;

align-items: center;

height: 200px;

border: 3px solid green;

}

# 第56章 CSS 组合器

## 56.1 CSS 组合器

组合器是解释选择器之间关系的某种机制。

CSS 选择器可以包含多个简单选择器。在简单选择器之间，我们可以包含一个组合器。

CSS 中有四种不同的组合器：

后代选择器 (空格)

子选择器 (>)

相邻兄弟选择器 (+)

通用兄弟选择器 (~)

## 56.2 后代选择器

后代选择器匹配属于指定元素后代的所有元素。

下面的例子选择 <div> 元素内的所有 <p> 元素：

实例

div p {

background-color: yellow;

}

## 56.3 子选择器

子选择器匹配属于指定元素子元素的所有元素。

下面的例子选择属于 <div> 元素子元素的所有 <p> 元素：

实例

div > p {

background-color: yellow;

}

## 56.4 相邻兄弟选择器

相邻兄弟选择器匹配所有作为指定元素的相邻同级的元素。

兄弟（同级）元素必须具有相同的父元素，“相邻”的意思是“紧随其后”。

下面的例子选择紧随 <div> 元素之后的所有 <p> 元素：

实例

div + p {

background-color: yellow;

}

## 56.5 通用兄弟选择器

通用兄弟选择器匹配属于指定元素的同级元素的所有元素。

下面的例子选择属于 <div> 元素的同级元素的所有 <p> 元素：

实例

div ~ p {

background-color: yellow;

}

## 56.6 所有 CSS 组合选择器

|  |  |  |
| --- | --- | --- |
| 选择器 | 示例 | 示例描述 |
| element element | div p | 选择 <div> 元素内的所有 <p> 元素。 |
| element>element | div > p | 选择其父元素是 <div> 元素的所有 <p> 元素。 |
| element+element | div + p | 选择所有紧随 <div> 元素之后的 <p> 元素。 |
| element1~element2 | p ~ ul | 选择前面有 <p> 元素的每个 <ul> 元素。 |

# 第57章 CSS 伪类

## 57.1 什么是伪类？

伪类用于定义元素的特殊状态。

例如，它可以用于：

设置鼠标悬停在元素上时的样式

为已访问和未访问链接设置不同的样式

设置元素获得焦点时的样式

## 57.2 语法

伪类的语法：

selector:pseudo-class {

property: value;

}

## 57.3 锚伪类

链接能够以不同的方式显示：

实例

/\* 未访问的链接 \*/

a:link {

color: #FF0000;

}

/\* 已访问的链接 \*/

a:visited {

color: #00FF00;

}

/\* 鼠标悬停链接 \*/

a:hover {

color: #FF00FF;

}

/\* 已选择的链接 \*/

a:active {

color: #0000FF;

}

注意：a:hover 必须在 CSS 定义中的 a:link 和 a:visited 之后，才能生效！a:active 必须在 CSS 定义中的 a:hover 之后才能生效！伪类名称对大小写不敏感。

## 57.4 伪类和 CSS 类

伪类可以与 CSS 类结合使用：

当您将鼠标悬停在例子中的链接上时，它会改变颜色：

实例

a.highlight:hover {

color: #ff0000;

}

## 57.5 悬停在 <div> 上

在 <div> 元素上使用 :hover 伪类的实例：

实例

div:hover {

background-color: blue;

}

## 57.6 简单的工具提示悬停

把鼠标悬停到 <div> 元素以显示 <p> 元素（类似工具提示的效果）：

悬停到我上面来显示 <p> 元素。

实例

p {

display: none;

background-color: yellow;

padding: 20px;

}

div:hover p {

display: block;

}

## 57.7 CSS - :first-child 伪类

:first-child 伪类与指定的元素匹配：该元素是另一个元素的第一个子元素。

匹配首个 <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任何 <p> 元素：

实例

p:first-child {

color: blue;

}

## 57.8 匹配所有 <p> 元素中的首个 <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：

实例

p i:first-child {

color: blue;

}

## 57.9 匹配所有首个 <p> 元素中的所有 <i> 元素

在下面的例子中，选择器匹配作为另一个元素的第一个子元素的 <p> 元素中的所有 <i> 元素：

实例

p:first-child i {

color: blue;

}

## 57.10 CSS - :lang 伪类

: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，:lang 为属性为 lang="en" 的 <q> 元素定义引号：

实例

<html>

<head>

<style>

q:lang(en) {

quotes: "~" "~";

}

</style>

</head>

<body>

<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>

</body>

</html>

## 57.11 为超链接添加不同样式

<!DOCTYPE html>

<html>

<head>

<style>

a.one:link {color:#ff0000;}

a.one:visited {color:#0000ff;}

a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}

a.two:visited {color:#0000ff;}

a.two:hover {font-size:150%;}

a.three:link {color:#ff0000;}

a.three:visited {color:#0000ff;}

a.three:hover {background:#66ff66;}

a.four:link {color:#ff0000;}

a.four:visited {color:#0000ff;}

a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}

a.five:visited {color:#0000ff;text-decoration:none;}

a.five:hover {text-decoration:underline;}

</style>

</head>

<body>

<p>请把鼠标移到链接上并观察样式的变化：</p>

<p><b><a class="one" href="default.asp" target="\_blank">此链接改变颜色</a></b></p>

<p><b><a class="two" href="default.asp" target="\_blank">此链接改变字体大小</a></b></p>

<p><b><a class="three" href="default.asp" target="\_blank">此链接改变背景色</a></b></p>

<p><b><a class="four" href="default.asp" target="\_blank">此链接改变字体族</a></b></p>

<p><b><a class="five" href="default.asp" target="\_blank">此链接改变文本装饰</a></b></p>

</body>

</html>

## 57.12 使用 :focus

<!DOCTYPE html>

<html>

<head>

<style>

input:focus {

background-color: yellow;

}

</style>

</head>

<body>

<form action="/action\_page.php" method="get">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

## 57.13 所有 CSS 伪类

|  |  |  |
| --- | --- | --- |
| 选择器 | 例子 | 例子描述 |
| :active | a:active | 选择活动的链接。 |
| :checked | input:checked | 选择每个被选中的 <input> 元素。 |
| :disabled | input:disabled | 选择每个被禁用的 <input> 元素。 |
| :empty | p:empty | 选择没有子元素的每个 <p> 元素。 |
| :enabled | input:enabled | 选择每个已启用的 <input> 元素。 |
| :first-child | p:first-child | 选择作为其父的首个子元素的每个 <p> 元素。 |
| :first-of-type | p:first-of-type | 选择作为其父的首个 <p> 元素的每个 <p> 元素。 |
| :focus | input:focus | 选择获得焦点的 <input> 元素。 |
| :hover | a:hover | 选择鼠标悬停其上的链接。 |
| :in-range | input:in-range | 选择具有指定范围内的值的 <input> 元素。 |
| :invalid | input:invalid | 选择所有具有无效值的 <input> 元素。 |
| :lang(language) | p:lang(it) | 选择每个 lang 属性值以 "it" 开头的 <p> 元素。 |
| :last-child | p:last-child | 选择作为其父的最后一个子元素的每个 <p> 元素。 |
| :last-of-type | p:last-of-type | 选择作为其父的最后一个 <p> 元素的每个 <p> 元素。 |
| :link | a:link | 选择所有未被访问的链接。 |
| :not(selector) | :not(p) | 选择每个非 <p> 元素的元素。 |
| :nth-child(n) | p:nth-child(2) | 选择作为其父的第二个子元素的每个 <p> 元素。 |
| :nth-last-child(n) | p:nth-last-child(2) | 选择作为父的第二个子元素的每个<p>元素，从最后一个子元素计数。 |
| :nth-last-of-type(n) | p:nth-last-of-type(2) | 选择作为父的第二个<p>元素的每个<p>元素，从最后一个子元素计数 |
| :nth-of-type(n) | p:nth-of-type(2) | 选择作为其父的第二个 <p> 元素的每个 <p> 元素。 |
| :only-of-type | p:only-of-type | 选择作为其父的唯一 <p> 元素的每个 <p> 元素。 |
| :only-child | p:only-child | 选择作为其父的唯一子元素的 <p> 元素。 |
| :optional | input:optional | 选择不带 "required" 属性的 <input> 元素。 |
| :out-of-range | input:out-of-range | 选择值在指定范围之外的 <input> 元素。 |
| :read-only | input:read-only | 选择指定了 "readonly" 属性的 <input> 元素。 |
| :read-write | input:read-write | 选择不带 "readonly" 属性的 <input> 元素。 |
| :required | input:required | 选择指定了 "required" 属性的 <input> 元素。 |
| :root | root | 选择元素的根元素。 |
| :target | #news:target | 选择当前活动的 #news 元素（单击包含该锚名称的 URL）。 |
| :valid | input:valid | 选择所有具有有效值的 <input> 元素。 |
| :visited | a:visited | 选择所有已访问的链接。 |

## 57.14 所有 CSS 伪元素

|  |  |  |
| --- | --- | --- |
| 选择器 | 例子 | 例子描述 |
| ::after | p::after | 在每个 <p> 元素之后插入内容。 |
| ::before | p::before | 在每个 <p> 元素之前插入内容。 |
| ::first-letter | p::first-letter | 选择每个 <p> 元素的首字母。 |
| ::first-line | p::first-line | 选择每个 <p> 元素的首行。 |
| ::selection | p::selection | 选择用户选择的元素部分。 |

# 第58章 CSS 伪元素

## 58.1 什么是伪元素？

CSS 伪元素用于设置元素指定部分的样式。

例如，它可用于：

设置元素的首字母、首行的样式

在元素的内容之前或之后插入内容

## 58.2 语法

伪元素的语法：

selector::pseudo-element {

property: value;

}

## 58.3 ::first-line 伪元素

::first-line 伪元素用于向文本的首行添加特殊样式。

下面的例子为所有 <p> 元素中的首行添加样式：

实例

p::first-line {

color: #ff0000;

font-variant: small-caps;

}

注意：::first-line 伪元素只能应用于块级元素。

以下属性适用于 ::first-line 伪元素：

字体属性

颜色属性

背景属性

word-spacing

letter-spacing

text-decoration

vertical-align

text-transform

line-height

clear

请注意双冒号表示法 - ::first-line 对比 :first-line

在 CSS3 中，双冒号取代了伪元素的单冒号表示法。这是 W3C 试图区分伪类和伪元素的尝试。

在 CSS2 和 CSS1 中，伪类和伪元素都使用了单冒号语法。

为了向后兼容，CSS2 和 CSS1 伪元素可接受单冒号语法。

## 58.4 ::first-letter 伪元素

::first-letter 伪元素用于向文本的首字母添加特殊样式。

下面的例子设置所有 <p> 元素中文本的首字母格式：

实例

p::first-letter {

color: #ff0000;

font-size: xx-large;

}

注意：::first-letter 伪元素只适用于块级元素。

下面的属性适用于 ::first-letter 伪元素：

字体属性

颜色属性

背景属性

外边距属性

内边距属性

边框属性

text-decoration

vertical-align（仅当 "float" 为 "none"）

text-transform

line-height

float

Clear

## 58.5 伪元素和 CSS 类

伪元素可以与 CSS 类结合使用：

实例

p.intro::first-letter {

color: #ff0000;

font-size: 200%;

}

## 58.6 多个伪元素

也可以组合几个伪元素。

在下面的例子中，段落的第一个字母将是红色，字体大小为 xx-large。第一行的其余部分将变为蓝色，并使用小型大写字母。该段的其余部分将是默认的字体大小和颜色：

实例

p::first-letter {

color: #ff0000;

font-size: xx-large;

}

p::first-line {

color: #0000ff;

font-variant: small-caps;

}

## 58.7 CSS - ::before 伪元素

::before 伪元素可用于在元素内容之前插入一些内容。

下面的例子在每个 <h1> 元素的内容之前插入一幅图像：

实例

h1::before {

content: url(smiley.gif);

}

## 58.8 CSS - ::after 伪元素

::after 伪元素可用于在元素内容之后插入一些内容。

下面的例子在每个 <h1> 元素的内容之后插入一幅图像：

实例

h1::after {

content: url(smiley.gif);

}

## 58.9 CSS - ::selection 伪元素

::selection 伪元素匹配用户选择的元素部分。

以下 CSS 属性可以应用于 ::selection：

color

background

cursor

outline

下例使所选文本在黄色背景上显示为红色：

实例

::selection {

color: red;

background: yellow;

}

## 58.10 所有 CSS 伪元素

|  |  |  |
| --- | --- | --- |
| 选择器 | 例子 | 例子描述 |
| ::after | p::after | 在每个 <p> 元素之后插入内容。 |
| ::before | p::before | 在每个 <p> 元素之前插入内容。 |
| ::first-letter | p::first-letter | 选择每个 <p> 元素的首字母。 |
| ::first-line | p::first-line | 选择每个 <p> 元素的首行。 |
| ::selection | p::selection | 选择用户选择的元素部分。 |

58.11 所有 CSS 伪类

|  |  |  |
| --- | --- | --- |
| 选择器 | 例子 | 例子描述 |
| :active | a:active | 选择活动的链接。 |
| :checked | input:checked | 选择每个被选中的 <input> 元素。 |
| :disabled | input:disabled | 选择每个被禁用的 <input> 元素。 |
| :empty | p:empty | 选择没有子元素的每个 <p> 元素。 |
| :enabled | input:enabled | 选择每个已启用的 <input> 元素。 |
| :first-child | p:first-child | 选择作为其父的首个子元素的每个 <p> 元素。 |
| :first-of-type | p:first-of-type | 选择作为其父的首个 <p> 元素的每个 <p> 元素。 |
| :focus | input:focus | 选择获得焦点的 <input> 元素。 |
| :hover | a:hover | 选择鼠标悬停其上的链接。 |
| :in-range | input:in-range | 选择具有指定范围内的值的 <input> 元素。 |
| :invalid | input:invalid | 选择所有具有无效值的 <input> 元素。 |
| :lang(language) | p:lang(it) | 选择每个 lang 属性值以 "it" 开头的 <p> 元素。 |
| :last-child | p:last-child | 选择作为其父的最后一个子元素的每个 <p> 元素。 |
| :last-of-type | p:last-of-type | 选择作为其父的最后一个 <p> 元素的每个 <p> 元素。 |
| :link | a:link | 选择所有未被访问的链接。 |
| :not(selector) | :not(p) | 选择每个非 <p> 元素的元素。 |
| :nth-child(n) | p:nth-child(2) | 选择作为其父的第二个子元素的每个 <p> 元素。 |
| :nth-last-child(n) | p:nth-last-child(2) | 选择作为父的第二个子元素的每个<p>元素，从最后一个子元素计数。 |
| :nth-last-of-type(n) | p:nth-last-of-type(2) | 选择作为父的第二个<p>元素的每个<p>元素，从最后一个子元素计数 |
| :nth-of-type(n) | p:nth-of-type(2) | 选择作为其父的第二个 <p> 元素的每个 <p> 元素。 |
| :only-of-type | p:only-of-type | 选择作为其父的唯一 <p> 元素的每个 <p> 元素。 |
| :only-child | p:only-child | 选择作为其父的唯一子元素的 <p> 元素。 |
| :optional | input:optional | 选择不带 "required" 属性的 <input> 元素。 |
| :out-of-range | input:out-of-range | 选择值在指定范围之外的 <input> 元素。 |
| :read-only | input:read-only | 选择指定了 "readonly" 属性的 <input> 元素。 |
| :read-write | input:read-write | 选择不带 "readonly" 属性的 <input> 元素。 |
| :required | input:required | 选择指定了 "required" 属性的 <input> 元素。 |
| :root | root | 选择元素的根元素。 |
| :target | #news:target | 选择当前活动的 #news 元素（单击包含该锚名称的 URL）。 |
| :valid | input:valid | 选择所有具有有效值的 <input> 元素。 |
| :visited | a:visited | 选择所有已访问的链接。 |

# 第59章 CSS 不透明度 / 透明度

opacity 属性指定元素的不透明度/透明度。

## 59.1 透明图像

opacity 属性的取值范围为 0.0-1.0。值越低，越透明：

实例

img {

opacity: 0.5;

}

## 59.2 透明悬停效果

opacity 属性通常与 :hover 选择器一同使用，这样就可以在鼠标悬停时更改不透明度：

实例

img {

opacity: 0.5;

}

img:hover {

opacity: 1.0;

}

例子解释

第一个 CSS 块类似于实例 1 中的代码。此外，我们还添加了当用户将鼠标悬停在其中一个图像上时的效果。在这种情况下，当用户将鼠标悬停在图像上时，我们希望图像不透明。这条 CSS 是 opacity:1;。

当鼠标指针离开图像时，图像将再次透明。

反向悬停效果的例子：

实例

img:hover {

opacity: 0.5;

}

## 59.3 透明盒

使用 opacity 属性为元素的背景添加透明度时，其所有子元素都继承相同的透明度。这可能会使完全透明的元素内的文本难以阅读：

div {

opacity: 0.3;

}

## 59.4 使用 RGBA 的透明度

如果您不希望对子元素应用不透明度，如上面的例子，请使用 RGBA 颜色值。下面的例子设置背景色而不是文本的不透明度：

您已经从我们的 CSS 颜色 这一章中学到了可以将 RGB 用作颜色值。除 RGB 外，还可以将 RGB 颜色值与 alpha 通道（RGBA）一起使用 - 该通道规定颜色的不透明度。

RGBA 颜色值指定为：rgba(red, green, blue, alpha)。 alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字。

提示：您将在我们的 CSS 颜色 这一章中学到有关 RGBA 颜色的更多知识。

实例

div {

background: rgba(76, 175, 80, 0.3) /\* Green background with 30% opacity \*/

}

## 59.5 透明盒中的文本

<html>

<head>

<style>

div.background {

background: url(klematis.jpg) repeat;

border: 2px solid black;

}

div.transbox {

margin: 30px;

background-color: #ffffff;

border: 1px solid black;

opacity: 0.6;

}

div.transbox p {

margin: 5%;

font-weight: bold;

color: #000000;

}

</style>

</head>

<body>

<div class="background">

<div class="transbox">

<p>这是一些位于透明框中的文本。</p>

</div>

</div>

</body>

</html>

例子解释

首先，我们创建一个带有背景图像和边框的 <div> 元素（class="background"）。

然后，我们在第一个 <div> 中创建另一个 <div>（class="transbox"）。

<div class="transbox"> 有背景色和边框 - 这个 div 是透明的。

在透明的 <div> 内，我们在 <p> 元素内添加了一些文本。

# 第60章 CSS 导航栏

## 60.1 导航栏

易用的导航对于任何网站都很重要。

通过使用 CSS，您可以将无聊的 HTML 菜单转换为美观的导航栏。

## 60.2 导航栏 = 链接列表

导航栏需要标准 HTML 作为基础。

在我们的实例中，将用标准的 HTML 列表构建导航栏。

导航栏基本上就是链接列表，因此使用 <ul> 和 <li> 元素会很有意义：

实例

<ul>

<li><a href="default.asp">Home</a></li>

<li><a href="news.asp">News</a></li>

<li><a href="contact.asp">Contact</a></li>

<li><a href="about.asp">About</a></li>

</ul>

现在，从列表中删除项目符号以及外边距和内边距（填充）：

实例

ul {

list-style-type: none;

margin: 0;

padding: 0;

}

例子解释：

list-style-type: none; - 删除项目符号。导航条不需要列表项标记。

设置 margin: 0; 和 padding: 0; 删除浏览器的默认设置。

上例中的代码是垂直和水平导航栏中使用的标准代码，您将在下一章中学习更多内容。

# 第61章 CSS 垂直导航栏

## 61.1 垂直导航栏

如需构建垂直导航栏，除了上一章中的代码外，还可以在列表中设置 <a> 元素的样式：

实例

li a {

display: block;

width: 60px;

}

例子解释：

display: block; - 将链接显示为块元素可以使整个链接区域都可以被单击（而不仅仅是文本），我们还可以指定宽度（如果需要，还可以指定内边距、外边距、高度等）。

width: 60px; - 默认情况下，块元素会占用全部可用宽度。我们需要指定 60 像素的宽度。

您还可以设置 <ul> 的宽度，并删除 <a> 的宽度，因为当显示为块元素时，它们将占据可用的全部宽度。这将导致与我们之前的例子相同的结果：

实例

ul {

list-style-type: none;

margin: 0;

padding: 0;

width: 60px;

}

li a {

display: block;

}

## 61.2 垂直导航栏实例

创建背景色为灰色的基础垂直导航栏，并在用户将鼠标移到链接上时改变链接的背景色：

实例

ul {

list-style-type: none;

margin: 0;

padding: 0;

width: 200px;

background-color: #f1f1f1;

}

li a {

display: block;

color: #000;

padding: 8px 16px;

text-decoration: none;

}

/\* 鼠标悬停时改变链接颜色 \*/

li a:hover {

background-color: #555;

color: white;

}

## 61.3 活动/当前导航链接

向当前链接添加 "active" 类，以使用户知道他/她在哪个页面上：

.active {

background-color: #4CAF50;

color: white;

}

## 61.4 居中链接以及添加边框

把 text-align:center 添加到 <li> 或 <a>，使链接居中。

将 border 属性添加到 <ul>，在导航栏周围添加边框。如果您还希望在导航栏内添加边框，请为所有 <li> 元素添加 border-bottom，最后一个元素除外：

ul {

border: 1px solid #555;

}

li {

text-align: center;

border-bottom: 1px solid #555;

}

li:last-child {

border-bottom: none;

}

## 61.5 全高固定垂直导航栏

创建全高的“粘性”侧面导航：

ul {

list-style-type: none;

margin: 0;

padding: 0;

width: 25%;

background-color: #f1f1f1;

height: 100%; /\* 全高 \*/

position: fixed; /\* 使它产生粘性，即使在滚动时 \*/

overflow: auto; /\* 如果侧栏的内容太多，则启用滚动条 \*/

}

# 第62章 CSS 水平导航栏

## 62.1 水平导航栏

有两种创建水平导航栏的方法：使用行内或浮动列表项。

## 62.2 行内列表项

构建水平导航栏的一种方法是，除了上一章中的“标准”代码外，还要将 <li> 元素指定为 inline：

实例

li {

display: inline;

}

例子解释：

display: inline; -默认情况下，<li> 元素是块元素。在这里，我们删除每个列表项之前和之后的换行符，这样它们才能显示在一行。

## 62.3 浮动列表项

创建水平导航栏的另一种方法是浮动 <li> 元素，并为导航链接规定布局：

实例

li {

float: left;

}

a {

display: block;

padding: 8px;

background-color: #dddddd;

}

## 62.4 例子解释：

float: left; - 使用 float 使块元素滑动为彼此相邻

display: block; - 将链接显示为块元素可以使整个链接区域都可单击（不仅是文本），而且允许我们指定填充（如果需要，还可以指定高度，宽度，边距等）

padding: 8px; - 使块元素更美观

background-color: #dddddd; - 为每个元素添加灰色背景色

提示：如需全宽的背景色，请将 background-color 添加到 <ul> 而不是每个 <a> 元素：

实例

ul {

background-color: #dddddd;

}

## 62.5 水平导航栏实例

创建具有深色背景色的基础水平导航栏，并在用户将鼠标移到链接上方时改变链接的背景色：

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

## 62.6 活动/当前导航链接

向当前链接添加 "active" 类，这样用户就知道他/她在哪个页面上：

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover:not(.active) {

background-color: #111;

}

.active {

background-color: #4CAF50;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

## 62.7 右对齐链接

通过将列表项向右浮动来右对齐链接（float:right;）：

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover:not(.active) {

background-color: #111;

}

.active {

background-color: #4CAF50;

}

</style>

</head>

<body>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li style="float:right"><a class="active" href="#about">About</a></li>

</ul>

</body>

</html>

## 62.8 边框分隔栏

将 border-right 属性添加到 <li>，以创建链接分隔符：

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

border-right:1px solid #bbb;

}

li:last-child {

border-right: none;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover:not(.active) {

background-color: #111;

}

.active {

background-color: #4CAF50;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li style="float:right"><a href="#about">About</a></li>

</ul>

</body>

</html>

## 62.9 固定的导航栏

使导航栏保持在页面的顶部或底部，即使用户滚动页面也是如此：

**固定在顶部**

<!DOCTYPE html>

<html>

<head>

<style>

body {margin:0;}

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

position: fixed;

top: 0;

width: 100%;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover:not(.active) {

background-color: #111;

}

.active {

background-color: #4CAF50;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

<div style="padding:20px;margin-top:30px;background-color:#1abc9c;height:1500px;">

<h1>固定的顶部导航栏</h1>

<h2>请滚动页面以查看效果。</h2>

<h2>页面滚动时导航栏将位于页面顶部。</h2>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

</div>

</body>

</html>

**固定在底部**

<!DOCTYPE html>

<html>

<head>

<style>

body {margin:0;}

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

position: fixed;

bottom: 0;

width: 100%;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover:not(.active) {

background-color: #111;

}

.active {

background-color: #4CAF50;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

<div style="padding:20px;background-color:#1abc9c;height:1500px;">

<h1>固定的底部导航栏</h1>

<h2>请滚动页面以查看效果。</h2>

<h2>页面滚动时导航栏将位于页面底部。</h2>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

<p>Some text some text some text some text..</p>

</div>

</body>

</html>

注意：固定定位在移动设备上可能无法正常工作。

## 62.10 灰色水平导航栏

带有细灰色边框的灰色水平导航栏的实例

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

border: 1px solid #e7e7e7;

background-color: #f3f3f3;

}

li {

float: left;

}

li a {

display: block;

color: #666;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover:not(.active) {

background-color: #ddd;

}

li a.active {

color: white;

background-color: #4CAF50;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

## 62.11 粘性导航栏

为 <ul> 添加 position: sticky;，以创建粘性导航栏。

粘性元素会根据滚动位置在相对和固定之间切换。它是相对定位的，直到在视口中遇到给定的偏移位置为止 - 然后将其“粘贴”在适当的位置（比如 position:fixed）。

<!DOCTYPE html>

<html>

<head>

<style>

body {

font-size: 28px;

}

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

position: -webkit-sticky; /\* Safari \*/

position: sticky;

top: 0;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

.active {

background-color: #4CAF50;

}

</style>

</head>

<body>

<div class="header">

<h1>向下滚动</h1>

<p>请向下滚动以查看粘性效果。</p>

</div>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

<h2>粘性导航栏实例</h2>

<p>如果导航栏到了滚动位置，它会<strong>粘</strong>到顶部。</p>

<p><b>注释：</b>Internet Explorer 不支持粘性定位并且 Safari 需要 -webkit- 前缀。</p>

<p>Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>

<p>Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>

<p>Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>

<p>Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>

<p>Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>

<p>Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>

<p>Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>

</body>

</html>

注意：Internet Explorer、Edge 15 和更早版本不支持粘性定位。 Safari 需要 -webkit- 前缀（请参见上面的例子）。您还必须指定 top、right、bottom 或 left 至少之一，以使粘性定位起作用。

## 62.12 响应式的上导航栏

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {margin: 0;}

ul.topnav {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

ul.topnav li {float: left;}

ul.topnav li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

ul.topnav li a:hover:not(.active) {background-color: #111;}

ul.topnav li a.active {background-color: #4CAF50;}

ul.topnav li.right {float: right;}

@media screen and (max-width: 600px) {

ul.topnav li.right,

ul.topnav li {float: none;}

}

</style>

</head>

<body>

<ul class="topnav">

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li class="right"><a href="#about">About</a></li>

</ul>

<div style="padding:0 16px;">

<h1>响应式顶部导航栏实例</h1>

<p>此示例使用媒体查询在屏幕尺寸小于或等于 600 像素时垂直堆叠 topnav。</p>

<p>您稍后将在我们的 CSS 教程中学到有关媒体查询和响应式 Web 设计的更多知识。</p>

<p><b>请调整浏览器窗口的大小以查看效果。</b></p>

</div>

</body>

</html>

## 62.13 响应式的侧导航栏

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {margin: 0;}

ul.sidenav {

list-style-type: none;

margin: 0;

padding: 0;

width: 25%;

background-color: #f1f1f1;

position: fixed;

height: 100%;

overflow: auto;

}

ul.sidenav li a {

display: block;

color: #000;

padding: 8px 16px;

text-decoration: none;

}

ul.sidenav li a.active {

background-color: #4CAF50;

color: white;

}

ul.sidenav li a:hover:not(.active) {

background-color: #555;

color: white;

}

div.content {

margin-left: 25%;

padding: 1px 16px;

height: 1000px;

}

@media screen and (max-width: 900px) {

ul.sidenav {

width: 100%;

height: auto;

position: relative;

}

ul.sidenav li a {

float: left;

padding: 15px;

}

div.content {margin-left: 0;}

}

@media screen and (max-width: 400px) {

ul.sidenav li a {

text-align: center;

float: none;

}

}

</style>

</head>

<body>

<ul class="sidenav">

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

<div class="content">

<h1>响应式侧导航栏实例</h1>

<p>当屏幕尺寸为 900 像素或更小时，此例使用媒体查询将 sidenav 转换为顶部导航栏。</p>

<p>我们还为屏幕小于等于 400 像素的屏幕添加了媒体查询，它将垂直堆叠并居中放置导航链接。</p>

<p>您稍后将在我们的 CSS 教程中学到有关媒体查询和响应式 Web 设计的更多知识。</p>

<p><b>请调整浏览器窗口的大小以查看效果。</b></p>

</div>

</body>

</html>

## 62.14 下拉式导航栏

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a, .dropbtn {

display: inline-block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover, .dropdown:hover .dropbtn {

background-color: red;

}

li.dropdown {

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

text-align: left;

}

.dropdown-content a:hover {background-color: #f1f1f1;}

.dropdown:hover .dropdown-content {

display: block;

}

</style>

</head>

<body>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li class="dropdown">

<a href="javascript:void(0)" class="dropbtn">Dropdown</a>

<div class="dropdown-content">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</li>

</ul>

<h1>导航栏内的下拉菜单</h1>

<p>请悬停到 "Dropdown" 链接上，以查看下拉菜单。</p>

</body>

</html>

# 第63章 CSS 下拉菜单

使用 CSS 创建可悬停的下拉列表。

## 63.1 基础的下拉菜单

创建当用户将鼠标移到元素上时出现的下拉框。

实例

<!DOCTYPE html>

<html>

<head>

<style>

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

padding: 12px 16px;

z-index: 1;

}

.dropdown:hover .dropdown-content {

display: block;

}

</style>

</head>

<body>

<h1>可悬停的下拉菜单</h1>

<p>请把鼠标移到文本上，已查看下拉内容。</p>

<div class="dropdown">

<span>把鼠标移到我上面</span>

<div class="dropdown-content">

<p>Hello World!</p>

</div>

</div>

</body>

</html>

例子解释：

HTML

使用任何元素打开下拉菜单内容，例如 <span> 或 <button> 元素。

使用容器元素（如 <div>）创建下拉内容，并在其中添加任何内容。

用 <div> 元素包围这些元素，使用 CSS 正确定位下拉内容。

CSS

.dropdown 类使用 position:relative，当我们希望将下拉内容放置在下拉按钮的正下方（使用 position:absolute）时，需要使用该类。

.dropdown-content 类保存实际的下拉菜单内容。默认情况下它是隐藏的，并将在悬停时显示（请看下文）。请注意，min-width 设置为 160px。可随时更改此设置。提示：如果您希望下拉内容的宽度与下拉按钮的宽度一样，请将宽度设置为 100％（设置 overflow:auto 可实现在小屏幕上滚动）。

我们用了 CSS box-shadow 属性，而不是边框，这样下拉菜单看起来像一张“卡片”。

当用户将鼠标移到下拉按钮上时，:hover 选择器用于显示下拉菜单。

## 63.2 下拉式菜单

<!DOCTYPE html>

<html>

<head>

<style>

.dropbtn {

background-color: #4CAF50;

color: white;

padding: 16px;

font-size: 16px;

border: none;

cursor: pointer;

}

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

}

.dropdown-content a:hover {background-color: #f1f1f1}

.dropdown:hover .dropdown-content {

display: block;

}

.dropdown:hover .dropbtn {

background-color: #3e8e41;

}

</style>

</head>

<body>

<h1>下拉菜单</h1>

<p>请把鼠标移到按钮上，以打开下拉菜单。</p>

<div class="dropdown">

<button class="dropbtn">Dropdown</button>

<div class="dropdown-content">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</div>

<p><b>Note:</b>我们的测试链接用了 href="#"。真实的网站会用 URL。</p>

</body>

</html>

## 63.3 右对齐的下拉菜单内容

如果希望下拉菜单从右到左而不是从左到右，请添加 right: 0;：

<!DOCTYPE html>

<html>

<head>

<style>

.dropbtn {

background-color: #4CAF50;

color: white;

padding: 16px;

font-size: 16px;

border: none;

cursor: pointer;

}

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

right: 0;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

}

.dropdown-content a:hover {background-color: #f1f1f1;}

.dropdown:hover .dropdown-content {

display: block;

}

.dropdown:hover .dropbtn {

background-color: #3e8e41;

}

</style>

</head>

<body>

<h1>对齐的下拉内容</h1>

<p>通过 left 和 right 属性，决定下拉内容是左到右还是右到左。</p>

<div class="dropdown" style="float:left;">

<button class="dropbtn">Left</button>

<div class="dropdown-content" style="left:0;">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</div>

<div class="dropdown" style="float:right;">

<button class="dropbtn">Right</button>

<div class="dropdown-content">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</div>

</body>

</html>

## 63.4 更多实例

1 - 下拉式图像

如何在下拉框中添加图像和其他内容。

请把鼠标指针悬停在图像上：

<!DOCTYPE html>

<html>

<head>

<style>

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown:hover .dropdown-content {

display: block;

}

.desc {

padding: 15px;

text-align: center;

}

</style>

</head>

<body>

<h1>下拉图片</h1>

<p>请把鼠标移动到图像上，以打开下拉内容。</p>

<div class="dropdown">

<img src="/i/photo/coffee.jpg" alt="Coffee" width="100">

<div class="dropdown-content">

<img src="/i/photo/coffee.jpg" alt="Coffee" width="300" height="200">

<div class="desc">Coffee</div>

</div>

</div>

</body>

</html>

2 - 下拉式导航

如何在导航栏中添加下拉菜单。

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a, .dropbtn {

display: inline-block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover, .dropdown:hover .dropbtn {

background-color: red;

}

li.dropdown {

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

text-align: left;

}

.dropdown-content a:hover {background-color: #f1f1f1;}

.dropdown:hover .dropdown-content {

display: block;

}

</style>

</head>

<body>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li class="dropdown">

<a href="javascript:void(0)" class="dropbtn">Dropdown</a>

<div class="dropdown-content">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</li>

</ul>

<h1>导航栏内的下拉菜单</h1>

<p>请悬停到 "Dropdown" 链接上，以查看下拉菜单。</p>

</body>

</html>

# 第64章 CSS 图片库

## 64.1 图片库

下面这个图片库是使用 CSS 创建的：

实例

<!DOCTYPE html>

<html>

<head>

<style>

div.gallery {

margin: 5px;

border: 1px solid #ccc;

float: left;

width: 180px;

}

div.gallery:hover {

border: 1px solid #777;

}

div.gallery img {

width: 100%;

height: auto;

}

div.desc {

padding: 15px;

text-align: center;

}

</style>

</head>

<body>

<div class="gallery">

<a target="\_blank" href="/i/photo/tulip-yellow.jpg">

<img src="/i/photo/tulip-yellow.jpg" alt="Cinque Terre" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

<div class="gallery">

<a target="\_blank" href="/i/photo/tulip-red.jpg">

<img src="/i/photo/tulip-red.jpg" alt="Forest" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

<div class="gallery">

<a target="\_blank" href="/i/photo/flower-1.jpg">

<img src="/i/photo/flower-1.jpg" alt="Northern Lights" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

<div class="gallery">

<a target="\_blank" href="/i/photo/flower-2.jpg">

<img src="/i/photo/flower-2.jpg" alt="Mountains" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</body>

</html>

## 64.2 响应式图库

<!DOCTYPE html>

<html>

<head>

<style>

div.gallery {

border: 1px solid #ccc;

}

div.gallery:hover {

border: 1px solid #777;

}

div.gallery img {

width: 100%;

height: auto;

}

div.desc {

padding: 15px;

text-align: center;

}

\* {

box-sizing: border-box;

}

.responsive {

padding: 0 6px;

float: left;

width: 24.99999%;

}

@media only screen and (max-width: 700px) {

.responsive {

width: 49.99999%;

margin: 6px 0;

}

}

@media only screen and (max-width: 500px) {

.responsive {

width: 100%;

}

}

.clearfix:after {

content: "";

display: table;

clear: both;

}

</style>

</head>

<body>

<h1>响应式图片库</h1>

<h2>请调整窗口大小来查看效果。</h2>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/tulip-yellow.jpg">

<img src="/i/photo/tulip-yellow.jpg" alt="黄色郁金香" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/tulip-red.jpg">

<img src="/i/photo/tulip-red.jpg" alt="红色郁金香" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/flower-1.jpg">

<img src="/i/photo/flower-1.jpg" alt="花花草草" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/flower-2.jpg">

<img src="/i/photo/flower-2.jpg" alt="花花草草" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="clearfix"></div>

<div style="padding:6px;">

<p>本例使用媒体查询来重新排列不同屏幕尺寸的图像：对于宽于 700 像素的屏幕，它将并排显示四幅图像；对于小于 700 像素的屏幕，将并排显示两幅图像。对于小于 500 像素的屏幕，图像将垂直堆叠（100％）。</p>

<p>您稍后将在我们的 CSS 教程中学到有关媒体查询和响应式 Web 设计的更多知识。</p>

</div>

</body>

</html>

# 第65章 CSS 图像精灵

## 65.1 图像精灵

图像精灵是单个图像中包含的图像集合。

包含许多图像的网页可能需要很长时间才能加载，同时会生成多个服务器请求。

使用图像精灵将减少服务器请求的数量并节约带宽。

## 65.2 图像精灵 - 简单的例子

我们使用以下单幅图像（"navsprites.gif"）而不是使用三幅单独的图像：

通过使用 CSS，我们可以仅显示所需图像的某个部分。

在下面的例子中，CSS 指定了显示 "navsprites.gif" 图像的哪部分：

实例

<!DOCTYPE html>

<html>

<head>

<style>

#home {

width: 46px;

height: 44px;

background: url(/i/css/navsprites.gif) 0 0;

}

#next {

width: 43px;

height: 44px;

background: url(/i/css/navsprites.gif) -91px 0;

}

</style>

</head>

<body>

<img id="home" src="/i/css/trans.gif" width="1" height="1">

<img id="next" src="/i/css/trans.gif" width="1" height="1">

</body>

</html>

例子解释：

<img id="home" src="trans.gif"> - 仅定义小的透明图像，因为 src 属性不能为空。而实际显示的图像将是我们在 CSS 中指定的背景图像。

width: 46px; height: 44px; - 定义我们要使用的图像部分

background: url(navsprites.gif) 0 0; - 定义背景图片及其位置（left 0px, top 0px）

## 65.3 图像精灵 - 创建导航列表

我们希望使用精灵图片（"navsprites.gif"）来创建一个导航列表。

我们将使用 HTML 列表，因为它可以是链接，同时还支持背景图片：

<!DOCTYPE html>

<html>

<head>

<style>

#navlist {

position: relative;

}

#navlist li {

margin: 0;

padding: 0;

list-style: none;

position: absolute;

top: 0;

}

#navlist li, #navlist a {

height: 44px;

display: block;

}

#home {

left: 0px;

width: 43px;

background: url('/i/css/navsprites.gif') 0 0;

}

#prev {

left: 63px;

width: 42px;

background: url('/i/css/navsprites.gif') -47px 0;

}

#next {

left: 129px;

width: 42px;

background: url('/i/css/navsprites.gif') -91px 0;

}

</style>

</head>

<body>

<ul id="navlist">

<li id="home"><a href="/css/index.asp"></a></li>

<li id="prev"><a href="/css/css\_jianjie.asp"></a></li>

<li id="next"><a href="/css/css\_syntax.asp"></a></li>

</ul>

</body>

</html>

例子解释：

#navlist {position:relative;} - 位置设置为相对，以允许在其中进行绝对定位

#navlist li {margin:0;padding:0;list-style:none;position:absolute;top:0;} - 外边距和内边距设置为 0，删除 list-style，并且所有列表项都均为绝对定位

#navlist li, #navlist a {height:44px;display:block;} - 所有图片的高度均为 44px

现在开始为每个特定部分设置定位和样式：

#home {left:0px;width:46px;} - 一直向左定位，图像宽度 46px

#home {background:url(navsprites.gif) 0 0;} - 定义背景图片及其位置（left 0px, top 0px）

#prev {left:63px;width:43px;} - 向右定位 63px（#home 宽度 46px + 项目之间的一些额外空间），宽度 43px。

#prev {background:url('navsprites.gif') -47px 0;} - 定义背景图片向右 47px（#home 宽度 46px + 1px 分隔线）

#next {left:129px;width:43px;} - 向右定位 129px（#prev 开始是 63px + #prev 的宽度是 43px + 多余的空格），宽度 43px。

#next {background:url('navsprites.gif') -91px 0;} - 定义背景图片向右 91px（#home 宽度 46px + 1px 分隔线+ #prev 宽度 43px + 1px 分隔线）

## 65.4 图像精灵 - 悬停效果

现在，我们要向导航列表中添加悬停效果。

提示：:hover 选择器可用于所有元素，而不仅限于链接。

我们的新图像（"navsprites\_hover.gif"）包含三幅导航图像和三幅用于悬停效果的图像：

因为这是一幅图像，而不是六个单独的文件，所以当用户将鼠标悬停在图像上时，不会有加载延迟。

我们仅添加三行代码来实现悬停效果：

实例

<!DOCTYPE html>

<html>

<head>

<style>

#navlist {

position: relative;

}

#navlist li {

margin: 0;

padding: 0;

list-style: none;

position: absolute;

top: 0;

}

#navlist li, #navlist a {

height: 44px;

display: block;

}

#home {

left: 0px;

width: 43px;

background: url('/i/css/navsprites\_hover.gif') 0 0;

}

#prev {

left: 63px;

width: 42px;

background: url('/i/css/navsprites\_hover.gif') -47px 0;

}

#next {

left: 129px;

width: 42px;

background: url('/i/css/navsprites\_hover.gif') -91px 0;

}

#home a:hover {

background: url('/i/css/navsprites\_hover.gif') 0 -45px;

}

#prev a:hover {

background: url('/i/css/navsprites\_hover.gif') -47px -45px;

}

#next a:hover {

background: url('/i/css/navsprites\_hover.gif') -91px -45px;

}

</style>

</head>

<body>

<ul id="navlist">

<li id="home"><a href="/css/index.asp"></a></li>

<li id="prev"><a href="/css/css\_jianjie.asp"></a></li>

<li id="next"><a href="/css/css\_syntax.asp"></a></li>

</ul>

</body>

</html>

例子解释：

#home a:hover {background: transparent url('img\_navsprites\_hover.gif') 0 -45px;} - 我们为所有三个悬停图像指定相同的背景位置，仅向下 45 像素

# 第66章 CSS 属性选择器

为带有特定属性的 HTML 元素设置样式

我们可以设置带有特定属性或属性值的 HTML 元素的样式。

## 66.1 CSS [attribute] 选择器

[attribute] 选择器用于选取带有指定属性的元素。

下例选择所有带有 target 属性的 <a> 元素；

实例

a[target] {

background-color: yellow;

}

## 66.2 CSS [attribute="value"] 选择器

[attribute="value"] 选择器用于选取带有指定属性和值的元素。

下例选取所有带有 target="\_blank" 属性的 <a> 元素：

实例

a[target="\_blank"] {

background-color: yellow;

}

## 66.3 CSS [attribute~="value"] 选择器

[attribute~="value"] 选择器选取属性值包含指定词的元素。

下例选取 title 属性包含 "flower" 单词的所有元素：

实例

[title~="flower"] {

border: 5px solid yellow;

}

上面的例子会匹配以下属性的元素：title="flower"、title="summer flower" 以及 title="flower new"，但不匹配：title="my-flower" 或 title="flowers"。

## 66.4 CSS [attribute|="value"] 选择器

[attribute|="value"] 选择器用于选取指定属性以指定值开头的元素。

下例选取 class 属性以 "top" 开头的所有元素：

注释：值必须是完整或单独的单词，比如 class="top" 或者后跟连字符的，比如 class="top-text"。

实例

[class|="top"] {

background: yellow;

}

## 66.5 CSS [attribute^="value"] 选择器

[attribute^="value"] 选择器用于选取指定属性以指定值开头的元素。

下例选取 class 属性以 "top" 开头的所有元素：

提示：值不必是完整单词！

实例

[class^="top"] {

background: yellow;

}

## 66.6 CSS [attribute$="value"] 选择器

[attribute$="value"] 选择器用于选取指定属性以指定值结尾的元素。

下例选取 class 属性以 "test" 结尾的所有元素：

提示：值不必是完整单词！

实例

[class$="test"] {

background: yellow;

}

## 66.7 CSS [attribute\*="value"] 选择器

[attribute\*="value"] 选择器选取属性值包含指定词的元素。

下例选取 class 属性包含 "te" 的所有元素：

提示：值不必是完整单词！

实例

[class\*="te"] {

background: yellow;

}

## 66.8 设置表单样式

若需为不带 class 或 id 的表单设置样式，属性选择器会很有用：

实例

input[type="text"] {

width: 150px;

display: block;

margin-bottom: 10px;

background-color: yellow;

}

input[type="button"] {

width: 120px;

margin-left: 35px;

display: block;

}

## 66.9 所有 CSS 属性选择器

|  |  |  |
| --- | --- | --- |
| 选择器 | 例子 | 例子描述 |
| [attribute] | [target] | 选择带有 target 属性的所有元素。 |
| [attribute=value] | [target=\_blank] | 选择带有 target="\_blank" 属性的所有元素。 |
| [attribute~=value] | [title~=flower] | 选择带有包含 "flower" 一词的 title 属性的所有元素。 |
| [attribute|=value] | [lang|=en] | 选择带有以 "en" 开头的 lang 属性的所有元素。 |
| [attribute^=value] | a[href^="https"] | 选择其 href 属性值以 "https" 开头的每个 <a> 元素。 |
| [attribute$=value] | a[href$=".pdf"] | 选择其 href 属性值以 ".pdf" 结尾的每个 <a> 元素。 |
| [attribute\*=value] | a[href\*="w3school"] | 选择其 href 属性值包含子串 "w3school" 的每个 <a> 元素。 |

# 第67章 CSS 表单

## 67.1 设置输入字段的样式

使用 width 属性来确定输入字段的宽度：

<!DOCTYPE html>

<html>

<head>

<style>

input {

width: 100%;

}

</style>

</head>

<body>

<p>全宽的输入字段：</p>

<form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="fname">

</form>

</body>

</html>

上例适用于所有 <input> 元素。如果只想设置特定输入类型的样式，则可以使用属性选择器：

input[type=text] - 将仅选择文本字段

input[type=password] - 将仅选择密码字段

input[type=number] - 将仅选择数字字段

## 67.2 填充输入框

使用 padding 属性在文本字段内添加空间。

提示：若有很多输入，那么您可能还需要添加外边距，以便在它们之外添加更多空间：

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

box-sizing: border-box;

}

</style>

</head>

<body>

<p>已填充的文本字段：</p>

<form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="fname">

<label for="lname">Last Name</label>

<input type="text" id="lname" name="lname">

</form>

</body>

</html>

请注意，我们已将 box-sizing 属性设置为 border-box。这样可以确保元素的总宽度和高度中包括内边距（填充）和最终的边框。

## 67.3 带边框的输入框

请使用 border 属性更改边框的粗细和颜色，并使用 border-radius 属性添加圆角：

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

box-sizing: border-box;

border: 2px solid red;

border-radius: 4px;

}

</style>

</head>

<body>

<p>带边框的文本字段：</p>

<form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="fname">

<label for="lname">Last Name</label>

<input type="text" id="lname" name="lname">

</form>

</body>

</html>

如果仅需要下边框，请使用 border-bottom 属性：

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

box-sizing: border-box;

border: none;

border-bottom: 2px solid red;

}

</style>

</head>

<body>

<p>只有下边框的文本字段：</p>

<form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="fname">

<label for="lname">Last Name</label>

<input type="text" id="lname" name="lname">

</form>

</body>

</html>

## 67.4 彩色的输入框

请使用 background-color 属性为输入添加背景色，并使用 color 属性更改文本颜色：

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

box-sizing: border-box;

border: none;

background-color: #3CBC8D;

color: white;

}

</style>

</head>

<body>

<p>有颜色的文本字段：</p>

<form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="fname" value="Bill">

<label for="lname">Last Name</label>

<input type="text" id="lname" name="lname" value="Gates">

</form>

</body>

</html>

## 67.5 获得焦点的输入框

默认情况下，某些浏览器在获得焦点（单击）时会在输入框周围添加蓝色轮廓。您可以通过向输入添加 outline: none; 来消除此行为。

使用 :focus 选择器可以在输入字段获得焦点时为其设置样式：

实例 1

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

box-sizing: border-box;

border: 1px solid #555;

outline: none;

}

input[type=text]:focus {

background-color: lightblue;

}

</style>

</head>

<body>

<p>在本例中，我们使用了 :focus 选择器，在文本字段获得焦点时（被点击）为其添加背景色：</p>

<form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="fname" value="Bill">

<label for="lname">Last Name</label>

<input type="text" id="lname" name="lname" value="Gates">

</form>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

box-sizing: border-box;

border: 3px solid #ccc;

-webkit-transition: 0.5s;

transition: 0.5s;

outline: none;

}

input[type=text]:focus {

border: 3px solid #555;

}

</style>

</head>

<body>

<p>在本例中，我们使用了 :focus 选择器，在文本字段获得焦点时（被点击）为其添加黑色边框：</p>

<p>请注意，我们已添加 CSS 过渡属性以设置边框颜色的动画（改变颜色需 0.5 秒）。</p>

<form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="fname" value="Bill">

<label for="lname">Last Name</label>

<input type="text" id="lname" name="lname" value="Gates">

</form>

</body>

</html>

## 67.6 带有图标/图像的输入框

如果希望在输入框中包含图标，请使用 background-image 属性，并将其与 background-position 属性一起设置。还要注意，我们添加了一个较大的左内边距（padding-left）来留出图标的空间：

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 100%;

box-sizing: border-box;

border: 2px solid #ccc;

border-radius: 4px;

font-size: 16px;

background-color: white;

background-image: url('searchicon.png');

background-position: 10px 10px;

background-repeat: no-repeat;

padding: 12px 20px 12px 40px;

}

</style>

</head>

<body>

<p>带图标的输入框：</p>

<form>

<input type="text" name="search" placeholder="Search..">

</form>

</body>

</html>

## 67.7 带动画效果的搜索输入框

在本例中，我们使用 CSS transition 属性为搜索输入框获得焦点时的宽度变化设置动画。稍后，您将在我们的 CSS 过渡 一章中学到更多有关 transition 属性的知识。

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 130px;

box-sizing: border-box;

border: 2px solid #ccc;

border-radius: 4px;

font-size: 16px;

background-color: white;

background-image: url('searchicon.png');

background-position: 10px 10px;

background-repeat: no-repeat;

padding: 12px 20px 12px 40px;

transition: width 0.4s ease-in-out;

}

input[type=text]:focus {

width: 100%;

}

</style>

</head>

<body>

<p>有动画效果的输入字段：</p>

<form>

<input type="text" name="search" placeholder="Search..">

</form>

</body>

</html>

## 67.8 设置文本域的样式

提示：使用 resize 属性可防止对 textareas 调整大小（禁用右下角的“抓取器”）：

<!DOCTYPE html>

<html>

<head>

<style>

textarea {

width: 100%;

height: 150px;

padding: 12px 20px;

box-sizing: border-box;

border: 2px solid #ccc;

border-radius: 4px;

background-color: #f8f8f8;

font-size: 16px;

resize: none;

}

</style>

</head>

<body>

<p><b>提示：</b>使用 resize 属性可防止调整 textareas 的大小（禁用右下角的“抓取器”）：</p>

<form>

<textarea>Some text...

## 67.9 设置选择菜单的样式

<!DOCTYPE html>

<html>

<head>

<style>

select {

width: 100%;

padding: 16px 20px;

border: none;

border-radius: 4px;

background-color: #f1f1f1;

}

</style>

</head>

<body>

<p>有样式的选择菜单：</p>

<form>

<select id="country" name="country">

<option value="au">Australia</option>

<option value="ca">Canada</option>

<option value="usa">USA</option>

</select>

</form>

</body>

</html>

## 67.10 设置输入按钮的样式

<!DOCTYPE html>

<html>

<head>

<style>

input[type=button], input[type=submit], input[type=reset] {

background-color: #4CAF50;

border: none;

color: white;

padding: 16px 32px;

text-decoration: none;

margin: 4px 2px;

cursor: pointer;

}

</style>

</head>

<body>

<h1>有样式的输入按钮</h1>

<input type="button" value="按钮">

<input type="reset" value="重置">

<input type="submit" value="提交">

</body>

</html>

## 67.11 响应式菜单

请调整 TIY 编辑器窗口的大小来查看效果。当屏幕的宽度小于 600 像素时，使两列上下堆叠而不是左右堆叠。

高级：接下来的例子使用 媒体查询 来创建响应式表单。在下一章中，您将学到更多相关知识。

# 第68章 CSS 计数器

CSS 计数器是由 CSS 保持的“变量”，其值可以通过 CSS 规则递增（以跟踪其使用次数）。

计数器使您可以根据内容在文档中的位置来调整其外观。

## 68.1 带计数器的自动编号

CSS 计数器就像“变量”。变量值可以通过 CSS 规则递增（将跟踪它们的使用次数）。

如需使用 CSS 计数器，我们将使用以下属性：

counter-reset - 创建或重置计数器

counter-increment - 递增计数器值

content - 插入生成的内容

counter() 或 counters() 函数 - 将计数器的值添加到元素

如需使用 CSS 计数器，必须首先使用 counter-reset 创建它。

下面的例子为页面（在 body 选择器中）创建一个计数器，然后为每个 <h2> 元素增加计数器值，并在每个 <h2> 元素的开头添加 "Section <value of the counter>:"：

实例

<!DOCTYPE html>

<html>

<head>

<style>

body {

counter-reset: section;

}

h2::before {

counter-increment: section;

content: "Section " counter(section) ": ";

}

</style>

</head>

<body>

<h1>使用 CSS 计数器：</h1>

<h2>HTML 教程</h2>

<h2>CSS 教程</h2>

<h2>JavaScript 教程</h2>

</body>

</html>

## 68.2 嵌套计数器

下面的例子为页面（section）创建一个计数器，为每个 <h1> 元素（subsection）创建一个计数器。

"section" 计数器为每个 <h1> 元素计数，同时写入 "Section" 以及 section 计数器的值，"subsection" 计数器为每个 <h2> 元素计数，同时写入 section 计数器的值以及 subsection 计数器的值：

<!DOCTYPE html>

<html>

<head>

<style>

body {

counter-reset: section;

}

h1 {

counter-reset: subsection;

}

h1::before {

counter-increment: section;

content: "Section " counter(section) ". ";

}

h2::before {

counter-increment: subsection;

content: counter(section) "." counter(subsection) " ";

}

</style>

</head>

<body>

<h1>HTML 教程：</h1>

<h2>HTML 教程</h2>

<h2>CSS 教程</h2>

<h1>Scripting 教程:</h1>

<h2>JavaScript</h2>

<h2>VBScript</h2>

<h1>XML 教程：</h1>

<h2>XML</h2>

<h2>XSL</h2>

</body>

</html>

计数器对于创建概述列表也很有用，因为在子元素中会自动创建一个计数器的新实例。在这里，我们使用 counters() 函数在不同级别的嵌套计数器之间插入一个字符串：

<!DOCTYPE html>

<html>

<head>

<style>

ol {

counter-reset: section;

list-style-type: none;

}

li::before {

counter-increment: section;

content: counters(section,".") " ";

}

</style>

</head>

<body>

<ol>

<li>item</li>

<li>item

<ol>

<li>item</li>

<li>item</li>

<li>item

<ol>

<li>item</li>

<li>item</li>

<li>item</li>

</ol>

</li>

<li>item</li>

</ol>

</li>

<li>item</li>

<li>item</li>

</ol>

<ol>

<li>item</li>

<li>item</li>

</ol>

</body>

</html>

## 68.3 CSS 计数器属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| content | 与 ::before 和 ::after 伪元素一同使用，来插入生成的内容。 |
| counter-increment | 递增一个或多个计数器值。 |
| counter-reset | 创建或重置一个或多个计数器。 |

# 第69章 CSS 网站布局

网站布局

网站通常分为页眉、菜单、内容和页脚：

页眉

页眉（header）通常位于网站顶部（或顶部导航菜单的正下方）。它通常包含徽标（logo）或网站名称：

实例

.header {

background-color: #F1F1F1;

text-align: center;

padding: 20px;

}

## 69.1 导航栏

导航栏包含链接列表，以帮助访问者浏览您的网站：

<!DOCTYPE html>

<html lang="en">

<head>

<title>CSS Website Layout</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0;

}

/\* 设置页眉的样式 \*/

.header {

background-color: #f1f1f1;

padding: 20px;

text-align: center;

}

/\* 设置顶部导航栏的样式 \*/

.topnav {

overflow: hidden;

background-color: #333;

}

/\* 设置 topnav 链接的样式 \*/

.topnav a {

float: left;

display: block;

color: #f2f2f2;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

/\* 改变鼠标悬停时的颜色 \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

</style>

</head>

<body>

<div class="header">

<h1>Header</h1>

</div>

<div class="topnav">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

</div>

</body>

</html>

## 69.2 内容

使用哪种布局通常取决于您的目标用户。最常见的布局是以下布局之一（或将它们组合在一起）：

1-列布局（通常用于移动浏览器）

2-列布局（通常用于平板电脑和笔记本电脑）

3-列布局 （仅用于台式机）

<!DOCTYPE html>

<html lang="en">

<head>

<title>CSS 网站布局</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0;

}

/\* 设置页眉的样式 \*/

.header {

background-color: #f1f1f1;

padding: 20px;

text-align: center;

}

/\* 设置顶部导航栏的样式 \*/

.topnav {

overflow: hidden;

background-color: #333;

}

/\* 设置 topnav 链接的样式 \*/

.topnav a {

float: left;

display: block;

color: #f2f2f2;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

/\* 改变鼠标悬停时的颜色 \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

/\* 创建并排的三个非等列 \*/

.column {

float: left;

width: 33.33%;

padding: 15px;

}

/\* 清除列之后的浮动 \*/

.row:after {

content: "";

display: table;

clear: both;

}

/\* 响应式布局 - 创建堆叠而非并排的三列 \*/

@media screen and (max-width:600px) {

.column {

width: 100%;

}

}

</style>

</head>

<body>

<div class="header">

<h1>Header</h1>

<p>请调整浏览器窗口的大小以查看响应效果。</p>

</div>

<div class="topnav">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

</div>

<div class="row">

<div class="column">

<h2>Column</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>

</div>

<div class="column">

<h2>Column</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>

</div>

<div class="column">

<h2>Column</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>

</div>

</div>

</body>

</html>

提示：如需创建 2 列布局，请将宽度更改为 50％。如需创建 4 列布局，请使用 25％。

提示：您是否想知道 @media 规则如何工作？请在我们的 CSS 媒体查询 这一章中学习更多相关知识。

提示：创建列布局的一种更现代的方法是使用 CSS Flexbox。但是，Internet Explorer 10 和更早版本不支持它。如果需要 IE6-10 支持，请使用浮动（如上所示）。

## 69.3 不相等的栏

主要内容（main content）是您网站上最大、最重要的部分。

列宽不相等的情况很常见，因为大部分空间都为主内容保留。附属内容（如果有）通常用作替代导航或指定与主要内容有关的信息。您可以随意更改宽度，只要记住它的总和应为 100％：

<!DOCTYPE html>

<html lang="en">

<head>

<title>CSS 网站布局</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0;

}

/\* 设置页眉的样式 \*/

.header {

background-color: #f1f1f1;

padding: 20px;

text-align: center;

}

/\* 设置顶部导航栏的样式 \*/

.topnav {

overflow: hidden;

background-color: #333;

}

/\* 设置 topnav 链接的样式 \*/

.topnav a {

float: left;

display: block;

color: #f2f2f2;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

/\* 改变鼠标悬停时的颜色 \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

/\* 创建并排的三个不相等的列 \*/

.column {

float: left;

padding: 10px;

}

/\* 左右列 \*/

.column.side {

width: 25%;

}

/\* 中间列 \*/

.column.middle {

width: 50%;

}

/\* 清除列之后的浮动 \*/

.row:after {

content: "";

display: table;

clear: both;

}

/\* 响应式布局 - 创建堆叠而非并排的三列 \*/

@media screen and (max-width: 600px) {

.column.side, .column.middle {

width: 100%;

}

}

</style>

</head>

<body>

<div class="header">

<h1>Header</h1>

<p>请调整浏览器窗口的大小以查看响应效果。</p>

</div>

<div class="topnav">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

</div>

<div class="row">

<div class="column side">

<h2>Side</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>

</div>

<div class="column middle">

<h2>Main Content</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>

</div>

<div class="column side">

<h2>Side</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>

</div>

</div>

</body>

</html>

## 69.4 页脚

页脚位于页面底部。它通常包含诸如版权和联系方式之类的信息：

实例

<!DOCTYPE html>

<html lang="en">

<head>

<title>CSS 网站布局</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0;

}

/\* 设置页眉样式 \*/

.header {

background-color: #f1f1f1;

padding: 20px;

text-align: center;

}

/\* 设置顶部导航栏的样式 \*/

.topnav {

overflow: hidden;

background-color: #333;

}

/\* 设置 topnav 链接的样式 \*/

.topnav a {

float: left;

display: block;

color: #f2f2f2;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

/\* 改变鼠标悬停时的颜色 \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

/\* 创建并排的三个非等列 \*/

.column {

float: left;

padding: 10px;

}

/\* 左和右列 \*/

.column.side {

width: 25%;

}

/\* 中间列 \*/

.column.middle {

width: 50%;

}

/\* 清除列之后的浮动 \*/

.row:after {

content: "";

display: table;

clear: both;

}

/\* 响应式布局 - 创建堆叠而非并排的三列 \*/

@media screen and (max-width: 600px) {

.column.side, .column.middle {

width: 100%;

}

}

/\* 设置页脚的样式 \*/

.footer {

background-color: #f1f1f1;

padding: 10px;

text-align: center;

}

</style>

</head>

<body>

<div class="header">

<h1>Header</h1>

<p>请调整浏览器窗口的大小以查看响应效果。</p>

</div>

<div class="topnav">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

</div>

<div class="row">

<div class="column side">

<h2>Side</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>

</div>

<div class="column middle">

<h2>Main Content</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.</p>

</div>

<div class="column side">

<h2>Side</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit..</p>

</div>

</div>

<div class="footer">

<p>Footer</p>

</div>

</body>

</html>

## 69.5 响应式网站布局

通过使用上面的这些 CSS 代码，我们创建了一个自适应的网站布局，会根据屏幕宽度切换两列与全宽列：

<!DOCTYPE html>

<html>

<head>

<style>

\* {

box-sizing: border-box;

}

body {

font-family: Arial;

padding: 10px;

background: #f1f1f1;

}

/\* 页眉/Blog 标题 \*/

.header {

padding: 30px;

text-align: center;

background: white;

}

.header h1 {

font-size: 50px;

}

/\* 设置上导航栏的样式 \*/

.topnav {

overflow: hidden;

background-color: #333;

}

/\* 设置 topnav 链接的样式 \*/

.topnav a {

float: left;

display: block;

color: #f2f2f2;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

/\* 改变鼠标悬停时的颜色 \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

/\* 创建两个不相等的彼此并排的浮动列 \*/

/\* 左列 \*/

.leftcolumn {

float: left;

width: 75%;

}

/\* 右列 \*/

.rightcolumn {

float: left;

width: 25%;

background-color: #f1f1f1;

padding-left: 20px;

}

/\* 伪图像 \*/

.fakeimg {

background-color: #aaa;

width: 100%;

padding: 20px;

}

/\* 为文章添加卡片效果 \*/

.card {

background-color: white;

padding: 20px;

margin-top: 20px;

}

/\* 清除列之后的浮动 \*/

.row:after {

content: "";

display: table;

clear: both;

}

/\* 页脚 \*/

.footer {

padding: 20px;

text-align: center;

background: #ddd;

margin-top: 20px;

}

/\* 响应式布局 - 当屏幕的宽度小于 800 像素时，使两列堆叠而不是并排 \*/

@media screen and (max-width: 800px) {

.leftcolumn, .rightcolumn {

width: 100%;

padding: 0;

}

}

/\* 响应式布局 - 当屏幕的宽度小于 400 像素时，使导航链接堆叠而不是并排 \*/

@media screen and (max-width: 400px) {

.topnav a {

float: none;

width: 100%;

}

}

</style>

</head>

<body>

<div class="header">

<h1>My Website</h1>

<p>Resize the browser window to see the effect.</p>

</div>

<div class="topnav">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#" style="float:right">Link</a>

</div>

<div class="row">

<div class="leftcolumn">

<div class="card">

<h2>TITLE HEADING</h2>

<h5>Title description, Dec 7, 2017</h5>

<div class="fakeimg" style="height:200px;">Image</div>

<p>Some text..</p>

<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>

</div>

<div class="card">

<h2>TITLE HEADING</h2>

<h5>Title description, Sep 2, 2017</h5>

<div class="fakeimg" style="height:200px;">Image</div>

<p>Some text..</p>

<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>

</div>

</div>

<div class="rightcolumn">

<div class="card">

<h2>About Me</h2>

<div class="fakeimg" style="height:100px;">Image</div>

<p>Some text about me in culpa qui officia deserunt mollit anim..</p>

</div>

<div class="card">

<h3>Popular Post</h3>

<div class="fakeimg"><p>Image</p></div>

<div class="fakeimg"><p>Image</p></div>

<div class="fakeimg"><p>Image</p></div>

</div>

<div class="card">

<h3>Follow Me</h3>

<p>Some text..</p>

</div>

</div>

</div>

<div class="footer">

<h2>Footer</h2>

</div>

</body>

</html>

# 第70章 CSS 单位

## 70.1 CSS 单位

CSS 有几种表示长度的不同单位。

许多 CSS 属性接受“长度”值，诸如 width、margin、padding、font-size 等。

长度是一个后面跟着长度单位的数字，诸如 10px、2em 等。

实例

使用 px（像素）设置不同的长度值：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

font-size: 60px;

}

p {

font-size: 25px;

line-height: 50px;

}

</style>

</head>

<body>

<h1>这是标题 1</h1>

<h2>这是标题 2</h2>

<p>这是一个段落。</p>

<p>这是另一个段落。</p>

</body>

</html>

数字和单位之间不能出现空格。但是，如果值为 0，则可以省略单位。

对于某些 CSS 属性，允许使用负的长度。

长度单位有两种类型：绝对单位和相对单位。

## 70.2 绝对长度

绝对长度单位是固定的，用任何一个绝对长度表示的长度都将恰好显示为这个尺寸。

不建议在屏幕上使用绝对长度单位，因为屏幕尺寸变化很大。但是，如果已知输出介质，则可以使用它们，例如用于打印布局（print layout）。

|  |  |  |
| --- | --- | --- |
| 单位 | 描述 | TIY |
| cm | 厘米 | <!DOCTYPE html>  <html>  <head>  <style>  h1 {font-size: 1.5cm;}  h2 {font-size: 1cm;}  p {  font-size: 0.5cm;  line-height: 1cm;  }  </style>  </head>  <body>  <h1>这是标题 1</h1>  <h2>这是标题 2</h2>  <p>这是一个段落。</p>  <p>这是另一个段落。</p>  </body>  </html> |
| mm | 毫米 | <!DOCTYPE html>  <html>  <head>  <style>  h1 {font-size: 15mm;}  h2 {font-size: 10mm;}  p {  font-size: 5mm;  line-height: 10mm;  }  </style>  </head>  <body>  <h1>这是标题 1</h1>  <h2>这是标题 2</h2>  <p>这是一个段落。</p>  <p>这是另一个段落。</p>  </body>  </html> |
| in | 英寸 (1in = 96px = 2.54cm) | <!DOCTYPE html>  <html>  <head>  <style>  h1 {font-size: 1in;}  h2 {font-size: 0.5in;}  p {  font-size: 0.2in;  line-height: 0.5in;  }  </style>  </head>  <body>  <h1>这是标题 1</h1>  <h2>这是标题 2</h2>  <p>这是一个段落。</p>  <p>这是另一个段落。</p>  </body>  </html> |
| px \* | 像素 (1px = 1/96th of 1in) | <!DOCTYPE html>  <html>  <head>  <style>  h1 {font-size: 50px;}  h2 {font-size: 30px;}  p {  font-size: 15px;  line-height: 20px;  }  </style>  </head>  <body>  <h1>这是标题 1</h1>  <h2>这是标题 2</h2>  <p>这是一个段落。</p>  <p>这是另一个段落。</p>  </body>  </html> |
| pt | 点 (1pt = 1/72 of 1in) | <!DOCTYPE html>  <html>  <head>  <style>  h1 {font-size: 50pt;}  h2 {font-size: 25pt;}  p {  font-size: 15pt;  line-height: 25pt;  }  </style>  </head>  <body>  <h1>这是标题 1</h1>  <h2>这是标题 2</h2>  <p>这是一个段落。</p>  <p>这是另一个段落。</p>  </body>  </html> |
| pc | 派卡 (1pc = 12 pt) | <!DOCTYPE html>  <html>  <head>  <style>  h1 {font-size: 4.5pc;}  h2 {font-size: 3pc;}  p {  font-size: 1.5pc;  line-height: 3pc;  }  </style>  </head>  <body>  <h1>这是标题 1</h1>  <h2>这是标题 2</h2>  <p>这是一个段落。</p>  <p>这是另一个段落。</p>  </body>  </html> |

\* 像素（px）是相对于观看设备的。对于低 dpi 的设备，1px 是显示器的一个设备像素（点）。对于打印机和高分辨率屏幕，1px 表示多个设备像素。

## 70.3 相对长度

相对长度单位规定相对于另一个长度属性的长度。相对长度单位在不同渲染介质之间缩放表现得更好。

|  |  |  |
| --- | --- | --- |
| 单位 | 描述 | TIY |
| em | 相对于元素的字体大小（font-size）（2em 表示当前字体大小的 2 倍） | <!DOCTYPE html>  <html>  <head>  <style>  p {  font-size: 16px;  line-height: 2em;  }  div {  font-size: 30px;  border: 1px solid black;  }  span {  font-size: 0.5em;  }  </style>  </head>  <body>  <p>这些段落的 line-height 这样计算：2x16px = 32px。</p>  <p>这些段落的 line-height 这样计算：2x16px = 32px。</p>  <p>这些段落的 line-height 这样计算：2x16px = 32px。</p>  <div>这个 div 元素的 font-size 被设置为 30px。<span>div 元素内部的 span 元素的 font-size 为 0.5em，其等于 0.5x30 = 15px。</span></div>  </body>  </html> |
| ex | 相对于当前字体的 x-height(极少使用) | <!DOCTYPE html>  <html>  <head>  <style>  div {  font-size: 30px;  border: 1px solid black;  }  span {  font-size: 1ex;  }  </style>  </head>  <body>  <div>这个 div 元素的 font-size 被设置为 30px。<span>div 元素内部的 span 元素的 font-size 为 1ex。</span></div>  </body>  </html> |
| ch | 相对于 "0"（零）的宽度 | <!DOCTYPE html>  <html>  <head>  <style>  body {  font-size:16px;  }  div {  font-size: 3ch;  border: 1px solid black;  }  </style>  </head>  <body>  <p>此文档的 font-size 是 16px。</p>  <div>这个 div 元素的 font-size 是 3ch。</div>  <p>ch 单位设置相对于字符 "0" 宽度的 font-size。</p>  </body>  </html> |
| rem | 相对于根元素的字体大小（font-size） | <!DOCTYPE html>  <html>  <head>  <style>  html {  font-size:16px;  }  div {  font-size: 3rem;  border: 1px solid black;  }  #top-div {  font-size: 2rem;  border: 1px solid red;  }  </style>  </head>  <body>  <p>此文档的 font-size 是 16px。</p>  <div id="top-div">  这个 div 的 font-size 为 2rem，转换为浏览器字体大小的2倍。  <div>这个 div 元素的 font-size 为 3rem。它同时也展示出不会继承其父元素的字体尺寸。</div>  </div>  <p>rem 单位设置的 font-size 相对于浏览器的基准字体尺寸，并不从其父元素继承。</p>  </body>  </html> |
| vw | 相对于视口\*宽度的 1% | <!DOCTYPE html>  <html>  <head>  <style>  h1 {  font-size: 20vw;  }  </style>  </head>  <body>  <h1>Hello</h1>  <p>请调整浏览器窗口的宽度，来查看 h1 的字体大小如何变化。</p>  <p>1vw = 1% 的视口宽度。</p>  <p>IE8 以及更早的版本不支持 vw 单位。</p>  </body>  </html> |
| vh | 相对于视口\*高度的 1% | <!DOCTYPE html>  <html>  <head>  <style>  h1 {  font-size: 20vh;  }  </style>  </head>  <body>  <h1>Hello</h1>  <p>请调整浏览器窗口的高度，来查看 h1 的字体大小如何变化。</p>  <p>1vh = 1% 的视口宽度。</p>  <p>IE8 或更早的版本不支持 vh 单位。</p>  </body>  </html> |
| vmin | 相对于视口\*较小尺寸的 1％ | <!DOCTYPE html>  <html>  <head>  <style>  h1 {  font-size: 15vmin;  }  </style>  </head>  <body>  <h1>Hello</h1>  <p>请调整浏览器窗口的大小（宽度和高度），来查看 h1 的字体大小如何变化。</p>  <p>1vmin = 1vw 或 1vh，以较小者为准。</p>  </body>  </html> |
| vmax | 相对于视口\*较大尺寸的 1％ | <!DOCTYPE html>  <html>  <head>  <style>  h1 {  font-size: 15vmax;  }  </style>  </head>  <body>  <h1>Hello</h1>  <p>Re请调整浏览器窗口的大小（宽度和高度），来查看 h1 的字体大小如何变化。</p>  <p>1vmax = 1vw 或 1vh，以较大者为准。</p>  <p>Edge 15 或更早的版本不支持 vmax 单位，Safari 6.1 或更早的版本也不支持。</p>  </body>  </html> |
| % | 相对于父元素 | <!DOCTYPE html>  <html>  <head>  <style>  body {  font-size:16px;  }  div {  font-size: 150%;  border: 1px solid black;  }  </style>  </head>  <body>  <p>此文档的 font-size 为 16px。</p>  <div>这个 div 元素的 font-size 是 150%。</div>  <p>% 单位设置相对于当前 font-size 的字体尺寸。</p>  </body>  </html> |

提示：em 和 rem 单元可用于创建完美的可扩展布局！

\* 视口（Viewport）= 浏览器窗口的尺寸。如果视口为 50 厘米宽，则 1vw = 0.5 厘米。

# 第71章 CSS 特异性

## 71.1 什么是特异性？

如果有两条或两条以上指向同一元素的冲突 CSS 规则，则浏览器将遵循一些原则来确定哪一条是最具体的，并因此胜出。

将特异性（specificity）视为得分/等级，能够确定最终将哪些样式声明应用于元素。

通用选择器（\*）具有较低的特异性，而 ID 选择器具有较高的特异性！

注意：这种特异性是 CSS 规则不适用于某些元素的常见原因，尽管您可能会认为应该适用。

## 71.2 特异性层次

每个选择器在特异性层次结构中都有其位置。以下四种类别定义了选择器的特异性级别：

行内样式 - 行内（内联）样式直接附加到要设置样式的元素。实例：<h1 style="color: #ffffff;">。

ID - ID 是页面元素的唯一标识符，例如 #navbar。

类、属性和伪类 - 此类别包括 .classes、[attributes] 和伪类，例如：:hover、:focus 等。

元素和伪元素 - 此类别包括元素名称和伪元素，比如 h1、div、:before 和 :after。

## 71.3 如何计算特异性？

请您牢记计算特异性的方法！

从 0 开始，为 style 属性添加 1000，为每个 ID 添加 100，为每个属性、类或伪类添加 10，为每个元素名称或伪元素添加 1。

请思考以下三个代码片段：

实例

A: h1

B: #content h1

C: <div id="content"><h1 style="color: #ffffff">Heading</h1></div>

A 的特异性为 1（一个元素）

B 的特异性为 101（一个 ID 引用以及一个元素）

C 的特异性为 1000（行内样式）

由于 1 < 101 < 1000，因此第三条规则（C）具有更高的特异性，所以将被应用。

71.4 特异性规则 1：

在特异性相同的情况下：最新的规则很重要 - 如果将同一规则两次写入外部样式表，那么样式表中后面的规将更靠近要设置样式的元素，因此会被应用：

实例

<!DOCTYPE html>

<html>

<head>

<style>

h1 {background-color: yellow;}

h1 {background-color: red;}

</style>

</head>

<body>

<h1>这是标题 1</h1>

</body>

</html>

## 71.5 特异性规则 2：

ID 选择器比属性选择器拥有更高的特异性 - 请看以下三行代码：

实例

<!DOCTYPE html>

<html>

<head>

<style>

div#a {background-color: green;}

#a {background-color: yellow;}

div[id=a] {background-color: blue;}

</style>

</head>

<body>

<div id="a">这是一个 div</div>

</body>

</html>

## 71.6 特异性规则 3：

上下文选择器比单一元素选择器更具体 - 嵌入式样式表更靠近要设置样式的元素。所以在以下情况下：

实例

来自外部 CSS 文件：

#content h1 {background-color: red;}

在 HTML 文件中：

<style>

#content h1 {

background-color: yellow;

}

</style>

将适用后一条规则。

## 71.7 特异性规则 4：

类选择器会击败任意数量的元素选择器 - 类选择器（诸如 .intro 会击败 h1、p、div 等）：

实例

<!DOCTYPE html>

<html>

<head>

<style>

.intro {background-color: yellow;}

h1 {background-color: red;}

</style>

</head>

<body>

<h1 class="intro">这是一个标题</h1>

</body>

</html>

此外，通用选择器以及被继承的值拥有 0 - \* 的特异性，body \* 及类似拥有 0 的特异性。被继承的值的特异性也为 0。

# 第72章 CSS 圆角

## 72.1 CSS 圆角

通过 CSS border-radius 属性，可以实现任何元素的“圆角”样式。

## 72.2 CSS border-radius 属性

CSS border-radius 属性定义元素角的半径。

提示：您可以使用此属性为元素添加圆角！

这里有三个例子：

1. 带有指定背景颜色的元素的圆角：
2. 带边框元素的圆角：
3. 带有背景图像的元素的圆角：

<!DOCTYPE html>

<html>

<head>

<style>

#rcorners1 {

border-radius: 25px;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners2 {

border-radius: 25px;

border: 2px solid #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners3 {

border-radius: 25px;

background: url(/i/paper.jpg);

background-position: left top;

background-repeat: repeat;

padding: 20px;

width: 200px;

height: 150px;

}

</style>

</head>

<body>

<h1>border-radius 属性</h1>

<p>拥有指定背景颜色的元素的圆角：</p>

<p id="rcorners1">Rounded corners!</p>

<p>带边框元素的圆角：</p>

<p id="rcorners2">Rounded corners!</p>

<p>拥有背景图片的元素的圆角：</p>

<p id="rcorners3">Rounded corners!</p>

</body>

</html>

提示：border-radius 属性实际上是以下属性的简写属性：

border-top-left-radius

border-top-right-radius

border-bottom-right-radius

border-bottom-left-radius

## 72.3 CSS border-radius - 指定每个角

border-radius 属性可以接受一到四个值。规则如下：

四个值 - border-radius: 15px 50px 30px 5px;（依次分别用于：左上角、右上角、右下角、左下角）：

三个值 - border-radius: 15px 50px 30px;（第一个值用于左上角，第二个值用于右上角和左下角，第三个用于右下角）：

两个值 - border-radius: 15px 50px;（第一个值用于左上角和右下角，第二个值用于右上角和左下角）：

一个值 - border-radius: 15px;（该值用于所有四个角，圆角都是一样的）：

这是代码：

实例

<!DOCTYPE html>

<html>

<head>

<style>

#rcorners1 {

border-radius: 15px 50px 30px 5px;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners2 {

border-radius: 15px 50px 30px;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners3 {

border-radius: 15px 50px;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners4 {

border-radius: 15px;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

</style>

</head>

<body>

<h1>border-radius 属性</h1>

<p>四个值 - border-radius: 15px 50px 30px 5px:</p>

<p id="rcorners1"></p>

<p>三个值 - border-radius: 15px 50px 30px:</p>

<p id="rcorners2"></p>

<p>两个值 - border-radius: 15px 50px:</p>

<p id="rcorners3"></p>

<p>一个值 - border-radius: 15px:</p>

<p id="rcorners4"></p>

</body>

</html>

您还可以创建椭圆角：

<!DOCTYPE html>

<html>

<head>

<style>

#rcorners1 {

border-radius: 50px / 15px;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners2 {

border-radius: 15px / 50px;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners3 {

border-radius: 50%;

background: #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

</style>

</head>

<body>

<h1>border-radius 属性</h1>

<p>椭圆边框 - border-radius: 50px / 15px:</p>

<p id="rcorners1"></p>

<p>椭圆边框 - border-radius: 15px / 50px:</p>

<p id="rcorners2"></p>

<p>椭圆边框 - border-radius: 50%:</p>

<p id="rcorners3"></p>

</body>

</html>

## 72.4 CSS 圆角属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| border-radius | 用于设置所有四个 border-\*-\*-radius 属性的简写属性。 |
| border-top-left-radius | 定义左上角边框的形状。 |
| border-top-right-radius | 定义右上角边框的形状。 |
| border-bottom-right-radius | 定义右下角边框的形状。 |
| border-bottom-left-radius | 定义左下角边框的形状。 |

# 第73章 CSS 边框图像

## 73.1 CSS border-image 属性

CSS border-image 属性允许您指定要使用的图像，而不是包围普通边框。

该属性有三部分：

用作边框的图像

在哪里裁切图像

定义中间部分应重复还是拉伸

我们将使用这幅图像（"border.png"）

border-image 属性接受图像，并将其切成九部分，就像井字游戏板。然后，将拐角放置在拐角处，并根据您的设置重复或拉伸中间部分。

注意：为了使 border-image 起作用，该元素还需要设置 border 属性！

此处，重复图像的中间部分以创建边框：

<!DOCTYPE html>

<html>

<head>

<style>

#borderimg {

border: 10px solid transparent;

padding: 15px;

border-image: url(border.png) 30 round;

}

</style>

</head>

<body>

<h1>border-image 属性</h1>

<p>在这里，重复图像的中间部分，来创建边框：</p>

<p id="borderimg">border-image: url(border.png) 30 round;</p>

<p>这是原始图像：</p><img src="border.png">

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 border-image 属性。</p>

</body>

</html>

此处，图像的中间部分被拉伸以创建边框：

<!DOCTYPE html>

<html>

<head>

<style>

#borderimg {

border: 10px solid transparent;

padding: 15px;

border-image: url(border.png) 30 stretch;

}

</style>

</head>

<body>

<h1>border-image 属性</h1>

<p>在这里，图像的中间部分被拉伸，来创建边框：</p>

<p id="borderimg">border-image: url(border.png) 30 stretch;</p>

<p>这是原始图像：</p><img src="border.png">

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 border-image 属性。</p>

</body>

</html>

提示：border-image 属性实际上是以下属性的简写属性：

border-image-source

border-image-slice

border-image-width

border-image-outset

border-image-repeat

## 73.2 CSS border-image - 不同的裁切值

不同的裁切值会完全改变边框的外观：

<!DOCTYPE html>

<html>

<head>

<style>

#borderimg1 {

border: 10px solid transparent;

padding: 15px;

border-image: url(border.png) 50 round;

}

#borderimg2 {

border: 10px solid transparent;

padding: 15px;

border-image: url(border.png) 20% round;

}

#borderimg3 {

border: 10px solid transparent;

padding: 15px;

border-image: url(border.png) 30% round;

}

</style>

</head>

<body>

<h1>border-image 属性</h1>

<p id="borderimg1">border-image: url(border.png) 50 round;</p>

<p id="borderimg2">border-image: url(border.png) 20% round;</p>

<p id="borderimg3">border-image: url(border.png) 30% round;</p>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 border-image 属性。</p>

</body>

</html>

## 73.3 CSS 边框图像属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| border-image | 用于设置所有 border-image-\* 属性的简写属性。 |
| border-image-source | 规定用作边框的图像的路径。 |
| border-image-slice | 规定如何裁切边框图像。 |
| border-image-width | 规定边框图像的宽度。 |
| border-image-outset | 规定边框图像区域超出边框盒的量。 |
| border-image-repeat | 规定边框图像应重复、圆角、还是拉伸。 |

# 第74章 CSS 多重背景

在本章中，您将学习如何将多个背景图像添加到一个元素。

您还将学到以下属性：

background-size

background-origin

background-clip

## 74.1 CSS 多重背景

CSS 允许您通过 background-image 属性为一个元素添加多幅背景图像。

不同的背景图像用逗号隔开，并且图像会彼此堆叠，其中的第一幅图像最靠近观看者。

下面的例子有两幅背景图像，第一幅图像是花朵（与底部和右侧对齐），第二幅图像是纸张背景（与左上角对齐）：

<!DOCTYPE html>

<html>

<head>

<style>

#example1 {

background-image: url(/i/photo/flower.gif), url(/i/paper.jpg);

background-position: right bottom, left top;

background-repeat: no-repeat, repeat;

padding: 15px;

}

</style>

</head>

<body>

<h1>多重背景</h1>

<p>下面的 div 元素有两副背景图像：</p>

<div id="example1">

<h1>Welcome to Shanghai</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

</body>

</html>

多重背景图像可以使用单独的背景属性（如上所述）或 background 简写属性来指定。

下面的例子使用 background 简写属性（结果与上例相同）：

实例

<!DOCTYPE html>

<html>

<head>

<style>

#example1 {

background: url(/i/photo/flower.gif) right bottom no-repeat, url(/i/paper.jpg) left top repeat;

padding: 15px;

}

</style>

</head>

<body>

<div id="example1">

<h1>Welcome to Shanghai</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

</body>

</html>

## 74.2 CSS 背景尺寸

CSS background-size 属性允许您指定背景图像的大小。

可以通过长度、百分比或使用以下两个关键字之一来指定背景图像的大小：contain 或 cover。

下面的例子将背景图像的大小调整为比原始图像小得多（使用像素）：

<!DOCTYPE html>

<html>

<head>

<style>

#example1 {

border: 1px solid black;

background: url(/i/photo/flower.gif);

background-size: 100px 80px;

background-repeat: no-repeat;

padding: 15px;

}

#example2 {

border: 1px solid black;

background: url(/i/photo/flower.gif);

background-repeat: no-repeat;

padding: 15px;

}

</style>

</head>

<body>

<h1>background-size 属性</h1>

<p>被调整大小的 background-image:</p>

<div id="example1">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

<p>background-image 的原始尺寸：</p>

<div id="example2">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

</body>

</html>

ackground-size 的其他两个可能的值是 contain 和 cover。

contain 关键字将背景图像缩放为尽可能大的尺寸（但其宽度和高度都必须适合内容区域）。这样，取决于背景图像和背景定位区域的比例，可能存在一些未被背景图像覆盖的背景区域。

cover 关键字会缩放背景图像，以使内容区域完全被背景图像覆盖（其宽度和高度均等于或超过内容区域）。这样，背景图像的某些部分可能在背景定位区域中不可见。

下面的例子展示了 contain 和 cover 的用法：

<!DOCTYPE html>

<html>

<head>

<style>

.div1 {

border: 1px solid black;

height: 120px;

width: 150px;

background: url(/i/photo/flower.gif);

background-repeat: no-repeat;

background-size: contain;

}

.div2 {

border: 1px solid black;

height: 120px;

width: 150px;

background: url(/i/photo/flower.gif);

background-repeat: no-repeat;

background-size: cover;

}

.div3 {

border: 1px solid black;

height: 120px;

width: 150px;

background: url(/i/photo/flower.gif);

background-repeat: no-repeat;

}

</style>

</head>

<body>

<h1>background-size 属性</h1>

<h2>background-size: contain:</h2>

<div class="div1">

<p>Lorem ipsum dolor sit amet.</p>

</div>

<h2>background-size: cover:</h2>

<div class="div2">

<p>Lorem ipsum dolor sit amet.</p>

</div>

<h2>background-size 未定义：</h2>

<div class="div3">

<p>Lorem ipsum dolor sit amet.</p>

</div>

<p>原始图像：</p>

<img src="/i/photo/flower.gif" alt="Flowers" width="224" height="162">

</body>

</html>

## 74.3 定义多个背景图像的尺寸

在处理多重背景时，background-size 属性还可以接受多个设置背景尺寸的值（使用逗号分隔的列表）。

下面的例子指定了三幅背景图像，每幅图像有不同的 background-size 值：

<!DOCTYPE html>

<html>

<head>

<style>

#example1 {

background: url(/i/photo/tree.png) left top no-repeat, url(/i/photo/flower.gif) right bottom no-repeat, url(/i/paper.jpg) left top repeat;

padding: 15px;

background-size: 50px, 130px, auto;

}

</style>

</head>

<body>

<div id="example1">

<h1>Welcome to Shanghai</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

</body>

</html>

## 74.4 全尺寸背景图像

现在，我们希望网站上的背景图像始终覆盖整个浏览器窗口。

具体要求如下：

用图像填充整个页面（无空白）

根据需要缩放图像

在页面上居中图像

不引发滚动条

下面的例子显示了如何实现它：使用 <html> 元素（<html> 元素始终至少是浏览器窗口的高度）。然后在其上设置固定且居中的背景。最后使用 background-size 属性调整其大小：

<!DOCTYPE html>

<html>

<head>

<style>

html {

background: url(/i/photo/tiyugongyuan.jpg) no-repeat center fixed;

background-size: cover;

}

body {

color: white;

}

</style>

</head>

<body>

<h1>完整页面的背景图像</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai! The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</body>

</html>

## 74.5 Hero Image

您还可以在 <div> 上使用不同的背景属性来创建 Hero Image（带有文本的大图像），并将其放置在您希望的位置上。

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body {

margin: 0;

font-family: Arial, Helvetica, sans-serif;

}

.hero-image {

background: url(/i/photo/tiyugongyuan.jpg) no-repeat center;

background-size: cover;

height: 500px;

position: relative;

}

.hero-text {

text-align: center;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

color: white;

}

</style>

</head>

<body>

<div class="hero-image">

<div class="hero-text">

<h1 style="font-size:50px">我是比尔盖茨</h1>

<h3>我是一位摄影师</h3>

<button>聘请我吧</button>

</div>

</div>

<p>页面内容..</p>

<p>请注意，这项技术会将使图像响应：请调整浏览器窗口的大小来查看效果。</p>

</body>

</html>

## 74.6 CSS background-origin 属性

CSS background-origin 属性指定背景图像的位置。

该属性接受三个不同的值：

border-box - 背景图片从边框的左上角开始

padding-box -背景图像从内边距边缘的左上角开始（默认）

content-box - 背景图片从内容的左上角开始

下面的示例展示了 background-origin 属性：

<!DOCTYPE html>

<html>

<head>

<style>

#example1 {

border: 10px solid black;

padding: 35px;

background: url(/i/photo/flower.gif);

background-repeat: no-repeat;

}

#example2 {

border: 10px solid black;

padding: 35px;

background: url(/i/photo/flower.gif);

background-repeat: no-repeat;

background-origin: border-box;

}

#example3 {

border: 10px solid black;

padding: 35px;

background: url(/i/photo/flower.gif);

background-repeat: no-repeat;

background-origin: content-box;

}

</style>

</head>

<body>

<h1>background-origin 属性</h1>

<p>未设置 background-origin (padding-box 为默认):</p>

<div id="example1">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

<p>background-origin: border-box:</p>

<div id="example2">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

<p>background-origin: content-box:</p>

<div id="example3">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>The city is located on the southern estuary of the Yangtze, with the Huangpu River flowing through it.</p>

</div>

</body>

</html>

## 74.7 CSS background-clip 属性

CSS background-clip 属性指定背景的绘制区域。

该属性接受三个不同的值：

border-box - 背景绘制到边框的外部边缘（默认）

padding-box - 背景绘制到内边距的外边缘

content-box - 在内容框中绘制背景

下面的例子展示了 background-clip 属性：

<!DOCTYPE html>

<html>

<head>

<style>

#example1 {

border: 10px dotted black;

padding: 35px;

background: yellow;

}

#example2 {

border: 10px dotted black;

padding: 35px;

background: yellow;

background-clip: padding-box;

}

#example3 {

border: 10px dotted black;

padding: 35px;

background: yellow;

background-clip: content-box;

}

</style>

</head>

<body>

<h1>background-clip 属性</h1>

<p>No background-clip (border-box is default):</p>

<div id="example1">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

<p>background-clip: padding-box:</p>

<div id="example2">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

<p>background-clip: content-box:</p>

<div id="example3">

<h2>Welcome to Shanghai</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

</body>

</html>

## 74.8 CSS 高级背景属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| background | 用于在一条声明中设置所有背景属性的简写属性。 |
| background-clip | 规定背景的绘制区域。 |
| background-image | 为一个元素指定一幅或多幅背景图像。 |
| background-origin | 规定背景图像的放置位置。 |
| background-size | 规定背景图像的大小。 |

# 第75章 CSS 颜色

## 75.1 RGBA 颜色

RGBA 颜色值是 RGB 颜色值的扩展，带有 alpha 通道 - 该通道规定颜色的不透明度。

RGBA 颜色值是这样规定的：rgba(red, green, blue, alpha)。 alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字。

<!DOCTYPE html>

<html>

<head>

<style>

#p1 {background-color:rgba(255,0,0,0.3);}

#p2 {background-color:rgba(0,255,0,0.3);}

#p3 {background-color:rgba(0,0,255,0.3);}

#p4 {background-color:rgba(192,192,192,0.3);}

#p5 {background-color:rgba(255,255,0,0.3);}

#p6 {background-color:rgba(255,0,255,0.3);}

</style>

</head>

<body>

<h1>用 RGBA 值定义颜色</h1>

<p id="p1">Red</p>

<p id="p2">Green</p>

<p id="p3">Blue</p>

<p id="p4">Grey</p>

<p id="p5">Yellow</p>

<p id="p6">Cerise</p>

</body>

</html>

## 75.2 HSL 颜色

HSL 指的是色相、饱和度和亮度（Hue、Saturation 以及 Lightness）。

HSL 颜色值是这样规定的：hsl(hue, saturation, lightness)。

色相是色轮上的度数（从 0 到 360）：

0（或 360）是红色

120 是绿色

240 是蓝色

饱和度是一个百分比值：100％ 是全色。

亮度也是一个百分比值：0％ 是深色（黑色），而 100％ 是白色。

<!DOCTYPE html>

<html>

<head>

<style>

#p1 {background-color:hsl(120,100%,50%);}

#p2 {background-color:hsl(120,100%,75%);}

#p3 {background-color:hsl(120,100%,25%);}

#p4 {background-color:hsl(120,60%,70%);}

#p5 {background-color:hsl(290,100%,50%);}

#p6 {background-color:hsl(290,60%,70%);}

</style>

</head>

<body>

<h1>用 HSL 值定义颜色</h1>

<p id="p1">绿色</p>

<p id="p2">浅绿色</p>

<p id="p3">深绿色</p>

<p id="p4">淡绿色</p>

<p id="p5">紫色</p>

<p id="p6">淡紫色</p>

</body>

</html>

## 75.3 HSLA 颜色

HSLA 颜色值是带有 Alpha 通道的 HSL 颜色值的扩展 - 它规定了颜色的不透明度。

HSLA 颜色值由以下参数规定：hsla(hue, saturation, lightness, alpha)，其中 alpha 参数定义不透明度。 alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字。

<!DOCTYPE html>

<html>

<head>

<style>

#p1 {background-color:hsla(120,100%,50%,0.3);}

#p2 {background-color:hsla(120,100%,75%,0.3);}

#p3 {background-color:hsla(120,100%,25%,0.3);}

#p4 {background-color:hsla(120,60%,70%,0.3);}

#p5 {background-color:hsla(290,100%,50%,0.3);}

#p6 {background-color:hsla(290,60%,70%,0.3);}

</style>

</head>

<body>

<h1>用 HSLA 值定义颜色</h1>

<p id="p1">绿色</p>

<p id="p2">浅绿色</p>

<p id="p3">深绿色</p>

<p id="p4">淡绿色</p>

<p id="p5">紫色</p>

<p id="p6">淡紫色</p>

</body>

</html>

## 75.4 不透明度

CSS opacity 属性设置整个元素的不透明度（背景颜色和文本都将是不透明/透明的）。

opacity 属性值必须是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字。

<!DOCTYPE html>

<html>

<head>

<style>

#p1 {background-color:rgb(255,0,0);opacity:0.6;}

#p2 {background-color:rgb(0,255,0);opacity:0.6;}

#p3 {background-color:rgb(0,0,255);opacity:0.6;}

#p4 {background-color:rgb(192,192,192);opacity:0.6;}

#p5 {background-color:rgb(255,255,0);opacity:0.6;}

#p6 {background-color:rgb(255,0,255);opacity:0.6;}

</style>

</head>

<body>

<h1>用 opacity 定义颜色</h1>

<p id="p1">红色</p>

<p id="p2">绿色</p>

<p id="p3">蓝色</p>

<p id="p4">灰色</p>

<p id="p5">黄色</p>

<p id="p6">樱桃色</p>

</body>

</html>

# 第76章 CSS 渐变

CSS 渐变使您可以显示两种或多种指定颜色之间的平滑过渡。

CSS 定义了两种渐变类型：

线性渐变（向下/向上/向左/向右/对角线）

径向渐变（由其中心定义）

## 76.1 CSS 线性渐变

如需创建线性渐变，您必须定义至少两个色标。色标是您要呈现平滑过渡的颜色。您还可以设置起点和方向（或角度）以及渐变效果。

语法

background-image: linear-gradient(direction, color-stop1, color-stop2, ...);

**线性渐变 - 从上到下（默认）**

下面的例子显示了从顶部开始的线性渐变。它从红色开始，过渡到黄色：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(red, yellow);

}

</style>

</head>

<body>

<h1>线性渐变 - 从上到下</h1>

<p>此线性渐变从顶部开始为红色，然后在底部过渡为黄色：</p>

<div id="grad1"></div>

</body>

</html>

**线性渐变 - 从左到右**

下面的例子展示了从左开始的线性渐变。它从红色开始，过渡到黄色：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(to right, red , yellow);

}

</style>

</head>

<body>

<h1>线性渐变 - 从左到右</h1>

<p>此线性渐变从左开始为红色，然后过渡到黄色（向右）：</p>

<div id="grad1"></div>

</body>

</html>

**线性渐变 - 对角线**

您可以通过指定水平和垂直起始位置来实现对角渐变。

下面的例子展示了从左上角开始（到右下角）的线性渐变。它从红色开始，过渡到黄色：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(to bottom right, red, yellow);

}

</style>

</head>

<body>

<h1>线性渐变 - 对角线</h1>

<p>此线性渐变从左上开始为红色，然后过渡到黄色（到右下）：</p>

<div id="grad1"></div>

</body>

</html>

## 76.2 使用角度

如果希望对渐变角度做更多的控制，您可以定义一个角度，来取代预定义的方向（向下、向上、向右、向左、向右下等等）。值 0deg 等于向上（to top）。值 90deg 等于向右（to right）。值 180deg 等于向下（to bottom）。

语法

background-image: linear-gradient(angle, color-stop1, color-stop2);

这个角度指定水平线和渐变线之间的角度。

下面的例子展示了如何在线性渐变上使用角度：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 100px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(0deg, red, yellow);

}

#grad2 {

height: 100px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(90deg, red, yellow);

}

#grad3 {

height: 100px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(180deg, red, yellow);

}

#grad4 {

height: 100px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(-90deg, red, yellow);

}

</style>

</head>

<body>

<h1>线性渐变 - 使用不同的角度</h1>

<div id="grad1" style="text-align:center;">0deg</div><br>

<div id="grad2" style="text-align:center;">90deg</div><br>

<div id="grad3" style="text-align:center;">180deg</div><br>

<div id="grad4" style="text-align:center;">-90deg</div>

</body>

</html>

## 76.3 使用多个色标

下面的例子展示了带有多个色标的线性渐变（从上到下）：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(red, yellow, green);

}

#grad2 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(red, orange, yellow, green, blue, indigo, violet);

}

#grad3 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(red 10%, green 85%, blue 90%);

}

</style>

</head>

<body>

<h1>线性渐变 - 多个色标</h1>

<p><b>注释：</b>如果不规定百分百，则会均匀分布色标。</p>

<h2>三个色标 (evenly spaced)：</h2>

<div id="grad1"></div>

<h2>七个色标 (evenly spaced)：</h2>

<div id="grad2"></div>

<h2>三个色标 (not evenly spaced):</h2>

<div id="grad3"></div>

</body>

</html>

下面的例子展示了如何使用彩虹色和一些文本创建线性渐变（从左到右）：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 55px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);

}

</style>

</head>

<body>

<div id="grad1" style="text-align:center;margin:auto;color:#888888;font-size:40px;font-weight:bold">

彩虹背景

</div>

</body>

</html>

## 76.4 使用透明度

CSS 渐变还支持透明度，也可用于创建渐变效果。

如需添加透明度，我们使用 rgba() 函数来定义色标。 rgba() 函数中的最后一个参数可以是 0 到 1 的值，它定义颜色的透明度：0 表示全透明，1 表示全彩色（无透明）。

下面的例子展示了从左开始的线性渐变。它开始完全透明，然后过渡为全色红色：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));

}

</style>

</head>

<body>

<h1>线性渐变 - 透明度</h1>

<p>为了增加透明度，我们使用 rgba() 函数定义色标。 rgba() 函数中的最后一个参数可以是 0 到 1 的值，它定义颜色的透明度：0 表示全透明，1 表示全色（不透明）。</p>

<div id="grad1"></div>

</body>

</html>

## 76.5 重复线性渐变

repeating-linear-gradient() 函数用于重复线性渐变：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: repeating-linear-gradient(red, yellow 10%, green 20%);

}

#grad2 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: repeating-linear-gradient(45deg,red,yellow 7%,green 10%);

}

#grad3 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: repeating-linear-gradient(190deg,red,yellow 7%,green 10%);

}

#grad4 {

height: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: repeating-linear-gradient(90deg,red,yellow 7%,green 10%);

}

</style>

</head>

<body>

<h1>重复线性渐变</h1>

<div id="grad1"></div>

<p>在 45deg 轴上重复的渐变，从红色开始到绿色结束：</p>

<div id="grad2"></div>

<p>在 190deg 轴上重复的渐变，从红色开始到绿色结束：</p>

<div id="grad3"></div>

<p>在 90deg 轴上重复的渐变，从红色开始到绿色结束：</p>

<div id="grad4"></div>

</body>

</html>

# 第77章 CSS 径向渐变

## 77.1 CSS 径向渐变

径向渐变由其中心定义。

如需创建径向渐变，您还必须定义至少两个色标。

语法

background-image: radial-gradient(shape size at position, start-color, ..., last-color);

默认地，shape 为椭圆形，size 为最远角，position 为中心。

## 77.2 径向渐变-均匀间隔的色标（默认）

下面的例子展示了带有均匀间隔的色标的径向渐变：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 150px;

width: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(red, yellow, green);

}

</style>

</head>

<body>

<h1>径向渐变 - 均匀间隔的色标</h1>

<div id="grad1"></div>

</body>

</html>

## 77.3 径向渐变-不同间距的色标

下面的例子展示了一个径向渐变，其色标之间的间隔不同：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 150px;

width: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(red 5%, yellow 15%, green 60%);

}

</style>

</head>

<body>

<h1>径向渐变 - 不均匀间隔的色标</h1>

<div id="grad1"></div>

</body>

</html>

## 77.4 设置形状

shape 参数定义形状。它可接受 circle 或 ellipse 值。默认值为 ellipse（椭圆）。

下面的例子展示了一个圆形的径向渐变：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 150px;

width: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(red, yellow, green);

}

#grad2 {

height: 150px;

width: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(circle, red, yellow, green);

}

</style>

</head>

<body>

<h1>径向渐变 - 形状</h1>

<h2>椭圆（默认）：</h2>

<div id="grad1"></div>

<h2>圆：</h2>

<div id="grad2"></div>

</body>

</html>

## 77.5 使用大小不同的关键字

size 参数定义渐变的大小。它可接受四个值：

closest-side

farthest-side

closest-corner

farthest-corner

实例

设置了不同 size 关键词的径向渐变：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 150px;

width: 150px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(closest-side at 60% 55%, red, yellow, black);

}

#grad2 {

height: 150px;

width: 150px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(farthest-side at 60% 55%, red, yellow, black);

}

#grad3 {

height: 150px;

width: 150px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(closest-corner at 60% 55%, red, yellow, black);

}

#grad4 {

height: 150px;

width: 150px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: radial-gradient(farthest-corner at 60% 55%, red, yellow, black);

}

</style>

</head>

<body>

<h1>径向渐变 - 不同的 size 关键字s</h1>

<h2>closest-side:</h2>

<div id="grad1"></div>

<h2>farthest-side:</h2>

<div id="grad2"></div>

<h2>closest-corner:</h2>

<div id="grad3"></div>

<h2>farthest-corner (默认):</h2>

<div id="grad4"></div>

</body>

</html>

## 77.6 重复径向渐变

repeating-radial-gradient() 函数用于重复径向渐变：

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 150px;

width: 200px;

background-color: red; /\* 针对不支持渐变的浏览器 \*/

background-image: repeating-radial-gradient(red, yellow 10%, green 15%);

}

</style>

</head>

<body>

<h1>重复的径向渐变</h1>

<div id="grad1"></div>

</body>

</html>

## 77.7 CSS 渐变属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| background-image | 为一个元素设置一幅或多幅背景图像。 |

# 第78章 CSS 阴影效果

## 78.1 CSS 阴影效果

通过使用 CSS，您可以在文本和元素上添加阴影。

在我们的教程中，您将学习如下属性：

text-shadow

box-shadow

## 78.2 CSS 文字阴影

CSS text-shadow 属性为文本添加阴影。

最简单的用法是只指定水平阴影（2px）和垂直阴影（2px）：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 2px 2px;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

接下来，为阴影添加颜色：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 2px 2px red;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

然后，向阴影添加模糊效果：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 2px 2px 5px red;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

下面的例子展示了带有黑色阴影的白色文本：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

color: white;

text-shadow: 2px 2px 4px #000000;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

下面的例子展示了红色的霓虹发光阴影：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 0 0 3px #FF0000;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

## 78.3 多个阴影

如需在文本中添加多个阴影，您可以添加以逗号分隔的阴影列表。

下面的例子展示了红色和蓝色的霓虹灯发光阴影：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

下面的例子展示了带有黑色、蓝色和深蓝色阴影的白色文本：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

color: white;

text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

您还可以使用 text-shadow 属性在文本周围创建纯边框（无阴影）：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

color: yellow;

text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;

}

</style>

</head>

<body>

<h1>围绕文本的边框！</h1>

</body>

</html>

# 第79章 CSS Box Shadow

## 79.1 CSS box-shadow 属性

CSS box-shadow 属性应用阴影于元素。

最简单的用法是只指定水平阴影和垂直阴影：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

padding: 15px;

background-color: yellow;

box-shadow: 10px 10px;

}

</style>

</head>

<body>

<h1>box-shadow 属性</h1>

<div>已设置 box-shadow 的 div 元素</div>

</body>

</html>

接下来，为阴影添加颜色：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

padding: 15px;

background-color: yellow;

box-shadow: 10px 10px grey;

}

</style>

</head>

<body>

<div>已设置 box-shadow 的 div 元素</div>

</body>

</html>

接下来，向阴影添加模糊效果：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

padding: 15px;

background-color: yellow;

box-shadow: 10px 10px 5px grey;

}

</style>

</head>

<body>

<div>已设置 box-shadow 的 div 元素</div>

</body>

</html>

## 79.2 卡片

您还可以使用 box-shadow 属性创建纸质卡片效果：

<!DOCTYPE html>

<html>

<head>

<style>

div.card {

width: 250px;

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);

text-align: center;

}

div.header {

background-color: #4CAF50;

color: white;

padding: 10px;

font-size: 40px;

}

div.container {

padding: 10px;

}

</style>

</head>

<body>

<h2>Cards</h2>

<p>box-shadow 属性可用于创建类似纸质的卡片：</p>

<div class="card">

<div class="header">

<h1>1</h1>

</div>

<div class="container">

<p>January 1, 2021</p>

</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

div.polaroid {

width: 250px;

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);

text-align: center;

}

div.container {

padding: 10px;

}

</style>

</head>

<body>

<h1>宝丽来图像 / 卡片</h1>

<p>box-shadow 属性可用于创建类似纸质的卡片：</p>

<div class="polaroid">

<img src="/i/photo/coffee.jpg" alt="Shanghai" style="width:100%">

<div class="container">

<p>Shanghai, China</p>

</div>

</div>

</body>

</html>

## 79.3 CSS 阴影属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| box-shadow | 向一个元素添加一个或多个阴影。 |
| text-shadow | 在文本中添加一个或多个阴影。 |

# 第80章 CSS 文本效果

CSS 文本溢出、整字换行、换行规则以及书写模式

在本章中，您将学习如下属性：

text-overflow

word-wrap

word-break

writing-mode

## 80.1 CSS 文字溢出

CSS text-overflow 属性规定应如何向用户呈现未显示的溢出内容。

可以被裁剪：

也可以将其呈现为省略号（...）

<!DOCTYPE html>

<html>

<head>

<style>

p.test1 {

white-space: nowrap;

width: 200px;

border: 1px solid #000000;

overflow: hidden;

text-overflow: clip;

}

p.test2 {

white-space: nowrap;

width: 200px;

border: 1px solid #000000;

overflow: hidden;

text-overflow: ellipsis;

}

</style>

</head>

<body>

<h1>text-overflow 属性</h1>

<p>以下两段包含不适合其框的长文本。</p>

<h2>text-overflow: clip:</h2>

<p class="test1">这里有一些无法容纳在框中的长文本</p>

<h2>text-overflow: ellipsis:</h2>

<p class="test2">这里有一些无法容纳在框中的长文本</p>

</body>

</html>

下面的例子展示了将鼠标悬停在元素上时如何显示溢出的内容：

<!DOCTYPE html>

<html>

<head>

<style>

div.test {

white-space: nowrap;

width: 200px;

overflow: hidden;

border: 1px solid #000000;

}

div.test:hover {

overflow: visible;

}

</style>

</head>

<body>

<p>请把鼠标移动到下面的 div 上，来查看完整文本。</p>

<div class="test" style="text-overflow:ellipsis;">这里有一些无法容纳在框中的长文本</div>

<br>

<div class="test" style="text-overflow:clip;">这里有一些无法容纳在框中的长文本</div>

</body>

</html>

## 80.2 CSS 字换行（Word Wrapping）

CSS word-wrap 属性使长文字能够被折断并换到下一行。

如果一个单词太长而无法容纳在一个区域内，它会向外扩展：

通过 word-wrap 属性，您可以强制对文本进行换行 - 即使这意味着在词中间将其拆分：

<!DOCTYPE html>

<html>

<head>

<style>

p.test {

width: 11em;

border: 1px solid #000000;

word-wrap: break-word;

}

</style>

</head>

<body>

<h1>word-wrap 属性</h1>

<p class="test">This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.</p>

</body>

</html>

## 80.3 CSS 换行规则

CSS word-break 属性指定换行规则。

本段包含一些文本。此行将连字符打断：

本段包含一些文本。这些行将在任何字符处中断：

<!DOCTYPE html>

<html>

<head>

<style>

p.test1 {

width: 140px;

border: 1px solid #000000;

word-break: keep-all;

}

p.test2 {

width: 140px;

border: 1px solid #000000;

word-break: break-all;

}

</style>

</head>

<body>

<h1>word-break 属性</h1>

<p class="test1">This paragraph contains some text. This line will-break-at-hyphens.</p>

<p class="test2">This paragraph contains some text. The lines will break at any character.</p>

<p><b>注释：</b>Opera 12 和更早版本不支持 word-break 属性。</p>

</body>

</html>

## 80.4 CSS 书写模式

CSS writing-mode 属性规定文本行是水平放置还是垂直放置。

Some text with a span element with a vertical-rl writing-mode.

下面的例子展示了一些不同的书写模式：

<!DOCTYPE html>

<html>

<head>

<style>

p.test1 {

writing-mode: horizontal-tb;

}

span.test2 {

writing-mode: vertical-rl;

}

p.test2 {

writing-mode: vertical-rl;

}

</style>

</head>

<body>

<h1>writing-mode 属性</h1>

<p class="test1">Some text with default writing-mode.</p>

<p>Some text with a span element with a <span class="test2">vertical-rl</span> writing-mode.</p>

<p class="test2">Some text with writing-mode: vertical-rl.</p>

</body>

</html>

## 80.5 CSS 文本效果属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| text-align-last | 指定如何对齐文本的最后一行。 |
| text-justify | 指定对齐的文本应如何对齐和间隔。 |
| text-overflow | 指定应如何向用户呈现未显示的溢出内容。 |
| word-break | 指定非 CJK 脚本的换行规则。 |
| word-wrap | 允许长单词被打断并换到下一行。 |
| writing-mode | 指定文本行是水平放置还是垂直放置。 |

# 第81章 CSS Web 字体

## 81.1 CSS @font-face 规则

Web 字体允许 Web 设计人员使用用户计算机上未安装的字体。

当您找到/购买了想要使用的字体后，只需将字体文件包含在您的 Web 服务器上，它将在需要时自动下载给用户。

您的“自有”字体在 CSS @font-face 规则中进行定义。

## 81.2 不同的字体格式

**TrueType 字体 (TTF)**

TrueType 是 1980 年代后期由 Apple 和 Microsoft 开发的字体标准。 TrueType 是 Mac OS 和 Microsoft Windows 操作系统最常用的字体格式。

**OpenType 字体 (OTF)**

OpenType 是可缩放计算机字体的格式。它基于 TrueType 构建，并且是 Microsoft 的注册商标。今天，OpenType 字体在主要计算机平台上得到普遍使用。

**Web 开放字体格式 (WOFF)**

WOFF 是用于网页的字体格式。它于 2009 年开发，现已成为 W3C 的推荐标准。 WOFF 本质上是具有压缩和其他元数据的 OpenType 或 TrueType。目标是支持在有带宽限制的网络上从服务器到客户端进行字体分发。

**Web 开放字体格式 (WOFF 2.0)**

TrueType/OpenType 字体比 WOFF 1.0 提供更好的压缩。

**SVG 字体/形状**

SVG 字体允许在显示文本时将 SVG 用作字形。 SVG 1.1 规范定义了一个字体模块，该模块允许在 SVG 文档中创建字体。您还可以将 CSS 应用于 SVG 文档，同时 @font-face 规则可以应用于 SVG 文档中的文本。

**嵌入式 OpenType 字体 (EOT)**

EOT 字体是 Microsoft 设计的 OpenType 字体的紧凑形式，用作网页上的嵌入式字体。

\*IE：该字体格式仅在设置为 "installable" 时有效。

\*Firefox：默认情况下不支持，但可以启用（需要将标志设置为 "true" 才能使用 WOFF2）。

## 81.3 使用您需要的字体

在 @font-face 规则中：首先定义字体的名称（例如 myFirstFont），然后指向该字体文件。

提示：字体 URL 始终使用小写字母。大写字母可能会在 IE 中产生意外结果。

如需将字体用于 HTML 元素，请通过 font-family 属性引用字体名称（myFirstFont）：

## 81.4 使用您需要的字体

在 @font-face 规则中：首先定义字体的名称（例如 myFirstFont），然后指向该字体文件。

提示：字体 URL 始终使用小写字母。大写字母可能会在 IE 中产生意外结果。

如需将字体用于 HTML 元素，请通过 font-family 属性引用字体名称（myFirstFont）：

<!DOCTYPE html>

<html>

<head>

<style>

@font-face {

font-family: myFirstFont;

src: url(sansation\_light.woff);

}

\* {

font-family: myFirstFont;

}

</style>

</head>

<body>

<h1>@font-face 规则</h1>

<div>

通过 CSS，网站可以使用<b>预选的 "web-safe" 字体以外的其他字体</b>。

</div>

<p><b>注释：</b>Internet Explorer 8 以及更早的版本不支持 @font-face 规则。</p>

</body>

</html>

## 81.5 使用粗体文本

您必须添加另一条 @font-face 规则，其中包含粗体文本的描述符：

<!DOCTYPE html>

<html>

<head>

<style>

@font-face {

font-family: myFirstFont;

src: url(sansation\_light.woff);

}

@font-face {

font-family: myFirstFont;

src: url(sansation\_bold.woff);

font-weight: bold;

}

\* {

font-family: myFirstFont;

}

</style>

</head>

<body>

<h1>@font-face 规则</h1>

<div>

通过 CSS，网站可以使用<b>预选的 "web-safe" 字体以外的其他字体</b>。

</div>

<p><b>注释：</b>Internet Explorer 8 以及更早的版本不支持 @font-face 规则。</p>

</body>

</html>

文件 "sansation\_bold.woff" 是另一个字体文件，其中包含 Sansation 字体的粗体字符。

每当带有 "myFirstFont" 字体族的一段文本应呈现粗体时，浏览器都会使用它。

这样，您就可以为同一字体设置许多 @font-face 规则。

## 81.6 CSS 字体描述

下表列出了能够在 @font-face 规则内定义的所有字体描述符（font descriptor）：

|  |  |  |
| --- | --- | --- |
| 描述符 | 值 | 描述 |
| font-family | name | 必需。定义字体名称。 |
| src | URL | 必需。定义字体文件的 URL。 |
| font-stretch | normal  condensed  ultra-condensed  extra-condensed  semi-condensed  expanded  semi-expanded  extra-expanded  ultra-expanded | 可选。定义应如何拉伸字体。默认值是 "normal"。 |
| font-style | normal  italic  oblique | 可选。定义字体的样式。默认值是 "normal"。 |
| font-weight | normal  bold  100  200  300  400  500  600  700  800  900 | 可选。定义字体的粗细。默认值是 "normal"。 |
| unicode-range | unicode-range | 可选。定义字体支持的 UNICODE 字符范围。默认值是 "U+0-10FFFF"。 |
|  |  |  |

# 第82章 CSS 2D 转换

## 82.1 CSS 2D 转换

CSS 转换（transforms）允许您移动、旋转、缩放和倾斜元素。

把鼠标悬停在下面的元素上，可以查看 2D 转换：

在本章中，您将学习如下 CSS 属性：

Transform

## 82.2 CSS 2D 转换方法

通过使用 CSS transform 属性，您可以利用以下 2D 转换方法：

translate()

rotate()

scaleX()

scaleY()

scale()

skewX()

skewY()

skew()

matrix()

提示：您将在下一章中学习 3D 转换。

## 82.3 translate() 方法

translate() 方法从其当前位置移动元素（根据为 X 轴和 Y 轴指定的参数）。

下面的例子把 <div> 元素从其当前位置向右移动 50 个像素，并向下移动 100 个像素：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: translate(50px,100px); /\* IE 9 \*/

transform: translate(50px,100px); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>translate() 方法</h1>

<p>translate() 方法从元素当前位置对其进行移动：</p>

<div>

该 div 元素从其当前位置向右移动 50 个像素，并向下移动 100 个像素。

</div>

</body>

</html>

## 82.4 rotate() 方法

rotate() 方法根据给定的角度顺时针或逆时针旋转元素。

下面的例子把 <div> 元素顺时针旋转 20 度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: rotate(20deg); /\* IE 9 \*/

transform: rotate(20deg); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>rotate() 方法</h1>

<p>rotation() 方法顺时针或逆时针旋转元素。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

这个 div 元素顺时针旋转 20 度。

</div>

</body>

</html>

使用负值将逆时针旋转元素。

下面的例子把 <div> 元素逆时针旋转 20 度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: rotate(-20deg); /\* IE 9 \*/

transform: rotate(-20deg); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>rotate() 方法</h1>

<p>rotation() 方法顺时针或逆时针旋转元素。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

这个 div 元素逆时针旋转 20 度。

</div>

</body>

</html>

## 82.5 scale() 方法

scale() 方法增加或减少元素的大小（根据给定的宽度和高度参数）。

下面的例子把 <div> 元素增大为其原始宽度的两倍和其原始高度的三倍：

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scale(2,3); /\* IE 9 \*/

transform: scale(2,3); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>scale() 方法</h1>

<p>scale() 方法增加或缩减元素的尺寸。</p>

<div>

该 div 元素是其原始宽度的两倍，是其原始高度的三倍。

</div>

</body>

</html>

下面的例子把 <div> 元素减小为其原始宽度和高度的一半：

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scale(0.5,0.5); /\* IE 9 \*/

transform: scale(0.5,0.5); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>scale() 方法</h1>

<p>scale() 方法增加或缩减元素的尺寸。</p>

<div>

该 div 元素减小到其原始宽度和高度的一半。

</div>

</body>

</html>

## 82.6 scaleX() 方法

scaleX() 方法增加或减少元素的宽度。

下面的例子把 <div> 元素增大为其原始宽度的两倍：

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scaleX(2); /\* IE 9 \*/

transform: scaleX(2); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>scaleX() 方法</h1>

<p>scaleX() 方法增加或减少元素的宽度。</p>

<div>

该 div 元素是其原始宽度的两倍。

</div>

</body>

</html>

以下例子把 <div> 元素缩减为其原始宽度的一半：

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scaleX(0.5); /\* IE 9 \*/

transform: scaleX(0.5); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>scaleX() 方法</h1>

<p>scaleX() 方法增加或减少元素的宽度。</p>

<div>

该 div 元素是其原始宽度的一半。

</div>

</body>

</html>

## 82.7 scaleY() 方法

scaleY() 方法增加或减少元素的高度。

下面的例子把 <div> 元素增大到其原始高度的三倍：

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scaleY(3); /\* IE 9 \*/

transform: scaleY(3); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>scaleY() 方法</h1>

<p>scaleY() 方法增加或减少元素的高度。</p>

<div>

该 div 元素是其原始高度的三倍。

</div>

</body>

</html>

下面的例子把 <div> 元素缩减为其原始高度的一半：

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scaleY(0.5); /\* IE 9 \*/

transform: scaleY(0.5); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>scaleY() 方法</h1>

<p>scaleY() 方法增加或减少元素的高度。</p>

<div>

该div元素是其原始高度的一半。

</div>

</body>

</html>

## 82.8 skewX() 方法

skewX() 方法使元素沿 X 轴倾斜给定角度。

下例把 <div> 元素沿X轴倾斜 20 度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: skewX(20deg); /\* IE 9 \*/

transform: skewX(20deg); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>skewX() 方法</h1>

<p>skewX() 方法使元素沿 X 轴倾斜给定角度。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

该 div 元素沿 X 轴倾斜 20 度。

</div>

</body>

</html>

## 82.8 skewY() 方法

skewY() 方法使元素沿 Y 轴倾斜给定角度。

下例把 <div> 元素沿 Y 轴倾斜 20 度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: skewY(20deg); /\* IE 9 \*/

transform: skewY(20deg); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>skewY() 方法</h1>

<p>skewY() 方法使元素沿 Y 轴倾斜给定角度。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

该 div 元素沿 Y 轴倾斜 20 度。

</div>

</body>

</html>

## 82.9 skew() 方法

skew() 方法使元素沿 X 和 Y 轴倾斜给定角度。

下面的例子使 <div> 元素沿 X 轴倾斜 20 度，同时沿 Y 轴倾斜 10 度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: skew(20deg,10deg); /\* IE 9 \*/

transform: skew(20deg,10deg); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>skew() 方法</h1>

<p>skew() 方法将元素倾斜到给定角度。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

该 div 元素沿 X 轴倾斜 20 度，沿 Y 轴倾斜 10 度。

</div>

</body>

</html>

如果未指定第二个参数，则值为零。因此，下例使 <div> 元素沿 X 轴倾斜 20 度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: skew(20deg); /\* IE 9 \*/

transform: skew(20deg); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>skew() 方法</h1>

<p>skew() 方法将元素倾斜到给定角度。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

该 div 元素沿 X 轴倾斜 20 度。

</div>

</body>

</html>

## 82.10 matrix() 方法

matrix() 方法把所有 2D 变换方法组合为一个。

matrix() 方法可接受六个参数，其中包括数学函数，这些参数使您可以旋转、缩放、移动（平移）和倾斜元素。

参数如下：matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv1 {

-ms-transform: matrix(1, -0.3, 0, 1, 0, 0); /\* IE 9 \*/

transform: matrix(1, -0.3, 0, 1, 0, 0); /\* 标准语法 \*/

}

div#myDiv2 {

-ms-transform: matrix(1, 0, 0.5, 1, 150, 0); /\* IE 9 \*/

transform: matrix(1, 0, 0.5, 1, 150, 0); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>matrix() 方法</h1>

<p>matrix() 方法将所有 2D 转换方法组合为一个。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv1">

使用 matrix() 方法。

</div>

<div id="myDiv2">

matrix() 方法的另一种用法。

</div>

</body>

</html>

## 82.11 CSS 转换属性

下表列出了所有 2D 变换属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| transform | 向元素应用 2D 或 3D 转换。 |
| transform-origin | 允许你改变被转换元素的位置。 |

## 82.12 CSS 2D 转换方法

|  |  |
| --- | --- |
| 函数 | 描述 |
| matrix(n,n,n,n,n,n) | 定义 2D 转换，使用六个值的矩阵。 |
| translate(x,y) | 定义 2D 转换，沿着 X 和 Y 轴移动元素。 |
| translateX(n) | 定义 2D 转换，沿着 X 轴移动元素。 |
| translateY(n) | 定义 2D 转换，沿着 Y 轴移动元素。 |
| scale(x,y) | 定义 2D 缩放转换，改变元素的宽度和高度。 |
| scaleX(n) | 定义 2D 缩放转换，改变元素的宽度。 |
| scaleY(n) | 定义 2D 缩放转换，改变元素的高度。 |
| rotate(angle) | 定义 2D 旋转，在参数中规定角度。 |
| skew(x-angle,y-angle) | 定义 2D 倾斜转换，沿着 X 和 Y 轴。 |
| skewX(angle) | 定义 2D 倾斜转换，沿着 X 轴。 |
| skewY(angle) | 定义 2D 倾斜转换，沿着 Y 轴。 |

# 第83章 CSS 3D 转换

## 83.1 CSS 3D 转换

CSS 还支持 3D 转换。

请将鼠标悬停在下面的元素上，即可查看 2D 和 3D 转换之间的区别：

在本章中，您将学习如下 CSS 属性：

transform

## 83.2 CSS 3D 转换方法

通过 CSS transform 属性，您可以使用以下 3D 转换方法：

rotateX()

rotateY()

rotateZ()

## 83.3 rotateX() 方法

rotateX() 方法使元素绕其 X 轴旋转给定角度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

#myDiv {

transform: rotateX(150deg);

}

</style>

</head>

<body>

<h1>rotateX() 方法</h1>

<p>rotationX() 方法将元素围绕其 X 轴旋转给定程度。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

该 div 元素旋转了 150 度。

</div>

<p><b>注释：</b>Internet Explorer 9（以及更早的版本）不支持 rotateX() 方法。</p>

</body>

</html>

## 83.4 rotateY() 方法

rotateY() 方法使元素绕其 Y 轴旋转给定角度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

#myDiv {

transform: rotateY(150deg);

}

</style>

</head>

<body>

<h1>rotateY() 方法</h1>

<p>rotateY() 方法将元素围绕其 Y 轴旋转给定程度。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

该 div 元素旋转了 150 度。

</div>

<p><b>注释：</b>Internet Explorer 9（以及更早的版本）不支持 rotateY() 方法。</p>

</body>

</html>

## 83.5 rotateZ() 方法

rotateZ() 方法使元素绕其 Z 轴旋转给定角度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

#myDiv {

transform: rotateZ(90deg);

}

</style>

</head>

<body>

<h1>rotateZ() 方法</h1>

<p>rotateZ() 方法将元素围绕其 Z 轴旋转给定程度。</p>

<div>

这是一个普通的 div 元素。

</div>

<div id="myDiv">

该 div 元素旋转了 90 度。

</div>

<p><b>注释：</b>Internet Explorer 9（以及更早的版本）不支持 rotateZ() 方法。</p>

</body>

</html>

## 83.6 CSS 转换属性

下表列出了所有 3D 变换属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| transform | 向元素应用 2D 或 3D 转换。 |
| transform-origin | 允许你改变被转换元素的位置。 |
| transform-style | 规定被嵌套元素如何在 3D 空间中显示。 |
| perspective | 规定 3D 元素的透视效果。 |
| perspective-origin | 规定 3D 元素的底部位置。 |
| backface-visibility | 定义元素在不面对屏幕时是否可见。 |

## 83.7 CSS 3D 转换方法

|  |  |
| --- | --- |
| 函数 | 描述 |
| matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n) | 定义 3D 转换，使用 16 个值的 4x4 矩阵。 |
| translate3d(x,y,z) | 定义 3D 转化。 |
| translateX(x) | 定义 3D 转化，仅使用用于 X 轴的值。 |
| translateY(y) | 定义 3D 转化，仅使用用于 Y 轴的值。 |
| translateZ(z) | 定义 3D 转化，仅使用用于 Z 轴的值。 |
| scale3d(x,y,z) | 定义 3D 缩放转换。 |
| scaleX(x) | 定义 3D 缩放转换，通过给定一个 X 轴的值。 |
| scaleY(y) | 定义 3D 缩放转换，通过给定一个 Y 轴的值。 |
| scaleZ(z) | 定义 3D 缩放转换，通过给定一个 Z 轴的值。 |
| rotate3d(x,y,z,angle) | 定义 3D 旋转。 |
| rotateX(angle) | 定义沿 X 轴的 3D 旋转。 |
| rotateY(angle) | 定义沿 Y 轴的 3D 旋转。 |
| rotateZ(angle) | 定义沿 Z 轴的 3D 旋转。 |
| perspective(n) | 定义 3D 转换元素的透视视图。 |

# 第84章 CSS 过渡

CSS 过渡允许您在给定的时间内平滑地改变属性值。

请把鼠标移动到这个元素上，来查看 CSS 过渡效果：

在本章中，您将学习如下属性：

transition

transition-delay

transition-duration

transition-property

transition-timing-function

## 84.1 如何使用 CSS 过渡？

如需创建过渡效果，必须明确两件事：

您要添加效果的 CSS 属性

效果的持续时间

注意：如果未规定持续时间部分，则过渡不会有效果，因为默认值为 0。

下面的例子展示了 100px \* 100px 的红色 <div> 元素。 <div> 元素还为 width 属性指定了过渡效果，持续时间为 2 秒：

实例

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s;

}

当指定的 CSS 属性（width）值发生变化时，将开始过渡效果。

现在，让我们为 width 属性指定一个鼠标悬停在 <div> 元素上时的新值：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s;

}

div:hover {

width: 300px;

}

</style>

</head>

<body>

<h1>transition 属性</h1>

<p>请把鼠标悬停在下面的 div 元素上，来查看过渡效果：</p>

<div></div>

<p><b>注释：</b>本例在 Internet Explorer 9 和更早版本中不起作用。</p>

</body>

</html>

## 84.2 改变若干属性值

下面的例子为 width 和 height 属性都添加了过渡效果，width 是 2 秒，height 是 4 秒：

实例

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s, height 4s;

}

div:hover {

width: 300px;

height: 300px;

}

</style>

</head>

<body>

<h1>transition 属性</h1>

<p>请把鼠标悬停在下面的 div 元素上，来查看过渡效果：</p>

<div></div>

<p><b>注释：</b>本例在 Internet Explorer 9 和更早版本中不起作用。</p>

</body>

</html>

## 84.3 指定过渡的速度曲线

transition-timing-function 属性规定过渡效果的速度曲线。

transition-timing-function 属性可接受以下值：

ease - 规定过渡效果，先缓慢地开始，然后加速，然后缓慢地结束（默认）

linear - 规定从开始到结束具有相同速度的过渡效果

ease-in -规定缓慢开始的过渡效果

ease-out - 规定缓慢结束的过渡效果

ease-in-out - 规定开始和结束较慢的过渡效果

cubic-bezier(n,n,n,n) - 允许您在三次贝塞尔函数中定义自己的值

下面的例子展示了可以使用的一些不同的速度曲线：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s;

}

#div1 {transition-timing-function: linear;}

#div2 {transition-timing-function: ease;}

#div3 {transition-timing-function: ease-in;}

#div4 {transition-timing-function: ease-out;}

#div5 {transition-timing-function: ease-in-out;}

div:hover {

width: 300px;

}

</style>

</head>

<body>

<h1>transition-timing-function 属性</h1>

<p>请把鼠标悬停在下面的 div 元素上，来查看不同的速度曲线：</p>

<div id="div1">linear</div><br>

<div id="div2">ease</div><br>

<div id="div3">ease-in</div><br>

<div id="div4">ease-out</div><br>

<div id="div5">ease-in-out</div><br>

<p><b>注释：</b>本例在 Internet Explorer 9 和更早版本中不起作用。</p>

</body>

</html>

## 84.4 延迟过渡效果

transition-delay 属性规定过渡效果的延迟（以秒计）。

下例在启动之前有 1 秒的延迟：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 3s;

transition-delay: 1s;

}

div:hover {

width: 300px;

}

</style>

</head>

<body>

<h1>transition-delay 属性</h1>

<p>请把鼠标悬停在下面的 div 元素上，来查看过渡效果：</p>

<div></div>

<p><b>注释：</b>过渡效果在开始之前有 1 秒的延迟。</p>

<p><b>注释：</b>本例在 Internet Explorer 9 和更早版本中不起作用。</p>

</body>

</html>

## 84.5 过渡 + 转换

下例为转换添加过渡效果：

实例

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s, height 2s, transform 2s;

}

div:hover {

width: 300px;

height: 300px;

transform: rotate(180deg);

}

</style>

</head>

<body>

<h1>Transition + Transform</h1>

<p>请把鼠标悬停在下面的 div 元素上：</p>

<div></div>

<p><b>注释：</b>本例在 Internet Explorer 9 和更早版本中不起作用。</p>

</body>

</html>

## 84.6 更多过渡实例

您可以可以一一指定 CSS 过渡属性，如下所示：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition-property: width;

transition-duration: 2s;

transition-timing-function: linear;

transition-delay: 1s;

}

div:hover {

width: 300px;

}

</style>

</head>

<body>

<h1>逐一指定过渡属性</h1>

<p>请把鼠标悬停在下面的 div 元素上，来查看过渡效果：</p>

<div></div>

<p><b>注释：</b>过渡效果在开始之前有 1 秒的延迟。</p>

<p><b>注释：</b>本例在 Internet Explorer 9 和更早版本中不起作用。</p>

</body>

</html>

或使用简写的 transition 属性：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s linear 1s;

}

div:hover {

width: 300px;

}

</style>

</head>

<body>

<h1>使用 transition 简写属性</h1>

<p>请把鼠标悬停在下面的 div 元素上，来查看过渡效果：</p>

<div></div>

<p><b>注释：</b>过渡效果在开始之前有 1 秒的延迟。</p>

<p><b>注释：</b>本例在 Internet Explorer 9 和更早版本中不起作用。</p>

</body>

</html>

## 84.7 CSS 过渡属性

下表列出了所有 CSS 过渡属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| transition | 简写属性，用于将四个过渡属性设置为单一属性。 |
| transition-delay | 规定过渡效果的延迟（以秒计）。 |
| transition-duration | 规定过渡效果要持续多少秒或毫秒。 |
| transition-property | 规定过渡效果所针对的 CSS 属性的名称。 |
| transition-timing-function | 规定过渡效果的速度曲线。 |

# 第85章 CSS 动画

CSS 可实现 HTML 元素的动画效果，而不使用 JavaScript 或 Flash！

在本章中，您将学习如下属性：

@keyframes

animation-name

animation-duration

animation-delay

animation-iteration-count

animation-direction

animation-timing-function

animation-fill-mode

Animation

## 85.1 什么是 CSS 动画？

动画使元素逐渐从一种样式变为另一种样式。

您可以随意更改任意数量的 CSS 属性。

如需使用 CSS 动画，您必须首先为动画指定一些关键帧。

关键帧包含元素在特定时间所拥有的样式。

## 85.2 @keyframes 规则

如果您在 @keyframes 规则中指定了 CSS 样式，动画将在特定时间逐渐从当前样式更改为新样式。

要使动画生效，必须将动画绑定到某个元素。

下面的例子将 "example" 动画绑定到 <div> 元素。动画将持续 4 秒钟，同时将 <div> 元素的背景颜色从 "red" 逐渐改为 "yellow"：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

from {background-color: red;}

to {background-color: yellow;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

<p><b>注释：</b>当动画结束后，会变回最初的样式。</p>

</body>

</html>

注意：animation-duration 属性定义需要多长时间才能完成动画。如果未指定 animation-duration 属性，则动画不会发生，因为默认值是 0s（0秒）。

在上面的例子中，通过使用关键字 "from" 和 "to"（代表 0％（开始）和 100％（完成）），我们设置了样式何时改变。

您也可以使用百分比值。通过使用百分比，您可以根据需要添加任意多个样式更改。

下面的例子将在动画完成 25％，完成 50％ 以及动画完成 100％ 时更改 <div> 元素的背景颜色：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

0% {background-color: red;}

25% {background-color: yellow;}

50% {background-color: blue;}

100% {background-color: green;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

下面的例子将在动画完成 25％，完成 50％ 以及动画完成 100％ 时更改背景颜色和 <div> 元素的位置：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

## 85.3 延迟动画

animation-delay 属性规定动画开始的延迟时间。

下面的例子在开始动画前有 2 秒的延迟：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-delay: 2s;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

负值也是允许的。如果使用负值，则动画将开始播放，如同已播放 N 秒。

在下面的例子中，动画将开始播放，就好像它已经播放了 2 秒钟一样：

实例

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-delay: -2s;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p>使用负值：在这里，动画将开始播放，就好像它已经播放 2 秒一样：</p>

<div></div>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

</body>

</html>

## 85.4 设置动画应运行多少次

animation-iteration-count 属性指定动画应运行的次数。

下面的例子在停止前把动画运行 3 次：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-iteration-count: 3;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

下面的例子使用值 "infinite" 使动画永远持续下去：

实例

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-iteration-count: infinite;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

## 85.5 反向或交替运行动画

animation-direction 属性指定是向前播放、向后播放还是交替播放动画。

animation-direction 属性可接受以下值：

normal - 动画正常播放（向前）。默认值

reverse - 动画以反方向播放（向后）

alternate - 动画先向前播放，然后向后

alternate-reverse - 动画先向后播放，然后向前

下例将以相反的方向（向后）运行动画：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-direction: reverse;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

下面的例子使用值 "alternate" 使动画先向前运行，然后向后运行：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-iteration-count: 2;

animation-direction: alternate;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

下面的例子使用值 "alternate-reverse" 使动画先向后运行，然后向前运行：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-iteration-count: 2;

animation-direction: alternate-reverse;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

## 85.6 指定动画的速度曲线

animation-timing-function 属性规定动画的速度曲线。

animation-timing-function 属性可接受以下值：

ease - 指定从慢速开始，然后加快，然后缓慢结束的动画（默认）

linear - 规定从开始到结束的速度相同的动画

ease-in - 规定慢速开始的动画

ease-out - 规定慢速结束的动画

ease-in-out - 指定开始和结束较慢的动画

cubic-bezier(n,n,n,n) - 运行您在三次贝塞尔函数中定义自己的值

下面这些例子展示了可以使用的一些不同速度曲线：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 50px;

background-color: red;

font-weight: bold;

position: relative;

animation: mymove 5s infinite;

}

#div1 {animation-timing-function: linear;}

#div2 {animation-timing-function: ease;}

#div3 {animation-timing-function: ease-in;}

#div4 {animation-timing-function: ease-out;}

#div5 {animation-timing-function: ease-in-out;}

@keyframes mymove {

from {left: 0px;}

to {left: 300px;}

}

</style>

</head>

<body>

<p><b>注释：</b>Internet Explorer 9 以及更早的版本不支持 animation-timing-funtion 属性。</p>

<div id="div1">linear</div>

<div id="div2">ease</div>

<div id="div3">ease-in</div>

<div id="div4">ease-out</div>

<div id="div5">ease-in-out</div>

</body>

</html>

## 85.7 指定动画的填充模式

CSS 动画不会在第一个关键帧播放之前或在最后一个关键帧播放之后影响元素。animation-fill-mode 属性能够覆盖这种行为。

在不播放动画时（在开始之前，结束之后，或两者都结束时），animation-fill-mode 属性规定目标元素的样式。

animation-fill-mode 属性可接受以下值：

none - 默认值。动画在执行之前或之后不会对元素应用任何样式。

forwards - 元素将保留由最后一个关键帧设置的样式值（依赖 animation-direction 和 animation-iteration-count）。

backwards - 元素将获取由第一个关键帧设置的样式值（取决于 animation-direction），并在动画延迟期间保留该值。

both - 动画会同时遵循向前和向后的规则，从而在两个方向上扩展动画属性。

下面的例子让 <div> 元素在动画结束时保留来自最后一个关键帧的样式值：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

position: relative;

animation-name: example;

animation-duration: 3s;

animation-fill-mode: forwards;

}

@keyframes example {

from {top: 0px;}

to {top: 200px; background-color: blue;}

}

</style>

</head>

<body>

<p>动画结束后，让 div 元素保留最后一个关键帧的样式值：</p>

<div></div>

<p><b>注释：</b>Internet Explorer 9 以及更早的版本不支持 animation-fill-mode 属性。</p>

</body>

</html>

下面的例子在动画开始之前（在动画延迟期间）使 <div> 元素获得由第一个关键帧设置的样式值：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

position: relative;

animation-name: example;

animation-duration: 3s;

animation-delay: 2s;

animation-fill-mode: backwards;

}

@keyframes example {

from {top: 0px; background-color: yellow;}

to {top: 200px;}

}

</style>

</head>

<body>

<p>动画开始之前（animation-delay 期间），让 div 元素获取第一个关键帧设置的样式值：</p>

<div></div>

<p><b>注释：</b>Internet Explorer 9 以及更早的版本不支持 animation-fill-mode 属性。</p>

</body>

</html>

下面的例子在动画开始之前使 <div> 元素获得第一个关键帧设置的样式值，以及在动画结束时保留最后一个关键帧的样式值：

实例

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

position: relative;

animation-name: example;

animation-duration: 3s;

animation-delay: 2s;

animation-fill-mode: both;

}

@keyframes example {

from {top: 0px; background-color: yellow;}

to {top: 200px; background-color: blue;}

}

</style>

</head>

<body>

<p>在动画开始之前，让 div 元素获取第一个关键帧设置的样式值，并在动画结束时保留最后一个关键帧的样式值：</p>

<div></div>

<p><b>注释：</b>Internet Explorer 9 以及更早的版本不支持 animation-fill-mode 属性。</p>

</body>

</html>

## 85.8 动画简写属性

下例使用六种动画属性：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 5s;

animation-timing-function: linear;

animation-delay: 2s;

animation-iteration-count: infinite;

animation-direction: alternate;

}

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

使用简写的 animation 属性也可以实现与上例相同的动画效果：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation: myfirst 5s linear 2s infinite alternate;

}

@keyframes myfirst {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 9 以及更早的版本中无效。</p>

<div></div>

</body>

</html>

## 85.9 CSS 动画属性

下表列出了 @keyframes 规则和所有 CSS 动画属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| @keyframes | 规定动画模式。 |
| animation | 设置所有动画属性的简写属性。 |
| animation-delay | 规定动画开始的延迟。 |
| animation-direction | 定动画是向前播放、向后播放还是交替播放。 |
| animation-duration | 规定动画完成一个周期应花费的时间。 |
| animation-fill-mode | 规定元素在不播放动画时的样式（在开始前、结束后，或两者同时）。 |
| animation-iteration-count | 规定动画应播放的次数。 |
| animation-name | 规定 @keyframes 动画的名称。 |
| animation-play-state | 规定动画是运行还是暂停。 |
| animation-timing-function | 规定动画的速度曲线。 |

# 第86章 CSS 工具提示

通过 CSS 创建工具提示（Tooltip）。

## 86.1 基础的工具提示

创建一个鼠标移到元素上时显示的工具提示：

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

/\* 定位工具提示 \*/

position: absolute;

z-index: 1;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<p>请把鼠标移到下面的文字上：</p>

<div class="tooltip">移到我上面

<span class="tooltiptext">工具提示文本</span>

</div>

<p>请注意，工具提示文本的位置不是很好。请返回教程并继续阅读如何以理想的方式放置工具提示。</p>

</body>

</html>

例子解释

HTML：

使用容器元素（例如 <div>）并向其添加 "tooltip" 类。当用户将鼠标悬停在此 <div> 上时，会显示工具提示文本。

工具提示文本位于 class="tooltiptext" 的嵌入式元素（如 <span>）中。

CSS：

tooltip 类使用 position:relative，用于放置工具提示文本（position:absolute）。注意：有关如何放置工具提示，请参见下面的例子。

tooltiptext 类保存实际的工具提示文本。默认情况下它是隐藏的，并会在鼠标悬停时可见（请参阅下文）。我们还为其添加了一些基本样式：120 像素的宽度、黑色背景、白色文本、文本居中以及 5px 的上下内边距（padding）。

CSS border-radius 属性用于向工具提示文本添加圆角。

当用户将鼠标移到 class="tooltip" 的 <div> 上时，:hover 选择器用于显示工具提示文本。

## 86.2 定位工具提示

在本例中，工具提示位于“可悬停”文本（<div>）的右侧（left:105%）。另外请注意，top:-5px 用于将其放置在其容器元素的中间。我们使用数字 5 是因为工具提示文本的上下内边距均为 5px。如果增加其内边距，还请您同时增加 top 属性的值，以确保它停留在中间。如果要将工具提示放在左侧，也同样适用。

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

/\* 定位工具提示 \*/

position: absolute;

z-index: 1;

top: -5px;

left: 105%;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h1>Right Tooltip</h1>

<p>请把鼠标移到下方文本上：</p>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

/\* 定位工具提示 \*/

position: absolute;

z-index: 1;

top: -5px;

right: 105%;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h1>左侧工具提示</h1>

<p>请把鼠标移到下方文本上：</p>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

如果您希望工具提示位于上方或下方，请看下面的例子。请注意，我们使用了负 60 像素的左外边距属性（margin-left）。这是为了把工具提示与可悬停文本进行居中对齐。该值是工具提示宽度的一半（120/2 = 60）。

**顶部工具提示**

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

/\* 定位工具提示 \*/

position: absolute;

z-index: 1;

bottom: 100%;

left: 50%;

margin-left: -60px;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h1>Top Tooltip</h1>

<p>请把鼠标移到下方文本上：</p>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

**底部工具提示**

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

/\* 定位工具提示 \*/

position: absolute;

z-index: 1;

top: 100%;

left: 50%;

margin-left: -60px;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h1>下方工具提示</h1>

<p>请把鼠标移到下方文本上：</p>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

## 86.3 工具提示箭头

如需创建在工具提示的指定侧面显示的箭头，请在工具提示后添加“空的”内容，并使用伪元素类 ::after 和 content 属性。箭头本身是使用边框创建的。这会使工具提示看起来像气泡。

本例演示如何在工具提示的底部添加箭头：

**底部箭头**

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

position: absolute;

z-index: 1;

bottom: 150%;

left: 50%;

margin-left: -60px;

}

.tooltip .tooltiptext::after {

content: "";

position: absolute;

top: 100%;

left: 50%;

margin-left: -5px;

border-width: 5px;

border-style: solid;

border-color: black transparent transparent transparent;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h2>Top Tooltip w/ Bottom Arrow</h2>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

例子解释

将箭头定位在工具提示内：top: 100% 将箭头放置在工具提示的底部。left: 50% 将使箭头居中。

注意：border-width 属性指定箭头的大小。如果您更改此设置，也请将 margin-left 值更改为相同值。这将使箭头居中。

border-color 用于将内容转换为箭头。我们将上边框设置为黑色，其余设置为透明。如果所有面都是黑色，则最终将得到一个黑色的方形框。

本例演示了如何在工具提示的顶部添加箭头。请注意，这次我们设置了下边框的颜色：

**顶部箭头**

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

position: absolute;

z-index: 1;

top: 150%;

left: 50%;

margin-left: -60px;

}

.tooltip .tooltiptext::after {

content: "";

position: absolute;

bottom: 100%;

left: 50%;

margin-left: -5px;

border-width: 5px;

border-style: solid;

border-color: transparent transparent black transparent;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h1>下方工具提示 w/ 上箭头</h1>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

本例演示如何在工具提示的左侧添加箭头：

**左侧箭头**

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

position: absolute;

z-index: 1;

top: -5px;

left: 110%;

}

.tooltip .tooltiptext::after {

content: "";

position: absolute;

top: 50%;

right: 100%;

margin-top: -5px;

border-width: 5px;

border-style: solid;

border-color: transparent black transparent transparent;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h1>右侧工具提示 w/ 左箭头</h1>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

本例演示如何在工具提示的右侧添加箭头：

**右侧箭头**

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

position: absolute;

z-index: 1;

top: -5px;

right: 110%;

}

.tooltip .tooltiptext::after {

content: "";

position: absolute;

top: 50%;

left: 100%;

margin-top: -5px;

border-width: 5px;

border-style: solid;

border-color: transparent transparent transparent black;

}

.tooltip:hover .tooltiptext {

visibility: visible;

}

</style>

<body style="text-align:center;">

<h1>左侧工具提示 w/ 右箭头</h1>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

## 86.4 淡入的工具提示（动画）

如果希望在即将显示的工具提示文本中淡入淡出，可以将 CSS transition 属性与 opacity 属性一同使用，并在指定的秒数（例子中是 1 秒）内从完全不可见变为 100％ 可见：

<!DOCTYPE html>

<html>

<style>

.tooltip {

position: relative;

display: inline-block;

border-bottom: 1px dotted black;

}

.tooltip .tooltiptext {

visibility: hidden;

width: 120px;

background-color: black;

color: #fff;

text-align: center;

border-radius: 6px;

padding: 5px 0;

position: absolute;

z-index: 1;

bottom: 100%;

left: 50%;

margin-left: -60px;

/\* 淡入工具提示 - 用 1 秒从完全不可见变为可见： \*/

opacity: 0;

transition: opacity 1s;

}

.tooltip:hover .tooltiptext {

visibility: visible;

opacity: 1;

}

</style>

<body style="text-align:center;">

<h1>悬停时淡入的工具提示</h1>

<p>当您将鼠标移到下方的文本上时，工具提示文本将淡入并花费 1 秒的时间从完全不可见变为可见。</p>

<div class="tooltip">Hover over me

<span class="tooltiptext">Tooltip text</span>

</div>

</body>

</html>

# 第87章 CSS 图像样式

## 87.1 圆角图像

使用 border-radius 属性创建圆形图像：

<!DOCTYPE html>

<html>

<head>

<style>

img {

border-radius: 8px;

}

</style>

</head>

<body>

<h1>圆角图像</h1>

<p>请使用 border-radius 属性来创建圆角图像：</p>

<img src="/i/photo/tulip.jpg" alt="Tulip" width="300" height="300">

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

img {

border-radius: 50%;

}

</style>

</head>

<body>

<h1>圆角图像</h1>

<p>请使用 border-radius 属性创建圆形的图像：</p>

<img src="/i/photo/tulip.jpg" alt="Tulip" width="300" height="300">

</body>

</html>

## 87.2 缩略图图像

使用 border 属性创建缩略图。

<!DOCTYPE html>

<html>

<head>

<style>

img {

border: 1px solid #ddd;

border-radius: 4px;

padding: 5px;

width: 150px;

}

</style>

</head>

<body>

<h1>缩略图</h1>

<p>请使用 border 属性来创建缩略图：</p>

<img src="/i/photo/tulip.jpg" alt="Tulip" style="width:150px">

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

img {

border: 1px solid #ddd;

border-radius: 4px;

padding: 5px;

width: 150px;

}

img:hover {

box-shadow: 0 0 2px 1px rgba(0, 140, 186, 0.5);

}

</style>

</head>

<body>

<h1>作链接的缩略图</h1>

<p>请使用 border 属性创建缩略图。用锚包围图像以将其用作链接。</p>

<p>请把鼠标悬停在图像上，然后单击以查看效果。</p>

<a target="\_blank" href="paris.jpg">

<img src="/i/photo/tulip.jpg" alt="Tulip" style="width:150px">

</a>

</body>

</html>

## 87.3 响应式图像

响应式图像会自动调整以适合屏幕尺寸。

如果您希望根据需要缩小图像，但需要杜绝放大到大于原始尺寸，请添加如下代码：

<!DOCTYPE html>

<html>

<head>

<style>

img {

max-width: 100%;

height: auto;

}

</style>

</head>

<body>

<h1>响应式图像</h1>

<p>自适应图像将自动调整以适合屏幕尺寸。</p>

<p>请调整浏览器窗口的大小以查看效果：</p>

<img src="/i/logo/w3logo-2.png" alt="W3School" width="800" height="450">

</body>

</html>

## 87.4 居中图像

如需使图像居中，请将左右外边距设置为 auto 并将其设置为块元素：

<!DOCTYPE html>

<html>

<head>

<style>

img {

display: block;

margin-left: auto;

margin-right: auto;

}

</style>

</head>

<body>

<h1>居中图像</h1>

<p>如需对图像进行居中，请把左右外边距设置为 auto，并转为块元素。</p>

<img src="/i/photo/tulip.jpg" alt="Tulip" style="width:50%">

</body>

</html>

## 87.5 宝丽来图片 / 卡片

<!DOCTYPE html>

<html>

<head>

<style>

body {margin:25px;}

div.polaroid {

width: 80%;

background-color: white;

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);

margin-bottom: 25px;

}

div.container {

text-align: center;

padding: 10px 20px;

}

</style>

</head>

<body>

<h1>响应式宝丽来图像 / 卡片</h1>

<div class="polaroid">

<img src="/i/photo/tulip-yellow.jpg" alt="Tulip" style="width:100%">

<div class="container">

<p>黄色郁金香</p>

</div>

</div>

<div class="polaroid">

<img src="/i/photo/tulip-red.jpg" alt="Tulip" style="width:100%">

<div class="container">

<p>红色郁金香</p>

</div>

</div>

</body>

</html>

## 87.6 透明图像

opacity 属性的取值范围为 0.0 - 1.0。值越低，越透明：

<!DOCTYPE html>

<html>

<head>

<style>

img {

opacity: 0.5;

}

</style>

</head>

<body>

<h1>图像透明度</h1>

<p>opacity 属性规定元素的透明度。值越低，越透明：</p>

<p>50% 不透明度的图像：</p>

<img src="/i/photo/tulip-red.jpg" alt="Tulip" width="600" height="400">

</body>

</html>

## 87.7 图像文本

如何在图像中定位文本：

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

}

.topleft {

position: absolute;

top: 8px;

left: 16px;

font-size: 18px;

}

img {

width: 100%;

height: auto;

opacity: 0.3;

}

</style>

</head>

<body>

<h1>图像文本</h1>

<p>在图像左上角添加一些文本：</p>

<div class="container">

<img src="/i/logo/w3logo-2.png" alt="W3School" width="800" height="450">

<div class="topleft">Top Left</div>

</div>

</body>

</html>

## 87.8 图像滤镜

CSS filter 属性把视觉效果（如模糊和饱和度）添加到元素。

注意：Internet Explorer 或 Edge 12 不支持 filter 属性。

实例

把所有图像的颜色更改为黑白（100％ 灰色）：

img {

filter: grayscale(100%);

}

<!DOCTYPE html>

<html>

<head>

<style>

img {

width: 33%;

height: auto;

float: left;

max-width: 235px;

}

.blur {filter: blur(4px);}

.brightness {filter: brightness(250%);}

.contrast {filter: contrast(180%);}

.grayscale {filter: grayscale(100%);}

.huerotate {filter: hue-rotate(180deg);}

.invert {filter: invert(100%);}

.opacity {filter: opacity(50%);}

.saturate {filter: saturate(7);}

.sepia {filter: sepia(100%);}

.shadow {filter: drop-shadow(8px 8px 10px green);}

</style>

</head>

<body>

<p><b>注释：</b>Internet Explorer 或 Edge 12 不支持 filter 属性。</p>

<img src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="blur" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="brightness" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="contrast" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="grayscale" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="huerotate" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="invert" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="opacity" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="saturate" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="sepia" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

<img class="shadow" src="/i/photo/tulip.jpg" alt="Pineapple" width="300" height="300">

</body>

</html>

## 87.9 图像悬停叠加

创建鼠标悬停时的叠加效果：

**淡入文本：**

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

width: 50%;

}

.image {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

top: 0;

bottom: 0;

left: 0;

right: 0;

height: 100%;

width: 100%;

opacity: 0;

transition: .5s ease;

background-color: #008CBA;

}

.container:hover .overlay {

opacity: 1;

}

.text {

color: white;

font-size: 20px;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<h1>淡入文本</h1>

<div class="container">

<img src="/i/css/avatar.png" alt="Avatar" class="image">

<div class="overlay">

<div class="text">Hello World</div>

</div>

</div>

</body>

</html>

**淡入框：**

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

width: 50%;

}

.image {

opacity: 1;

display: block;

width: 100%;

height: auto;

transition: .5s ease;

backface-visibility: hidden;

}

.middle {

transition: .5s ease;

opacity: 0;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%)

}

.container:hover .image {

opacity: 0.3;

}

.container:hover .middle {

opacity: 1;

}

.text {

background-color: #4CAF50;

color: white;

font-size: 16px;

padding: 16px 32px;

}

</style>

</head>

<body>

<h1>淡入框</h1>

<div class="container">

<img src="/i/css/avatar.png" alt="Avatar" class="image" style="width:100%">

<div class="middle">

<div class="text">Bill Gates</div>

</div>

</div>

</body>

</html>

**滑入（上）：**

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

width: 50%;

}

.image {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

bottom: 100%;

left: 0;

right: 0;

background-color: #008CBA;

overflow: hidden;

width: 100%;

height: 0;

transition: .5s ease;

}

.container:hover .overlay {

bottom: 0;

height: 100%;

}

.text {

white-space: nowrap;

color: white;

font-size: 20px;

position: absolute;

overflow: hidden;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<h1>滑入（从上）</h1>

<div class="container">

<img src="/i/css/avatar.png" alt="Avatar" class="image">

<div class="overlay">

<div class="text">Hello World</div>

</div>

</div>

</body>

</html>

**滑入（下）：**

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

width: 50%;

}

.image {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

bottom: 0;

left: 0;

right: 0;

background-color: #008CBA;

overflow: hidden;

width: 100%;

height: 0;

transition: .5s ease;

}

.container:hover .overlay {

height: 100%;

}

.text {

white-space: nowrap;

color: white;

font-size: 20px;

position: absolute;

overflow: hidden;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<h1>滑入（从下）</h1>

<div class="container">

<img src="/i/css/avatar.png" alt="Avatar" class="image">

<div class="overlay">

<div class="text">Hello World</div>

</div>

</div>

</body>

</html>

**滑入（左）：**

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

width: 50%;

}

.image {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

bottom: 0;

left: 0;

right: 0;

background-color: #008CBA;

overflow: hidden;

width: 0;

height: 100%;

transition: .5s ease;

}

.container:hover .overlay {

width: 100%;

}

.text {

white-space: nowrap;

color: white;

font-size: 20px;

position: absolute;

overflow: hidden;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<h1>滑入（从左）</h1>

<div class="container">

<img src="/i/css/avatar.png" alt="Avatar" class="image">

<div class="overlay">

<div class="text">Hello World</div>

</div>

</div>

</body>

</html>

**滑入（右）：**

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

width: 50%;

}

.image {

display: block;

width: 100%;

height: auto;

}

.overlay {

position: absolute;

bottom: 0;

left: 100%;

right: 0;

background-color: #008CBA;

overflow: hidden;

width: 0;

height: 100%;

transition: .5s ease;

}

.container:hover .overlay {

width: 100%;

left: 0;

}

.text {

white-space: nowrap;

color: white;

font-size: 20px;

position: absolute;

overflow: hidden;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<h1>滑入（从右）</h1>

<div class="container">

<img src="/i/css/avatar.png" alt="Avatar" class="image">

<div class="overlay">

<div class="text">Hello World</div>

</div>

</div>

</body>

</html>

## 87.10 翻转图像

实例

img:hover {

transform: scaleX(-1);

}

## 87.11 响应式图库

我们可以使用 CSS 创建自适应的图片库。

本例使用媒体查询来重新排列不同屏幕尺寸的图像。请调整浏览器窗口的大小以查看效果：

<!DOCTYPE html>

<html>

<head>

<style>

div.gallery {

border: 1px solid #ccc;

}

div.gallery:hover {

border: 1px solid #777;

}

div.gallery img {

width: 100%;

height: auto;

}

div.desc {

padding: 15px;

text-align: center;

}

\* {

box-sizing: border-box;

}

.responsive {

padding: 0 6px;

float: left;

width: 24.99999%;

}

@media only screen and (max-width: 700px) {

.responsive {

width: 49.99999%;

margin: 6px 0;

}

}

@media only screen and (max-width: 500px) {

.responsive {

width: 100%;

}

}

.clearfix:after {

content: "";

display: table;

clear: both;

}

</style>

</head>

<body>

<h1>响应式图片库</h1>

<h2>请调整窗口大小来查看效果。</h2>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/tulip-yellow.jpg">

<img src="/i/photo/tulip-yellow.jpg" alt="黄色郁金香" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/tulip-red.jpg">

<img src="/i/photo/tulip-red.jpg" alt="红色郁金香" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/flower-1.jpg">

<img src="/i/photo/flower-1.jpg" alt="花花草草" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="responsive">

<div class="gallery">

<a target="\_blank" href="/i/photo/flower-2.jpg">

<img src="/i/photo/flower-2.jpg" alt="花花草草" width="600" height="400">

</a>

<div class="desc">在此处添加图像描述</div>

</div>

</div>

<div class="clearfix"></div>

<div style="padding:6px;">

<p>本例使用媒体查询来重新排列不同屏幕尺寸的图像：对于宽于 700 像素的屏幕，它将并排显示四幅图像；对于小于 700 像素的屏幕，将并排显示两幅图像。对于小于 500 像素的屏幕，图像将垂直堆叠（100％）。</p>

<p>您稍后将在我们的 CSS 教程中学到有关媒体查询和响应式 Web 设计的更多知识。</p>

</div>

</body>

</html>

## 87.12 图像模态（Image Modal）

这是一个演示 CSS 和 JavaScript 如何协同工作的例子。

首先，请使用 CSS 创建模态窗口（对话框），并默认将其隐藏。

然后，当用户单击图像时，使用 JavaScript 显示模态窗口并在模态内部显示图像：

<!DOCTYPE html>

<html>

<head>

<style>

#myImg {

border-radius: 5px;

cursor: pointer;

transition: 0.3s;

}

#myImg:hover {opacity: 0.7;}

/\* The Modal (background) \*/

.modal {

display: none; /\* Hidden by default \*/

position: fixed; /\* Stay in place \*/

z-index: 1; /\* Sit on top \*/

padding-top: 100px; /\* Location of the box \*/

left: 0;

top: 0;

width: 100%; /\* Full width \*/

height: 100%; /\* Full height \*/

overflow: auto; /\* Enable scroll if needed \*/

background-color: rgb(0,0,0); /\* Fallback color \*/

background-color: rgba(0,0,0,0.9); /\* Black w/ opacity \*/

}

/\* Modal Content (image) \*/

.modal-content {

margin: auto;

display: block;

width: 80%;

max-width: 1500px;

}

/\* Caption of Modal Image \*/

#caption {

margin: auto;

display: block;

width: 80%;

max-width: 700px;

text-align: center;

color: #ccc;

padding: 10px 0;

height: 150px;

}

/\* Add Animation \*/

.modal-content, #caption {

animation-name: zoom;

animation-duration: 0.6s;

}

@keyframes zoom {

from {transform: scale(0.1)}

to {transform: scale(1)}

}

/\* The Close Button \*/

.close {

position: absolute;

top: 15px;

right: 35px;

color: #f1f1f1;

font-size: 40px;

font-weight: bold;

transition: 0.3s;

}

.close:hover,

.close:focus {

color: #bbb;

text-decoration: none;

cursor: pointer;

}

/\* 100% Image Width on Smaller Screens \*/

@media only screen and (max-width: 700px){

.modal-content {

width: 100%;

}

}

</style>

</head>

<body>

<h1>图像模态</h1>

<p>在此例中，我们使用 CSS 创建默认情况下隐藏的模式（对话框）。</p>

<p>我们使用 JavaScript 触发模态，并在单击模态时在模态内显示当前图像。还请注意，我们将图像的“alt”属性中的值用作模态内的图像标题文本。</p>

<p>如果您无法立即理解代码，请不要担心。学习完 CSS 后，请转到我们的 JavaScript 教程学习更多相关知识。</p>

<img id="myImg" src="/i/photo/tiyugongyuan.jpg" alt="绿茵场" style="width: 50%;">

<!-- The Modal -->

<div id="myModal" class="modal">

<span class="close">×</span>

<img class="modal-content" id="img01">

<div id="caption"></div>

</div>

<script>

// Get the modal

var modal = document.getElementById('myModal');

// Get the image and insert it inside the modal - use its "alt" text as a caption

var img = document.getElementById('myImg');

var modalImg = document.getElementById("img01");

var captionText = document.getElementById("caption");

img.onclick = function(){

modal.style.display = "block";

modalImg.src = this.src;

captionText.innerHTML = this.alt;

}

// Get the <span> element that closes the modal

var span = document.getElementsByClassName("close")[0];

// When the user clicks on <span> (x), close the modal

span.onclick = function() {

modal.style.display = "none";

}

</script>

</body>

</html>

# 第88章 CSS object-fit 属性

CSS object-fit 属性用于规定应如何调整 <img> 或 <video> 的大小来适应其容器。

## 88.1 CSS object-fit 属性

CSS object-fit 属性用于指定应如何调整 <img> 或 <video> 的大小以适合其容器。

这个属性告诉内容以不同的方式填充容器。比如“保留长宽比”或者“展开并占用尽可能多的空间”。

请看下面来自上海鲜花港的郁金香图片，它是 300x300 像素：

但是，如果我们把上面的图像设置为 200x300 像素，则它会看起来像这样：

<!DOCTYPE html>

<html>

<head>

<style>

img {

width:200px;

height:400px;

}

</style>

</head>

<body>

<h1>Image</h1>

<img src="/i/photo/tulip.jpg" alt="Tulip" width="400" height="300">

</body>

</html>

我们看到图像被压缩以适合 200x300 像素的容器，并且原始宽高比被破坏了。

如果我们使用 object-fit: cover;，它会剪切图像的侧面，保留长宽比，并填充空间，如下所示：

<!DOCTYPE html>

<html>

<head>

<style>

img {

width:200px;

height:400px;

object-fit:cover;

}

</style>

</head>

<body>

<h1>object-fit 属性</h1>

<img src="/i/photo/tulip.jpg" alt="Tulip" width="300" height="300">

</body>

</html>

## 88.2 另一个实例

在这里，我们有两幅图像，我们希望它们填充浏览器窗口的 50％ 的宽度和 100％ 的高度。

在下面的例子中，我们不使用 object-fit，因此，当我们调整浏览器窗口的大小时，图像的长宽比将被破坏：

实例

<!DOCTYPE html>

<html>

<body>

<h1>未使用 object-fit</h1>

<p>在这里，我们没有使用 "object-fit"，因此如果我们调整浏览器窗口的大小，会破坏图像的长宽比：</p>

<div style="width:100%;height:400px;">

<img src="/i/photo/tiyugongyuan.jpg" alt="Shanghai" style="float:left;width:50%;height:100%;">

<img src="/i/photo/tulip.jpg" alt="Tulip" style="float:left;width:50%;height:100%;">

</div>

</body>

</html>

在下一个例子中，我们使用 object-fit: cover;，因此，当我们调整浏览器窗口的大小时，将保留图像的长宽比：

<!DOCTYPE html>

<html>

<body>

<h1>使用 object-fit</h1>

<p>此处我们用了 "object-fit: cover;"，因此如果我们调整浏览器窗口大小，图像的高宽比会被保留：</p>

<div style="width:100%;height:400px;">

<img src="/i/photo/tiyugongyuan.jpg" alt="Shanghai" style="float:left;width:50%;height:100%;object-fit:cover;">

<img src="/i/photo/tulip.jpg" alt="Tulip" style="float:left;width:50%;height:100%;object-fit:cover;">

</div>

</body>

</html>

## 88.3 CSS object-fit 属性的所有值

object-fit 属性可接受如下值：

fill - 默认值。调整替换后的内容大小，以填充元素的内容框。如有必要，将拉伸或挤压物体以适应该对象。

contain - 缩放替换后的内容以保持其纵横比，同时将其放入元素的内容框。

cover - 调整替换内容的大小，以在填充元素的整个内容框时保持其长宽比。该对象将被裁剪以适应。

none - 不对替换的内容调整大小。

scale-down - 调整内容大小就像没有指定内容或包含内容一样（将导致较小的具体对象尺寸）

下面的例子演示了 object-fit 属性的所有可能值：

实例

<!DOCTYPE html>

<html>

<head>

<style>

.fill {object-fit: fill;}

.contain {object-fit: contain;}

.cover {object-fit: cover;}

.scale-down {object-fit: scale-down;}

.none {object-fit: none;}

</style>

</head>

<body>

<h1>object-fit 属性</h1>

<h2>No object-fit:</h2>

<img src="/i/photo/tulip.jpg" alt="Tulip" style="width:200px;height:400px">

<h2>object-fit: fill (this is default):</h2>

<img class="fill" src="/i/photo/tulip.jpg" alt="Tulip" style="width:200px;height:400px">

<h2>object-fit: contain:</h2>

<img class="contain" src="/i/photo/tulip.jpg" alt="Tulip" style="width:200px;height:400px">

<h2>object-fit: cover:</h2>

<img class="cover" src="/i/photo/tulip.jpg" alt="Tulip" style="width:200px;height:400px">

<h2>object-fit: scale-down:</h2>

<img class="scale-down" src="/i/photo/tulip.jpg" alt="Tulip" style="width:200px;height:400px">

<h2>object-fit: none:</h2>

<img class="none" src="/i/photo/tulip.jpg" alt="Tulip" style="width:200px;height:400px">

</body>

</html>

# 第89章 CSS 按钮

## 89.1 基本按钮样式

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50;

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

</style>

</head>

<body>

<h1>CSS 按钮</h1>

<button>默认按钮</button>

<a href="#" class="button">链接按钮</a>

<button class="button">按钮</button>

<input type="button" class="button" value="输入按钮">

</body>

</html>

## 89.2 按钮颜色

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* 绿色 \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.button2 {background-color: #008CBA;} /\* 蓝色 \*/

.button3 {background-color: #f44336;} /\* 黑色 \*/

.button4 {background-color: #e7e7e7; color: black;} /\* 灰色 \*/

.button5 {background-color: #555555;} /\* 黑色 \*/

</style>

</head>

<body>

<h1>按钮颜色</h1>

<p>通过 background-color 属性改变按钮的背景色：</p>

<button class="button">绿色</button>

<button class="button button2">蓝色</button>

<button class="button button3">红色</button>

<button class="button button4">灰色</button>

<button class="button button5">黑色</button>

</body>

</html>

## 89.3 按钮尺寸

请使用 font-size 属性更改按钮的字体大小

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* 绿色 \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

margin: 4px 2px;

cursor: pointer;

}

.button1 {font-size: 10px;}

.button2 {font-size: 12px;}

.button3 {font-size: 16px;}

.button4 {font-size: 20px;}

.button5 {font-size: 24px;}

</style>

</head>

<body>

<h1>按钮大小</h1>

<p>通过 font-size 属性改变按钮的字体大小：</p>

<button class="button button1">10px</button>

<button class="button button2">12px</button>

<button class="button button3">16px</button>

<button class="button button4">20px</button>

<button class="button button5">24px</button>

</body>

</html>

请使用 padding 属性更改按钮的内边距：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* 绿色 \*/

border: none;

color: white;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.button1 {padding: 10px 24px;}

.button2 {padding: 12px 28px;}

.button3 {padding: 14px 40px;}

.button4 {padding: 32px 16px;}

.button5 {padding: 16px;}

</style>

</head>

<body>

<h1>按钮大小</h1>

<p>通过 padding 属性改变按钮的内边距：</p>

<button class="button button1">10px 24px</button>

<button class="button button2">12px 28px</button>

<button class="button button3">14px 40px</button>

<button class="button button4">32px 16px</button>

<button class="button button5">16px</button>

</body>

</html>

## 89.4 圆角按钮

请使用 border-radius 属性为按钮添加圆角：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* 绿色 \*/

border: none;

color: white;

padding: 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.button1 {border-radius: 2px;}

.button2 {border-radius: 4px;}

.button3 {border-radius: 8px;}

.button4 {border-radius: 12px;}

.button5 {border-radius: 50%;}

</style>

</head>

<body>

<h1>圆角按钮</h1>

<p>通过 border-radius 属性为按钮添加圆角：</p>

<button class="button button1">2px</button>

<button class="button button2">4px</button>

<button class="button button3">8px</button>

<button class="button button4">12px</button>

<button class="button button5">50%</button>

</body>

</html>

## 89.5 彩色的按钮边框

请使用 border 属性为按钮添加彩色边框：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* 绿色 \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.button1 {

background-color: white;

color: black;

border: 2px solid #4CAF50;

}

.button2 {

background-color: white;

color: black;

border: 2px solid #008CBA;

}

.button3 {

background-color: white;

color: black;

border: 2px solid #f44336;

}

.button4 {

background-color: white;

color: black;

border: 2px solid #e7e7e7;

}

.button5 {

background-color: white;

color: black;

border: 2px solid #555555;

}

</style>

</head>

<body>

<h1>有颜色的按钮边框</h1>

<p>请使用 border 属性为按钮添加边框：</p>

<button class="button button1">绿色</button>

<button class="button button2">蓝色</button>

<button class="button button3">黑色</button>

<button class="button button4">灰色</button>

<button class="button button5">黑色</button>

</body>

</html>

## 89.6 可悬停按钮

当鼠标移动到按钮上方时，使用 :hover 选择器可更改按钮的样式。

提示：请使用 transition-duration 属性来确定“悬停”效果的速度：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* Green \*/

border: none;

color: white;

padding: 16px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

transition-duration: 0.4s;

cursor: pointer;

}

.button1 {

background-color: white;

color: black;

border: 2px solid #4CAF50;

}

.button1:hover {

background-color: #4CAF50;

color: white;

}

.button2 {

background-color: white;

color: black;

border: 2px solid #008CBA;

}

.button2:hover {

background-color: #008CBA;

color: white;

}

.button3 {

background-color: white;

color: black;

border: 2px solid #f44336;

}

.button3:hover {

background-color: #f44336;

color: white;

}

.button4 {

background-color: white;

color: black;

border: 2px solid #e7e7e7;

}

.button4:hover {background-color: #e7e7e7;}

.button5 {

background-color: white;

color: black;

border: 2px solid #555555;

}

.button5:hover {

background-color: #555555;

color: white;

}

</style>

</head>

<body>

<h1>可悬停的按钮</h1>

<p>使用 :hover 选择器在鼠标移动到按钮上时改变其样式。</p>

<p><b>提示：</b>请使用 transition-duration 属性来确定悬停效果的速度：</p>

<button class="button button1">绿色</button>

<button class="button button2">蓝色</button>

<button class="button button3">红色</button>

<button class="button button4">灰色</button>

<button class="button button5">黑色</button>

</body>

</html>

## 89.7 阴影按钮

请使用 box-shadow 属性为按钮添加阴影：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* Green \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

-webkit-transition-duration: 0.4s; /\* Safari \*/

transition-duration: 0.4s;

}

.button1 {

box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);

}

.button2:hover {

box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24),0 17px 50px 0 rgba(0,0,0,0.19);

}

</style>

</head>

<body>

<h1>阴影按钮</h1>

<p>使用 box-shadow 属性为按钮添加阴影：</p>

<button class="button button1">阴影按钮</button>

<button class="button button2">悬停时的阴影</button>

</body>

</html>

## 89.8 禁用的按钮

请使用 opacity 属性为按钮添加透明度（创建“禁用”外观）。

提示：您还可以添加带有 "not-allowed" 值的 cursor 属性，当您将鼠标悬停在按钮上时，该属性会显示 "no parking sign"（禁停标志）：

实例

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* 绿色 \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.disabled {

opacity: 0.6;

cursor: not-allowed;

}

</style>

</head>

<body>

<h1>被禁用的按钮</h1>

<p>使用 opacity 属性向按钮添加一定的透明度（使它看上去已被禁用）</p>

<button class="button">正常按钮</button>

<button class="button disabled">被禁用的按钮</button>

</body>

</html>

## 89.9 按钮宽度

默认情况下，按钮的大小取决于其文本内容（与内容的宽度一样）。请使用 width 属性来更改按钮的宽度：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

background-color: #4CAF50; /\* Green \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.button1 {width: 250px;}

.button2 {width: 50%;}

.button3 {width: 100%;}

</style>

</head>

<body>

<h1>按钮宽度</h1>

<p>使用 width 属性来改变按钮的宽度：</p>

<p><b>提示：</b>请使用像素设置固定宽度，并为响应式按钮使用百分百（例如其父元素的 50%）。请调整窗口大小来查看效果。</p>

<button class="button button1">250px</button><br>

<button class="button button2">50%</button><br>

<button class="button button3">100%</button>

</body>

</html>

## 89.10 按钮分组

删除外边距并向每个按钮添加 float:left，来创建按钮组：

<!DOCTYPE html>

<html>

<head>

<style>

.btn-group .button {

background-color: #4CAF50; /\* 绿色 \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

cursor: pointer;

float: left;

}

.btn-group .button:hover {

background-color: #3e8e41;

}

</style>

</head>

<body>

<h1>按钮组</h1>

<p>删除外边距并浮动按钮，来创建一个按钮组：</p>

<div class="btn-group">

<button class="button">Button</button>

<button class="button">Button</button>

<button class="button">Button</button>

<button class="button">Button</button>

</div>

<p style="clear:both"><br>请记得之后清除浮动，否则这个 p 元素会向按钮浮动。</p>

</body>

</html>

## 89.11 带边框的按钮组

使用 border 属性来创建带边框的按钮组：

<!DOCTYPE html>

<html>

<head>

<style>

.btn-group .button {

background-color: #4CAF50; /\* 绿色 \*/

border: 1px solid green;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

cursor: pointer;

float: left;

}

.btn-group .button:not(:last-child) {

border-right: none; /\* 阻止双边框 \*/

}

.btn-group .button:hover {

background-color: #3e8e41;

}

</style>

</head>

<body>

<h1>带边框的按钮组</h1>

<p>添加边框，来创建带按钮的按钮组：</p>

<div class="btn-group">

<button class="button">Button</button>

<button class="button">Button</button>

<button class="button">Button</button>

<button class="button">Button</button>

</div>

<p style="clear:both"><br>请记得之后清除浮动，否则这个 p 元素会向按钮浮动。</p>

</body>

</html>

## 89.12 垂直按钮组

使用 display:block 取代 float:left 将按钮上下分组，而不是并排：

<!DOCTYPE html>

<html>

<head>

<style>

.btn-group .button {

background-color: #4CAF50; /\* 绿色 \*/

border: 1px solid green;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

font-size: 16px;

cursor: pointer;

width: 150px;

display: block;

}

.btn-group .button:not(:last-child) {

border-bottom: none; /\* 阻止双边框 \*/

}

.btn-group .button:hover {

background-color: #3e8e41;

}

</style>

</head>

<body>

<h1>垂直按钮分组</h1>

<div class="btn-group">

<button class="button">Button</button>

<button class="button">Button</button>

<button class="button">Button</button>

<button class="button">Button</button>

</div>

</body>

</html>

## 89.13 图像上的按钮

<!DOCTYPE html>

<html>

<head>

<style>

.container {

position: relative;

width: 100%;

max-width: 400px;

}

.container img {

width: 100%;

height: auto;

}

.container .btn {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

background-color: #f1f1f1;

color: black;

font-size: 16px;

padding: 16px 30px;

border: none;

cursor: pointer;

border-radius: 5px;

text-align: center;

}

.container .btn:hover {

background-color: black;

color: white;

}

</style>

</head>

<body>

<h1>图像上的按钮</h1>

<p>为图像添加按钮：</p>

<div class="container">

<img src="img\_lights.jpg" alt="Snow" style="width:100%">

<button class="btn">按钮</button>

</div>

</body>

</html>

## 89.14 动画按钮

**实例 1**

在鼠标悬停时添加箭头：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

display: inline-block;

border-radius: 4px;

background-color: #f4511e;

border: none;

color: #FFFFFF;

text-align: center;

font-size: 28px;

padding: 20px;

width: 200px;

transition: all 0.5s;

cursor: pointer;

margin: 5px;

}

.button span {

cursor: pointer;

display: inline-block;

position: relative;

transition: 0.5s;

}

.button span:after {

content: '\00bb';

position: absolute;

opacity: 0;

top: 0;

right: -20px;

transition: 0.5s;

}

.button:hover span {

padding-right: 25px;

}

.button:hover span:after {

opacity: 1;

right: 0;

}

</style>

</head>

<body>

<h1>带动画效果的按钮</h1>

<button class="button" style="vertical-align:middle"><span>请悬停在我上方</span></button>

</body>

</html>

**实例 2**

添加点击时的“按键按下”效果：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

display: inline-block;

padding: 15px 25px;

font-size: 24px;

cursor: pointer;

text-align: center;

text-decoration: none;

outline: none;

color: #fff;

background-color: #4CAF50;

border: none;

border-radius: 15px;

box-shadow: 0 9px #999;

}

.button:hover {background-color: #3e8e41}

.button:active {

background-color: #3e8e41;

box-shadow: 0 5px #666;

transform: translateY(4px);

}

</style>

</head>

<body>

<h1>带动画效果的按钮 - 按键效果</h1>

<button class="button">请点击我</button>

</body>

</html>

**实例 3**

鼠标悬停时淡入：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

.button {

background-color: #f4511e;

border: none;

color: white;

padding: 16px 32px;

text-align: center;

font-size: 16px;

margin: 4px 2px;

opacity: 0.6;

transition: 0.3s;

display: inline-block;

text-decoration: none;

cursor: pointer;

}

.button:hover {opacity: 1}

</style>

</head>

<body>

<h1>淡入按钮 - 淡入效果</h1>

<button class="button">请悬停在我上方</button>

</body>

</html>

**实例 4**

添加点击时的“涟漪”效果：

<!DOCTYPE html>

<html>

<head>

<style>

.button {

position: relative;

background-color: #4CAF50;

border: none;

font-size: 28px;

color: #FFFFFF;

padding: 20px;

width: 200px;

text-align: center;

transition-duration: 0.4s;

text-decoration: none;

overflow: hidden;

cursor: pointer;

}

.button:after {

content: "";

background: #f1f1f1;

display: block;

position: absolute;

padding-top: 300%;

padding-left: 350%;

margin-left: -20px !important;

margin-top: -120%;

opacity: 0;

transition: all 0.8s

}

.button:active:after {

padding: 0;

margin: 0;

opacity: 1;

transition: 0s

}

</style>

</head>

<body>

<h1>带动画效果的按钮 - 涟漪效果</h1>

<button class="button">请点击我</button>

</body>

</html>

# 第90章 CSS 分页实例

## 90.1 简单的分页

如果网站上有很多页面，那么您可能希望在每张页面上添加某种分页功能：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

}

</style>

</head>

<body>

<h1>简单分页</h1>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a href="#">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.2 活动的可悬停分页

用 .active 类突出显示当前页面，并在鼠标移到它们上方时使用 :hover 选择器更改每个页面链接的颜色：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<h1>活动的可悬停分页</h1>

<p>请把鼠标移动到数字上：</p>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a class="active" href="#">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.3 圆角的活动可悬停分页

如果您需要圆角的 "active" 和 "hover" 按钮，请添加 border-radius 属性：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

border-radius: 5px;

}

.pagination a:hover:not(.active) {

background-color: #ddd;

border-radius: 5px;

}

</style>

</head>

<body>

<h1>活动的可悬停的圆角分页</h1>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a href="#" class="active">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.4 可悬停的过渡效果

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

transition: background-color .3s;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<h1>悬停时的过渡效果</h1>

<p>请把鼠标移动到数字上。</p>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a href="#" class="active">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.5 带边框的分页

请使用 border 属性为分页添加边框：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

transition: background-color .3s;

border: 1px solid #ddd;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

border: 1px solid #4CAF50;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<h1>带边框的分页</h1>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a href="#" class="active">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.6 圆角边框

提示：在分页的第一个和最后一个链接中添加圆角边框：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

border: 1px solid #ddd;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

border: 1px solid #4CAF50;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

.pagination a:first-child {

border-top-left-radius: 5px;

border-bottom-left-radius: 5px;

}

.pagination a:last-child {

border-top-right-radius: 5px;

border-bottom-right-radius: 5px;

}

</style>

</head>

<body>

<h1>带圆角边框的分页</h1>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a class="active" href="#">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.7 链接之间的空间

提示：如果不想组合页面链接，请添加 margin 属性：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

transition: background-color .3s;

border: 1px solid #ddd;

margin: 0 4px;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

border: 1px solid #4CAF50;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<h1>带外边距的分页：</h1>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a href="#" class="active">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.8 分页尺寸

请使用 font-size 属性更改分页的大小：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

transition: background-color .3s;

border: 1px solid #ddd;

font-size: 22px;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

border: 1px solid #4CAF50;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<h1>分页尺寸</h1>

<p>更改 font-size 属性以使分页变小或变大。</p>

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a href="#" class="active">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</body>

</html>

## 90.9 居中的分页

如需居中分页，请使用设置了 text-align:center 的容器元素（如 <div>）包围它：

<!DOCTYPE html>

<html>

<head>

<style>

.center {

text-align: center;

}

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

transition: background-color .3s;

border: 1px solid #ddd;

margin: 0 4px;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

border: 1px solid #4CAF50;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<h1>居中分页</h1>

<div class="center">

<div class="pagination">

<a href="#">«</a>

<a href="#">1</a>

<a href="#" class="active">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">»</a>

</div>

</div>

</body>

</html>

## 90.10 更多实例

实例

前进/后退按钮

导航分页：

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

transition: background-color .3s;

border: 1px solid #ddd;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

border: 1px solid #4CAF50;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<p>前进后退按钮：</p>

<div class="pagination">

<a href="#">❮</a>

<a href="#">❯</a>

</div>

<p>导航分页：</p>

<div class="pagination">

<a href="#" class="active">Home</a>

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</body>

</html>

## 90.11 面包屑

分页的另一种形式是所谓的“面包屑”（breadcrumbs）：

<!DOCTYPE html>

<html>

<head>

<style>

ul.breadcrumb {

padding: 8px 16px;

list-style: none;

background-color: #eee;

}

ul.breadcrumb li {display: inline;}

ul.breadcrumb li+li:before {

padding: 8px;

color: black;

content: "/\00a0";

}

ul.breadcrumb li a {color: green;}

</style>

</head>

<body>

<h1>面包屑分页</h1>

<ul class="breadcrumb">

<li><a href="#">Home</a></li>

<li><a href="#">Pictures</a></li>

<li><a href="#">Summer 15</a></li>

<li>Italy</li>

</ul>

</body>

</html>

# 第91章 CSS 多列

## 91.1 CSS 多列属性

在本章中，您将学到以下多列属性：

column-count

column-gap

column-rule-style

column-rule-width

column-rule-color

column-rule

column-span

column-width

## 91.2 CSS 创建多列

column-count 属性规定元素应被划分的列数。

下面的例子将 <div> 元素中的文本分为 3 列：

实例

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

}

</style>

</head>

<body>

<div class="newspaper">

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

## 91.3 CSS 指定列之间的间隙

column-gap 属性规定列之间的间隔。

下面的例子指定列之间的间距为 40 像素：

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

column-gap: 40px;

}

</style>

</head>

<body>

<div class="newspaper">

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

## 91.4 CSS 列规则

column-rule-style 属性规定列之间的规则样式：

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

column-gap: 40px;

column-rule-style: solid;

}

</style>

</head>

<body>

<div class="newspaper">

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

column-rule-width 属性规定列之间的规则宽度：

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

column-gap: 40px;

column-rule-style: solid;

column-rule-width: 1px;

}

</style>

</head>

<body>

<div class="newspaper">

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

column-rule-color 属性规定列之间的规则的颜色：

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

column-gap: 40px;

column-rule-style: solid;

column-rule-width: 1px;

column-rule-color: lightblue;

}

</style>

</head>

<body>

<div class="newspaper">

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

column-rule 属性是用于设置上面所有 column-rule-\* 属性的简写属性。

下例设置了列之间的规则的宽度、样式和颜色：

实例

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

column-gap: 40px;

column-rule: 1px solid lightblue;

}

</style>

</head>

<body>

<div class="newspaper">

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

## 91.5 指定元素应该横跨多少列

column-span 属性规定元素应跨越多少列。

下例规定了<h2> 元素应跨所有列：

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

column-gap: 40px;

column-rule: 1px solid lightblue;

}

h2 {

column-span: all;

}

</style>

</head>

<body>

<div class="newspaper">

<h2>第一回 宴桃园豪杰三结义 斩黄巾英雄首立功</h2>

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

## 91.6 指定列宽度

column-width 属性为列指定建议的最佳宽度。

下例规定了列的建议最佳宽度应为 100px：

<!DOCTYPE html>

<html>

<head>

<style>

.newspaper {

column-count: 3;

column-width: 100px;

}

</style>

</head>

<body>

<div class="newspaper">

话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。

</div>

</body>

</html>

## 91.7 CSS 多列属性

下表列出了所有的多列属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| column-count | 规定元素应划分的列数。 |
| column-fill | 规定如何填充列。 |
| column-gap | 指定列之间的间隙。 |
| column-rule | 用于设置所有 column-rule-\* 属性的简写属性。 |
| column-rule-color | 规定列之间规则的颜色。 |
| column-rule-style | 规定列之间的规则样式。 |
| column-rule-width | 规定列之间的规则宽度。 |
| column-span | 规定一个元素应该跨越多少列。 |
| column-width | 为列指定建议的最佳宽度。 |
| columns | 用于设置 column-width 和 column-count 的简写属性。 |

# 第92章 CSS 用户界面

## 92.1 CSS 用户界面

在本章中，您将学到以下 CSS 用户界面属性：

resize

outline-offset

## 92.2 CSS 调整大小

resize 属性规定元素是否应（以及如何）被用户调整大小。

注意：Internet Explorer 不支持 resize 属性。

下例只允许用户调整 <div> 元素的宽度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 2px solid;

padding: 20px;

width: 300px;

resize: horizontal;

overflow: auto;

}

</style>

</head>

<body>

<h1>resize 属性</h1>

<div>

<p>只允许用户调整 div 元素的宽度。</p>

<p>如需调整：请点击并拖动 div 元素的右下角。</p>

</div>

</body>

</html>

下例只允许用户调整 <div> 元素的高度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 2px solid;

padding: 20px;

width: 300px;

resize: vertical;

overflow: auto;

}

</style>

</head>

<body>

<h1>resize 属性</h1>

<div>

<p>只允许用户调整 div 元素的高度。</p>

<p>如需调整：请点击并拖动 div 元素的右下角。</p>

</div>

</body>

</html>

下例允许用户能够调整 <div> 元素的高度和宽度：

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 2px solid;

padding: 20px;

width: 300px;

resize: both;

overflow: auto;

}

</style>

</head>

<body>

<h1>resize 属性</h1>

<div>

<p>允许用户调整 div 元素的高度和宽度。</p>

<p>如需调整：请点击并拖动 div 元素的右下角。</p>

</div>

</body>

</html>

在许多浏览器中，<textarea> 默认可调整大小。在这里，我们使用了 resize 属性来禁用这种可缩放性：

<!DOCTYPE html>

<html>

<head>

<style>

textarea#test {

resize: none;

}

</style>

</head>

<body>

<h1>resize 属性</h1>

<p>在很多浏览器中，textarea 元素默认是可调整尺寸的。在本例中，我们已用 resize 属性来禁用这种可调整性：</p>

<textarea id="test">Textarea - 不可调整

## 92.3 CSS 轮廓偏移

outline-offset 属性在轮廓与元素的边缘边框之间添加空间。

注意：轮廓与边框不同！与边框不同，轮廓线是在元素边框之外绘制的，并且可能与其他内容重叠。同时，轮廓也不是元素尺寸的一部分：元素的总宽度和高度不受轮廓线宽度的影响。

下面的例子使用 outline-offset 属性添加了边框和轮廓之间的空间：

<!DOCTYPE html>

<html>

<head>

<style>

div.ex1 {

margin: 20px;

border: 1px solid black;

outline: 4px solid red;

outline-offset: 15px;

}

div.ex2 {

margin: 10px;

border: 1px solid black;

outline: 5px dashed blue;

outline-offset: 5px;

}

</style>

</head>

<body>

<h1>outline-offset 属性</h1>

<div class="ex1">这个 div 有 4 像素的红色实线轮廓，位于边框边缘外 15 像素处。</div>

<br>

<div class="ex2">这个 div 有 5 像素的蓝色虚线轮廓，位于边框边缘外 5 像素处。</div>

</body>

</html>

## 92.4 CSS 用户界面属性

下表列出了所有用户界面属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| outline-offset | 在轮廓和元素的边框边缘之间添加空间。 |
| resize | 规定元素是否可由用户调整大小。 |

# 第93章 CSS 变量

var() 函数用于插入 CSS 变量的值。

CSS 变量可以访问 DOM，这意味着您可以创建具有局部或全局范围的变量，使用 JavaScript 来修改变量，以及基于媒体查询来修改变量。

使用 CSS 变量的一种好方法涉及设计的颜色。您可以将它们放在变量中，而不必一遍又一遍地复制和粘贴相同的颜色。

**传统方式**

以下例子显示了在样式表中定义一些颜色的传统方式（通过为每个特定元素定义要使用的颜色）：

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: #1e90ff;

}

h2 {

border-bottom: 2px solid #1e90ff;

}

.container {

color: #1e90ff;

background-color: #ffffff;

padding: 15px;

}

button {

background-color: #ffffff;

color: #1e90ff;

border: 1px solid #1e90ff;

padding: 5px;

}

</style>

</head>

<body>

<h1>传统方式</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>

<button>Yes</button>

<button>No</button>

</p>

</div>

</body>

</html>

## 93.1 var() 函数的语法

var() 函数用于插入 CSS 变量的值。

var() 函数的语法如下：

var(name, value)

|  |  |
| --- | --- |
| 值 | 描述 |
| name | 必需。变量名（以两条破折号开头）。 |
| value | 可选。回退值（在未找到变量时使用） |

注释：变量名称必须以两个破折号（--）开头，且区分大小写！

## 93.2 var() 如何工作

首先：CSS 变量可以有全局或局部作用域。

全局变量可以在整个文档中进行访问/使用，而局部变量只能在声明它的选择器内部使用。

如需创建具有全局作用域的变量，请在 :root 选择器中声明它。 :root 选择器匹配文档的根元素。

如需创建具有局部作用域的变量，请在将要使用它的选择器中声明它。

下面的例子与上面的例子相同，但是在这里我们使用 var() 函数。

首先，我们声明两个全局变量（--blue 和 --white）。然后，我们使用 var() 函数稍后在样式表中插入变量的值：

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--blue: #1e90ff;

--white: #ffffff;

}

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

}

button {

background-color: var(--white);

color: var(--blue);

border: 1px solid var(--blue);

padding: 5px;

}

</style>

</head>

<body>

<h1>使用 var() 函数</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>

<button>Yes</button>

<button>No</button>

</p>

</div>

</body>

</html>

使用 var() 有如下优势：

使代码更易于阅读（更容易理解）

使修改颜色值更加容易

如需将蓝色和白色改为较柔和的蓝色和白色，您只需要修改两个变量值：

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--blue: #6495ed;

--white: #faf0e6;

}

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

}

button {

background-color: var(--white);

color: var(--blue);

border: 1px solid var(--blue);

padding: 5px;

}

</style>

</head>

<body>

<h1>使用 var() 函数</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>

<button>Yes</button>

<button>No</button>

</p>

</div>

</body>

</html>

## 93.3 CSS var() 函数

|  |  |
| --- | --- |
| 函数 | 描述 |
| var() | 插入 CSS 变量的值。 |

# 第94章 CSS 覆盖变量

## 94.1 用局部变量覆盖全局变量

从上一页我们了解到，可以在整个文档中访问/使用全局变量，而局部变量只能在声明它的选择器内使用。

请看上一页中的例子：

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--blue: #1e90ff;

--white: #ffffff;

}

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

}

button {

background-color: var(--white);

color: var(--blue);

border: 1px solid var(--blue);

padding: 5px;

}

</style>

</head>

<body>

<h1>使用 var() 函数</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>

<button>Yes</button>

<button>No</button>

</p>

</div>

</body>

</html>

有时，我们希望变量仅在页面的特定部分中进行更改。

假设我们想要按钮元素使用不同的蓝色。那么，我们可以在 button 选择器内重新声明 --blue 变量。当我们在这个选择器中使用 var(--blue) 时，它将使用此处声明的局部 --blue 变量值。

我们看到局部的 --blue 变量会覆盖 button 元素的全局 --blue 变量：

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--blue: #1e90ff;

--white: #ffffff;

}

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

}

button {

--blue: #0000ff;

background-color: var(--white);

color: var(--blue);

border: 1px solid var(--blue);

padding: 5px;

}

</style>

</head>

<body>

<h1>用局部变量覆盖全局变量</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>

<button>Yes</button>

<button>No</button>

</p>

</div>

</body>

</html>

## 94.2 添加一个新的局部变量

如果只在一个地方使用一个变量，我们也可以声明一个新的局部变量，就像这样：

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--blue: #1e90ff;

--white: #ffffff;

}

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

}

button {

--button-blue: #0000ff;

background-color: var(--white);

color: var(--button-blue);

border: 1px solid var(--button-blue);

padding: 5px;

}

</style>

</head>

<body>

<h1>新的局部变量</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>

<button>Yes</button>

<button>No</button>

</p>

</div>

</body>

</html>

# 第95章 CSS 使用 JavaScript 更改变量

## 95.1 使用 JavaScript 更改变量

CSS 变量可以访问 DOM，这意味着您可以通过 JavaScript 更改它们。

这个例子说明了如何创建脚本来显示并更改上一页中使用的示例中的 --blue 变量。此刻，如果您不熟悉 JavaScript，不要担心。您可以在我们的 JavaScript 教程中学到有关 JavaScript 的更多知识：

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--blue: #1e90ff;

--white: #ffffff;

}

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

}

.container button {

background-color: var(--white);

color: var(--blue);

border: 1px solid var(--blue);

padding: 5px;

}

</style>

<script>

// 获取根元素

var r = document.querySelector(':root');

// 创建获取变量值的函数

function myFunction\_get() {

// 获取根的样式（属性和值）

var rs = getComputedStyle(r);

// 弹出 --blue 变量的值

alert("The value of --blue is: " + rs.getPropertyValue('--blue'));

}

// 创建设置变量值的函数

function myFunction\_set() {

// 把变量 --blue 的值设置为另一个值（在这里是 "lightblue"）

r.style.setProperty('--blue', 'lightblue');

}

</script>

</head>

<body>

<h1>使用 JavaScript 获取和更改 CSS 变量</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China.</p>

<p>

<button>Yes</button>

<button>No</button>

</p>

</div>

<br>

<button type="button" onclick="myFunction\_get()">使用 JavaScript 来获取 CSS 变量</button>

<button type="button" onclick="myFunction\_set()">使用 JavaScript 来更改 CSS 变量</button>

</body>

</html>

# 第96章 CSS 在媒体查询中使用变量

## 96.1 在媒体查询中使用变量

现在，我们希望在媒体查询中修改变量值。

提示：媒体查询旨在为不同的设备（显示器、平板电脑、手机等）定义不同的样式规则。您可以在“媒体查询”一章中学习更多有关媒体查询的知识。

在这里，我们首先为 .container 类声明一个名为 --fontsize 的新局部变量。我们将其值设置为 25 像素。然后我们在 .container 类中进一步使用它。然后，我们创建一个 @media 规则，内容为“当浏览器的宽度为 450px 或更宽时，将 .container 类的 --fontsize 变量值更改为 50px。”

这是完整的实例：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

/\* 变量声明 \*/

:root {

--blue: #1e90ff;

--white: #ffffff;

}

.container {

--fontsize: 25px;

}

/\* 样式 \*/

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

font-size: var(--fontsize);

}

@media screen and (min-width: 450px) {

.container {

--fontsize: 50px;

}

}

</style>

</head>

<body>

<h1>在媒体查询中使用变量</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>当浏览器的宽度小于 450px 时，此 div 的字体大小为 25px。当其为 450px 或更宽时，请将 --fontsize 变量值设置为 50px。请调整浏览器窗口的大小来查看效果。</p>

</div>

</body>

</html>

这是另一个例子，在其中我们还更改了 @media 规则中 --blue 变量的值：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

/\* 变量声明 \*/

:root {

--blue: #1e90ff;

--white: #ffffff;

}

.container {

--fontsize: 25px;

}

/\* 样式 \*/

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

font-size: var(--fontsize);

}

@media screen and (min-width: 450px) {

.container {

--fontsize: 50px;

}

:root {

--blue: lightblue;

}

}

</style>

</head>

<body>

<h1>在媒体查询中使用变量</h1>

<div class="container">

<h2>Welcome to Shanghai!</h2>

<p>当浏览器的宽度为 450px 或更宽时，将 --fontsize 变量值设置为 50px，将 --blue 变量值设置为 lightblue。请调整浏览器窗口的大小以查看效果。</p>

</div>

</body>

</html>

# 第97章 CSS Box Sizing

## 97.1 CSS Box Sizing

CSS box-sizing 属性允许我们在元素的总宽度和高度中包括内边距（填充）和边框。

## 97.2 假如不指定 CSS box-sizing 属性

默认情况下，元素的宽度和高度是这样计算的：

width + padding + border = 元素的实际宽度

height + padding + border = 元素的实际高度

这意味着：当您设置元素的宽度/高度时，该元素通常看起来比您设置的更大（因为元素的边框和内边距已被添加到元素的指定宽度/高度中）。

下图展示了两个有相同指定宽度和高度的 <div> 元素：

<!DOCTYPE html>

<html>

<head>

<style>

.div1 {

width: 300px;

height: 100px;

border: 1px solid blue;

}

.div2 {

width: 300px;

height: 100px;

padding: 50px;

border: 1px solid red;

}

</style>

</head>

<body>

<div class="div1">这个 div 更小（宽度为 300 像素，高度为 100 像素）。</div>

<br>

<div class="div2">这个 div 更大（宽度也是 300 像素，高度也是 100 像素）。</div>

</body>

</html>

box-sizing 属性解决了这个问题。

## 97.3 如果使用 CSS box-sizing 属性

box-sizing 属性允许我们在元素的总宽度和高度中包括内边距和边框。

如果在元素上设置了 box-sizing: border-box;，则宽度和高度会包括内边距和边框：

<!DOCTYPE html>

<html>

<head>

<style>

.div1 {

width: 300px;

height: 100px;

border: 1px solid blue;

box-sizing: border-box;

}

.div2 {

width: 300px;

height: 100px;

padding: 50px;

border: 1px solid red;

box-sizing: border-box;

}

</style>

</head>

<body>

<div class="div1">现在，两个 div 的尺寸是一样的！</div>

<br>

<div class="div2">Hello！</div>

</body>

</html>

由于使用 box-sizing: border-box; 的效果如此之好，许多开发人员希望页面上的所有元素都能够以这种方式工作。

下面的代码能够确保以这种更直观的方式调整所有元素的大小。许多浏览器已经在对许多表单元素使用 box-sizing: border-box;（但并非全部 - 这就是为什么 input 和 textarea 在 width: 100%; 时看起来不同）。

将其应用于所有元素是安全且明智的：

<!DOCTYPE html>

<html>

<head>

<style>

body {

margin: 0;

}

\* {

box-sizing: border-box;

}

input, textarea {

width: 100%;

}

</style>

</head>

<body>

<form action="/action\_page.php">

First name:<br>

<input type="text" name="firstname" value="Mickey"><br>

Last name:<br>

<input type="text" name="lastname" value="Mouse"><br>

Comments:<br>

<textarea name="message" rows="5" cols="30">

## 97.4 CSS Box Sizing 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| box-sizing | 定义元素宽度和高度的计算方式：它们是否应包含内边距（padding）和边框。 |

# 第98章 CSS Flexbox

## 98.1 CSS Flexbox 布局模块

在 Flexbox 布局模块（问世）之前，可用的布局模式有以下四种：

块（Block），用于网页中的部分（节）

行内（Inline），用于文本

表，用于二维表数据

定位，用于元素的明确位置

弹性框布局模块，可以更轻松地设计灵活的响应式布局结构，而无需使用浮动或定位。

## 98.2 Flexbox 元素

如需开始使用 Flexbox 模型，您需要首先定义 Flex 容器。

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

margin: 10px;

padding: 20px;

font-size: 30px;

}

</style>

</head>

<body>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

<p>弹性布局中必须有一个 <em>display</em> 属性设置为 <em>flex</em> 的父元素。</p>

<p>弹性容器的直接子元素会自动成为弹性项目。</p>

</body>

</html>

## 98.3 父元素（容器）

通过将 display 属性设置为 flex，flex 容器将可伸缩：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

margin: 10px;

padding: 20px;

font-size: 30px;

}

</style>

</head>

<body>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

<p>弹性布局中必须有一个 <em>display</em> 属性设置为 <em>flex</em> 的父元素。</p>

<p>弹性容器的直接子元素会自动成为弹性项目。</p>

</body>

</html>

## 98.4 以下是 flex 容器属性：

flex-direction

flex-wrap

flex-flow

justify-content

align-items

align-content

## 98.5 flex-direction 属性

flex-direction 属性定义容器要在哪个方向上堆叠 flex 项目。

实例

column 值设置垂直堆叠 flex 项目（从上到下）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-direction: column;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-direction 属性</h1>

<p>"flex-direction: column;" 垂直堆叠弹性项目（从上到下）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

column-reverse 值垂直堆叠 flex 项目（但从下到上）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-direction: column-reverse;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-direction 属性</h1>

<p>"flex-direction: column-reverse;" 垂直堆叠弹性项目（但是从下到上）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

row 值水平堆叠 flex 项目（从左到右）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-direction: row;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-direction 属性</h1>

<p>"flex-direction: row;" 水平地并排弹性项目（从左到右）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

row-reverse 值水平堆叠 flex 项目（但从右到左）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-direction: row-reverse;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-direction 属性</h1>

<p>"flex-direction: row-reverse;" 水平地并排弹性项目（但是从右向左）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

## 98.6 flex-wrap 属性

flex-wrap 属性规定是否应该对 flex 项目换行。

下面的例子包含 12 个 flex 项目，以便更好地演示 flex-wrap 属性。

wrap 值规定 flex 项目将在必要时进行换行：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-wrap: wrap;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-wrap 属性</h1>

<p>"flex-wrap: wrap;" 规定 flex 项目将在必要时进行换行：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

<p>Try resizing the browser window.</p>

</body>

</html>

实例

nowrap 值规定将不对 flex 项目换行（默认）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-wrap: nowrap;

background-color: DodgerBlue;

}

.flex-container>div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-wrap 属性</h1>

<p>"flex-wrap: nowrap;" 规定将不对 flex 项目换行（默认）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

<p>请尝试调整浏览器窗口的尺寸。</p>

</body>

</html>

实例

wrap-reverse 值规定如有必要，弹性项目将以相反的顺序换行：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-wrap: wrap-reverse;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-wrap 属性</h1>

<p>"flex-wrap: wrap-reverse;" 规定弹性项目将以相反的顺序换行（如有必要）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

<p>Try resizing the browser window.</p>

</body>

</html>

## 98.7 flex-flow 属性

flex-flow 属性是用于同时设置 flex-direction 和 flex-wrap 属性的简写属性。

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-flow: row wrap;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-flow 属性</h1>

<p>flex-flow 属性是 flex-direction 和 flex-wrap 属性的简写属性。</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

<p>请尝试调整浏览器窗口的大小。</p>

</body>

</html>

## 98.8 justify-content 属性

justify-content 属性用于对齐 flex 项目：

center 值将 flex 项目在容器的中心对齐：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: center;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>"justify-content: center;" 在容器中央对齐弹性项目：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

flex-start 值将 flex 项目在容器的开头对齐（默认）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: flex-start;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>"justify-content: flex-start;" 在容器开头对齐弹性项目（默认）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

flex-end 值将 flex 项目在容器的末端对齐：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: flex-end;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>"justify-content: flex-end;" 在容器末端对齐弹性项目：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

space-around 值显示行之前、之间和之后带有空格的 flex 项目：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: space-around;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>"justify-content: space-around;" 显示行之前、之间和之后带有空格的 flex 项目：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

space-between 值显示行之间有空格的 flex 项目：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: space-between;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>"justify-content: space-between;" 显示行之间有空格的 flex 项目：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

## 98.9 align-items 属性

align-items 属性用于垂直对齐 flex 项目。

在这些例子中，我们使用 200 像素高的容器，以便更好地演示 align-items 属性。

center 值将 flex 项目在容器中间对齐：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

align-items: center;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-items 属性</h1>

<p>"align-items: center;" 将 flex 项目在容器中间对齐：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

flex-start 值将 flex 项目在容器顶部对齐：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

align-items: flex-start;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-items 属性</h1>

<p>"align-items: flex-start;" 将 flex 项目在容器顶部对齐：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

flex-end 值将弹性项目在容器底部对齐：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

align-items: flex-end;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-items 属性</h1>

<p>"align-items: flex-end;" 将弹性项目在容器底部对齐：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

stretch 值拉伸 flex 项目以填充容器（默认）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

align-items: stretch;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-items 属性</h1>

<p>"align-items: stretch;" 拉伸 flex 项目以填充容器（默认）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

实例

baseline 值使 flex 项目基线对齐：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

align-items: baseline;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-items 属性</h1>

<p>"align-items: baseline;" 使 flex 项目基线对齐：</p>

<div class="flex-container">

<div><h1>1</h1></div>

<div><h6>2</h6></div>

<div><h3>3</h3></div>

<div><small>4</small></div>

</div>

</body>

</html>

注意：该例使用不同的 font-size 来演示项目已按文本基线对齐：

## 98.9 align-content 属性

align-content 属性用于对齐弹性线。

在这些例子中，我们使用 600 像素高的容器，并将 flex-wrap 属性设置为 wrap，以便更好地演示 align-content 属性。

实例

space-between 值显示的弹性线之间有相等的间距：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 600px;

flex-wrap: wrap;

align-content: space-between;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>"align-content: space-between;" 显示的弹性线之间有相等的间距：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

实例

space-around 值显示弹性线在其之前、之间和之后带有空格：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 600px;

flex-wrap: wrap;

align-content: space-around;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>"align-content: space-around;" 显示弹性线在其之前、之间和之后带有空格：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

实例

stretch 值拉伸弹性线以占据剩余空间（默认）：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 600px;

flex-wrap: wrap;

align-content: stretch;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>"align-content: stretch;" 拉伸弹性线以占据剩余空间（默认）：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

实例

center 值在容器中间显示弹性线：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 600px;

flex-wrap: wrap;

align-content: center;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>"align-content: center;" 在容器中间显示弹性线：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

实例

flex-start 值在容器开头显示弹性线：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 600px;

flex-wrap: wrap;

align-content: flex-start;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>"align-content: flex-start;" 在容器开头显示弹性线：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

实例

flex-end 值在容器的末尾显示弹性线：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 600px;

flex-wrap: wrap;

align-content: flex-end;

background-color: DodgerBlue;

}

.flex-container > div {

background-color: #f1f1f1;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>"align-content: flex-end;" 在容器的末尾显示弹性线：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

<div>11</div>

<div>12</div>

</div>

</body>

</html>

## 98.10 完美的居中

在下面的例子中，我们会解决一个非常常见的样式问题：完美居中。

解决方案：将 justify-content 和 align-items 属性设置为居中，然后 flex 项目将完美居中：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: center;

align-items: center;

height: 300px;

background-color: DodgerBlue;

}

.flex-container>div {

background-color: #f1f1f1;

color: white;

width: 100px;

height: 100px;

}

</style>

</head>

<body>

<h1>完美的居中</h1>

<p>如果容器的 justify-content 和 align-items 属性都设置为 <em>center</em>，则项目会在两个轴的方向同时居中。</p>

<div class="flex-container">

<div></div>

</div>

</body>

</html>

## 98.11 子元素（项目）

flex 容器的直接子元素会自动成为弹性（flex）项目。

上面的元素代表一个灰色 flex 容器内的四个蓝色 flex 项目。

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

background-color: #f1f1f1;

}

.flex-container > div {

background-color: DodgerBlue;

color: white;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>弹性项目</h1>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

</div>

<p>弹性容器的所有直接子元素会（自动）成为弹性项目。</p>

</body>

</html>

**用于弹性项目的属性有：**

order

flex-grow

flex-shrink

flex-basis

flex

align-self

## 98.12 order 属性

order 属性规定 flex 项目的顺序。

代码中的首个 flex 项目不必在布局中显示为第一项。

order 值必须是数字，默认值是 0。

实例

order 属性可以改变 flex 项目的顺序：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

align-items: stretch;

background-color: #f1f1f1;

}

.flex-container>div {

background-color: DodgerBlue;

color: white;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>order 属性</h1>

<p>使用 order 属性可以根据需要对 flex 项目进行排序：</p>

<div class="flex-container">

<div style="order: 3">1</div>

<div style="order: 2">2</div>

<div style="order: 4">3</div>

<div style="order: 1">4</div>

</div>

</body>

</html>

## 98.13 flex-grow 属性

flex-grow 属性规定某个 flex 项目相对于其余 flex 项目将增长多少。

该值必须是数字，默认值是 0。

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

align-items: stretch;

background-color: #f1f1f1;

}

.flex-container > div {

background-color: DodgerBlue;

color: white;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-grow 属性</h1>

<p>使第三个弹性项目的增长速度比其他弹性项目快八倍：</p>

<div class="flex-container">

<div style="flex-grow: 1">1</div>

<div style="flex-grow: 1">2</div>

<div style="flex-grow: 8">3</div>

</div>

</body>

</html>

## 98.14 flex-shrink 属性

flex-shrink 属性规定某个 flex 项目相对于其余 flex 项目将收缩多少。

该值必须是数字，默认值是 0。

实例

不要让第三个弹性项目收缩得与其他弹性项目一样多：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

align-items: stretch;

background-color: #f1f1f1;

}

.flex-container>div {

background-color: DodgerBlue;

color: white;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-shrink 属性</h1>

<p>不要让第三个弹性项目收缩得与其他弹性项目一样多：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div style="flex-shrink: 0">3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

<div>9</div>

<div>10</div>

</div>

</body>

</html>

## 98.15 flex-basis 属性

flex-basis 属性规定 flex 项目的初始长度。

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

align-items: stretch;

background-color: #f1f1f1;

}

.flex-container > div {

background-color: DodgerBlue;

color: white;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex-basis 属性</h1>

<p>将第三个弹性项目的初始长度设置为 200 像素：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div style="flex-basis:200px">3</div>

<div>4</div>

</div>

</body>

</html>

## 98.16 flex 属性

flex 属性是 flex-grow、flex-shrink 和 flex-basis 属性的简写属性。

实例

使第三个弹性项目不可增长（0），不可收缩（0），且初始长度为 200 像素：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

align-items: stretch;

background-color: #f1f1f1;

}

.flex-container>div {

background-color: DodgerBlue;

color: white;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>flex 属性</h1>

<p>使第三个弹性项目不可增长（0），不可收缩（0），且初始长度为 200 像素：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div style="flex: 0 0 200px">3</div>

<div>4</div>

</div>

</body>

</html>

## 98.17 align-self 属性

align-self 属性规定弹性容器内所选项目的对齐方式。

align-self 属性将覆盖容器的 align-items 属性所设置的默认对齐方式。

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

background-color: #f1f1f1;

}

.flex-container > div {

background-color: DodgerBlue;

color: white;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-self 属性</h1>

<p>"align-self: center;" 在容器中间对齐所选的弹性项目：</p>

<div class="flex-container">

<div>1</div>

<div>2</div>

<div style="align-self: center">3</div>

<div>4</div>

</div>

<p>align-self 会覆盖容器的 align-items 属性。</p>

</body>

</html>

实例

将第二个弹性项目在容器顶部对齐，将第三个弹性项目在容器底部对齐：

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

height: 200px;

background-color: #f1f1f1;

}

.flex-container > div {

background-color: DodgerBlue;

color: white;

width: 100px;

margin: 10px;

text-align: center;

line-height: 75px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-self 属性</h1>

<p>"align-self: flex-start;" 在容器顶部对齐所选的弹性项目。</p>

<p>"align-self: flex-end;" 在容器底部对齐所选的弹性项目。</p>

<div class="flex-container">

<div>1</div>

<div style="align-self: flex-start">2</div>

<div style="align-self: flex-end">3</div>

<div>4</div>

</div>

<p>align-self 会覆盖容器的 align-items 属性。</p>

</body>

</html>

## 98.18 使用 Flexbox 的响应式图库

使用 flexbox 创建响应式图像库，该图像库根据屏幕大小在四幅、两幅或全宽图像之间变化：

<!DOCTYPE html>

<html>

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0;

font-family: Arial;

}

.header {

text-align: center;

padding: 32px;

}

.row {

display: flex;

flex-wrap: wrap;

padding: 0 4px;

}

/\* 创建并排的四个等列 \*/

.column {

flex: 25%;

max-width: 25%;

padding: 0 4px;

}

.column img {

margin-top: 8px;

vertical-align: middle;

}

/\* 响应式布局 - 创建两列而不是四列布局 \*/

@media (max-width: 800px) {

.column {

flex: 50%;

max-width: 50%;

}

}

/\* 响应式布局 - 创建上下堆叠而不是并排的两列布局 \*/

@media (max-width: 600px) {

.column {

flex: 100%;

max-width: 100%;

}

}

</style>

<body>

<!-- Header -->

<div class="header">

<h1>响应式图像网格</h1>

<p>请调整浏览器窗口，来查看响应效果。</p>

</div>

<!-- Photo Grid -->

<div class="row">

<div class="column">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

</div>

<div class="column">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

</div>

<div class="column">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

</div>

<div class="column">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

</div>

</div>

</body>

</html>

## 98.19 使用 Flexbox 的响应式网站

使用 flexbox 创建响应式网站，其中包含弹性导航栏和弹性内容：

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 设置 body 元素的样式 \*/

body {

font-family: Arial;

margin: 0;

}

/\* 标题 / LOGO \*/

.header {

padding: 60px;

text-align: center;

background: #1abc9c;

color: white;

}

/\* 设置顶部导航栏样式 \*/

.navbar {

display: flex;

background-color: #333;

}

/\* 设置导航条链接演示 \*/

.navbar a {

color: white;

padding: 14px 20px;

text-decoration: none;

text-align: center;

}

/\* 更改鼠标悬停时的颜色 \*/

.navbar a:hover {

background-color: #ddd;

color: black;

}

/\* 列容器 \*/

.row {

display: flex;

flex-wrap: wrap;

}

/\* 创建并排的非等列 \*/

/\* 侧栏 / 左侧列 \*/

.side {

flex: 30%;

background-color: #f1f1f1;

padding: 20px;

}

/\* 主列 \*/

.main {

flex: 70%;

background-color: white;

padding: 20px;

}

/\* 伪图像，仅供示例 \*/

.fakeimg {

background-color: #aaa;

width: 100%;

padding: 20px;

}

/\* 页脚 \*/

.footer {

padding: 20px;

text-align: center;

background: #ddd;

}

/\* 响应式布局 - 当屏幕小于 700 像素宽时，让两列堆叠而不是并排 \*/

@media screen and (max-width: 700px) {

.row, .navbar {

flex-direction: column;

}

}

</style>

</head>

<body>

<!-- 注释 -->

<div style="background:yellow;padding:5px">

<h4 style="text-align:center">请调整浏览器窗口来查看响应效果。</h4>

</div>

<!-- Header -->

<div class="header">

<h1>我的网站</h1>

<p>拥有 <b>弹性</b> 布局。</p>

</div>

<!-- 导航栏 -->

<div class="navbar">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

</div>

<!-- 弹性网格（内容） -->

<div class="row">

<div class="side">

<h2>关于我</h2>

<h5>我的照片：</h5>

<div class="fakeimg" style="height:200px;">图像</div>

<p>Some text about me in culpa qui officia deserunt mollit anim..</p>

<h3>More Text</h3>

<p>Lorem ipsum dolor sit ame.</p>

<div class="fakeimg" style="height:60px;">图像</div><br>

<div class="fakeimg" style="height:60px;">图像</div><br>

<div class="fakeimg" style="height:60px;">图像</div>

</div>

<div class="main">

<h2>标题</h2>

<h5>标题描述，2021 年 1 月 1 日</h5>

<div class="fakeimg" style="height:200px;">图像</div>

<p>一些文本..</p>

<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>

<br>

<h2>标题</h2>

<h5>标题描述，2021 年 1 月 2 日</h5>

<div class="fakeimg" style="height:200px;">图像</div>

<p>一些文本..</p>

<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>

</div>

</div>

<!-- Footer -->

<div class="footer">

<h2>页脚</h2>

</div>

</body>

</html>

## 98.20 CSS Flexbox 属性

下表列出了与 flexbox 一起使用的 CSS 属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| display | 规定用于 HTML 元素的盒类型。 |
| flex-direction | 规定弹性容器内的弹性项目的方向。 |
| justify-content | 当弹性项目没有用到主轴上的所有可用空间时，水平对齐这些项目。 |
| align-items | 当弹性项目没有用到主轴上的所有可用空间时，垂直对齐这些项。 |
| flex-wrap | 规定弹性项目是否应该换行，若一条 flex 线上没有足够的空间容纳它们。 |
| align-content | 修改 flex-wrap 属性的行为。与 align-items 相似，但它不对齐弹性项目，而是对齐 flex 线。 |
| flex-flow | flex-direction 和 flex-wrap 的简写属性。 |
| order | 规定弹性项目相对于同一容器内其余弹性项目的顺序。 |
| align-self | 用于弹性项目。覆盖容器的 align-items 属性。 |
| flex | flex-grow、flex-shrink 以及 flex-basis 属性的简写属性。 |

# 第99章 CSS 媒体查询

## 99.1 CSS2 引入了媒体类型

CSS2 中引入了 @media 规则，它让为不同媒体类型定义不同样式规则成为可能。

例如：您可能有一组用于计算机屏幕的样式规则、一组用于打印机、一组用于手持设备，甚至还有一组用于电视，等等。

不幸的是，除了打印媒体类型之外，这些媒体类型从未得到过设备的大规模支持。

## 99.2 CSS3 引入了媒体查询

CSS3 中的媒体查询扩展了 CSS2 媒体类型的概念：它们并不查找设备类型，而是关注设备的能力。

媒体查询可用于检查许多事情，例如：

视口的宽度和高度

设备的宽度和高度

方向（平板电脑/手机处于横向还是纵向模式）

分辨率

使用媒体查询是一种流行的技术，可以向台式机、笔记本电脑、平板电脑和手机（例如 iPhone 和 Android 手机）提供定制的样式表。

## 99.3 媒体查询语法

媒体查询由一种媒体类型组成，并可包含一个或多个表达式，这些表达式可以解析为 true 或 false。

@media not|only mediatype and (expressions) {

CSS-Code;

}

如果指定的媒体类型与正在显示文档的设备类型匹配，并且媒体查询中的所有表达式均为 true，则查询结果为 true。当媒体查询为 true 时，将应用相应的样式表或样式规则，并遵循正常的级联规则。

除非您使用 not 或 only 运算符，否则媒体类型是可选的，且隐含 all 类型。

您还可以针对不同的媒体使用不同的样式表：

<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">

## 99.4 CSS3 媒体类型

|  |  |
| --- | --- |
| 值 | 描述 |
| all | 用于所有媒体类型设备。 |
| print | 用于打印机。 |
| screen | 用于计算机屏幕、平板电脑、智能手机等等。 |
| speech | 用于大声“读出”页面的屏幕阅读器。 |

## 99.5 媒体查询的简单实例

使用媒体查询的一种方法是在样式表内有一个备用的 CSS 部分。

下面的例子在视口宽度为 480 像素或更宽时将背景颜色更改为浅绿色（如果视口小于 480 像素，则背景颜色会是粉色）：

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: pink;

}

@media screen and (min-width: 480px) {

body {

background-color: lightgreen;

}

}

</style>

</head>

<body>

<h1>请调整浏览器窗口大小来查看效果！</h1>

<p>此媒体查询值应用于：媒体类型为 screen，且视口为 480px 或更宽。</p>

</body>

</html>

下例显示了一个菜单，如果视口的宽度为 480 像素或更宽，则该菜单会浮动到页面的左侧（如果视口小于 480 像素，则该菜单将位于内容的顶部）：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

.wrapper {overflow: auto;}

#main {margin-left: 4px;}

#leftsidebar {

float: none;

width: auto;

}

#menulist {

margin: 0;

padding: 0;

}

.menuitem {

background: #CDF0F6;

border: 1px solid #d4d4d4;

border-radius: 4px;

list-style-type: none;

margin: 4px;

padding: 2px;

}

@media screen and (min-width: 480px) {

#leftsidebar {width: 200px; float: left;}

#main {margin-left: 216px;}

}

</style>

</head>

<body>

<div class="wrapper">

<div id="leftsidebar">

<ul id="menulist">

<li class="menuitem">Menu-item 1</li>

<li class="menuitem">Menu-item 2</li>

<li class="menuitem">Menu-item 3</li>

<li class="menuitem">Menu-item 4</li>

<li class="menuitem">Menu-item 5</li>

</ul>

</div>

<div id="main">

<h1>请调整浏览器窗口大小来查看效果！</h1>

<p>本例显示了一个菜单，如果视口为 480 像素或更宽，它将向页面左侧浮动。如果视口小于 480 像素，则菜单将位于内容的顶部。</p>

</div>

</div>

</body>

</html>

# 第100章 CSS 媒体查询 - 实例

## 100.1 CSS 媒体查询 - 更多实例

让我们看看使用媒体查询的更多例子。

媒体查询是一种流行的技术，用于将定制的样式表传递给不同的设备。

下面演示一个简单的例子，让我们来更改不同设备的背景色：

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: tan;

color: black;

}

/\* 在宽度为 992px 或更小的屏幕上，背景颜色为蓝色 \*/

@media screen and (max-width: 992px) {

body {

background-color: blue;

color: white;

}

}

/\* 在宽度为 600px 或更小的屏幕上，背景色为橄榄色 \*/

@media screen and (max-width: 600px) {

body {

background-color: olive;

color: white;

}

}

</style>

</head>

<body>

<h1>请调整浏览器窗口大小以查看效果！</h1>

<p>默认情况下，文档的背景色为棕褐色。如果屏幕尺寸为 992px 或更小，则颜色将变为蓝色。如果小于或等于 600px，它将变为橄榄色。</p>

</body>

</html>

## 100.2 菜单的媒体查询

在本例中，我们使用媒体查询来创建响应式导航菜单，该菜单在不同的屏幕尺寸上会有所不同。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 设置顶部导航栏样式 \*/

.topnav {

overflow: hidden;

background-color: #333;

}

/\* 设置 topnav 链接的样式 \*/

.topnav a {

float: left;

display: block;

color: #f2f2f2;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

/\* 悬停时改变颜色 \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

/\* 在宽度为 600 像素或更小的屏幕上，使菜单链接彼此堆叠，而不是并排 \*/

@media screen and (max-width: 600px) {

.topnav a {

float: none;

width: 100%;

}

}

</style>

</head>

<body>

<h1>响应式导航菜单</h1>

<p>请调整浏览器窗口的大小以查看效果：当屏幕小于 600 像素时，导航菜单将垂直显示，而不是水平显示。</p>

<div class="topnav">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

</div>

</body>

</html>

## 100.3 列的媒体查询

媒体查询的常见用法是创建弹性布局。在本例中，我们创建了一个布局，该布局在四列、两列和全宽列之间变化，具体取决于不同的屏幕尺寸：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 创建彼此相邻浮动的四个等列 \*/

.column {

float: left;

width: 25%;

padding: 20px;

}

/\* 清除列后的浮动 \*/

.row:after {

content: "";

display: table;

clear: both;

}

/\* 在 992px 或更小的屏幕上，从四列变为两列 \*/

@media screen and (max-width: 992px) {

.column {

width: 50%;

}

}

/\* 在宽度小于或等于 600px 的屏幕上，使列堆叠在一起，而不是彼此相邻 \*/

@media screen and (max-width: 600px) {

.column {

width: 100%;

}

}

</style>

</head>

<body>

<h1>响应式四列布局</h1>

<p><b>请调整浏览器窗口的大小以查看响应效果。</b>在宽度小于或等于 992px 的屏幕上，列的大小将从四列调整为两列。在宽度为 600px 或更小的屏幕上，这些列将堆叠在一起，而不是彼此相邻。</p>

<div class="row">

<div class="column" style="background-color:#aaa;">

<h2>Column 1</h2>

<p>Some text..</p>

</div>

<div class="column" style="background-color:#bbb;">

<h2>Column 2</h2>

<p>Some text..</p>

</div>

<div class="column" style="background-color:#ccc;">

<h2>Column 3</h2>

<p>Some text..</p>

</div>

<div class="column" style="background-color:#ddd;">

<h2>Column 4</h2>

<p>Some text..</p>

</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 弹性框的容器 \*/

.row {

display: flex;

flex-wrap: wrap;

}

/\* 创建四个等列 \*/

.column {

flex: 25%;

padding: 20px;

}

/\* 在 992px 或更小的屏幕上，从四列变为两列 \*/

@media screen and (max-width: 992px) {

.column {

flex: 50%;

}

}

/\* 在宽度小于或等于 60px 的屏幕上，使列堆叠在一起，而不是彼此相邻 \*/

@media screen and (max-width: 600px) {

.row {

flex-direction: column;

}

}

</style>

</head>

<body>

<h1>响应式四列弹性布局</h1>

<p><b>请调整浏览器窗口的大小以查看响应效果。</b>在宽度小于或等于 992px 的屏幕上，列的大小将从四列调整为两列。在宽度为 600px 或更小的屏幕上，这些列将堆叠在一起，而不是彼此相邻。</p>

<div class="row">

<div class="column" style="background-color:#aaa;">

<h2>Column 1</h2>

<p>Some text..</p>

</div>

<div class="column" style="background-color:#bbb;">

<h2>Column 2</h2>

<p>Some text..</p>

</div>

<div class="column" style="background-color:#ccc;">

<h2>Column 3</h2>

<p>Some text..</p>

</div>

<div class="column" style="background-color:#ddd;">

<h2>Column 4</h2>

<p>Some text..</p>

</div>

</div>

</body>

</html>

## 100.4 用媒体查询隐藏元素

媒体查询的另一种常见用法是在不同屏幕尺寸上隐藏元素：

在小屏幕上我会隐藏。

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

div.example {

background-color: yellow;

padding: 20px;

}

@media screen and (max-width: 600px) {

div.example {

display: none;

}

}

</style>

</head>

<body>

<h1>隐藏不同屏幕尺寸的元素</h1>

<div class="example">Example DIV.</div>

<p>当浏览器的宽度为 600 像素或更小时，隐藏 div 元素。请调整浏览器窗口的大小以查看效果。</p>

</body>

</html>

## 100.5 用媒体查询改变字体

您还可以使用媒体查询来更改不同屏幕尺寸上的元素的字体大小：

可变的字体大小。

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

div.example {

background-color: lightgrey;

padding: 20px;

}

@media screen and (min-width: 600px) {

div.example {

font-size: 80px;

}

}

@media screen and (max-width: 600px) {

div.example {

font-size: 30px;

}

}

</style>

</head>

<body>

<h1>在不同的屏幕尺寸上更改元素的字体大小</h1>

<div class="example">Example DIV.</div>

<p>当浏览器的宽度为 600 像素或更小时，将 DIV 的字体大小设置为 30px。当它是 601 像素或更宽时，将字体大小设置为 80 像素。请调整浏览器窗口的大小以查看效果。</p>

</body>

</html>

## 100.6 弹性图片库

在此例中，我们将媒体查询与 flexbox 一起使用来创建响应式图片库：

实例

<!DOCTYPE html>

<html>

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0;

font-family: Arial;

}

.header {

text-align: center;

padding: 32px;

}

.row {

display: flex;

flex-wrap: wrap;

padding: 0 4px;

}

/\* 创建彼此相邻的四个相等的列 \*/

.column {

flex: 25%;

max-width: 25%;

padding: 0 4px;

}

.column img {

margin-top: 8px;

vertical-align: middle;

}

/\* 响应式布局 - 制作两列而不是四列的布局 \*/

@media screen and (max-width: 800px) {

.column {

flex: 50%;

max-width: 50%;

}

}

/\* 响应式布局 - 使两列相互堆叠而不是彼此并排 \*/

@media screen and (max-width: 600px) {

.column {

flex: 100%;

max-width: 100%;

}

}

</style>

<body>

<!-- Header -->

<div class="header">

<h1>响应式图像网格</h1>

<p>请调整浏览器窗口的大小以查看响应效果。</p>

</div>

<!-- Photo Grid -->

<div class="row">

<div class="column">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

</div>

<div class="column">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

</div>

<div class="column">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

</div>

<div class="column">

<img src="/i/photo/tulip-yellow-2.jpg" style="width:100%">

<img src="/i/photo/tulip.jpg" style="width:100%">

<img src="/i/photo/flower-3.jpg" style="width:100%">

<img src="/i/photo/flower-1.jpg" style="width:100%">

<img src="/i/photo/flower-4.jpg" style="width:100%">

<img src="/i/photo/tulip-red.jpg" style="width:100%">

<img src="/i/photo/tulip-yellow.jpg" style="width:100%">

<img src="/i/photo/flower-2.jpg" style="width:100%">

</div>

</div>

</body>

</html>

## 100.7 弹性网站

在本例中，我们将媒体查询与 flexbox 一起使用，以创建响应式网站，其中包含弹性导航栏和弹性内容。

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 设置 body 的样式 \*/

body {

font-family: Arial;

margin: 0;

}

/\* 页眉/LOGO \*/

.header {

padding: 60px;

text-align: center;

background: #1abc9c;

color: white;

}

/\* 设置顶部导航栏的样式 \*/

.navbar {

display: flex;

background-color: #333;

}

/\* 设置导航栏链接的样式 \*/

.navbar a {

color: white;

padding: 14px 20px;

text-decoration: none;

text-align: center;

}

/\* 悬停时改变颜色 \*/

.navbar a:hover {

background-color: #ddd;

color: black;

}

/\* 列容器 \*/

.row {

display: flex;

flex-wrap: wrap;

}

/\* 创建两个彼此相邻的不相等的列 \*/

/\* 侧栏/左侧列 \*/

.side {

flex: 30%;

background-color: #f1f1f1;

padding: 20px;

}

/\* 主列 \*/

.main {

flex: 70%;

background-color: white;

padding: 20px;

}

/\* 伪图像，仅用于此例 \*/

.fakeimg {

background-color: #aaa;

width: 100%;

padding: 20px;

}

/\* 页脚 \*/

.footer {

padding: 20px;

text-align: center;

background: #ddd;

}

/\* 响应式布局 - 当屏幕宽度小于 700 像素时，使两列相互堆叠，而不是彼此相邻 \*/

@media screen and (max-width: 700px) {

.row, .navbar {

flex-direction: column;

}

}

</style>

</head>

<body>

<!-- Note -->

<div style="background:yellow;padding:5px">

<h4 style="text-align:center">请调整浏览器窗口的大小以查看响应效果。</h4>

</div>

<!-- Header -->

<div class="header">

<h1>My Website</h1>

<p>With a <b>flexible</b> layout.</p>

</div>

<!-- Navigation Bar -->

<div class="navbar">

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

<a href="#">Link</a>

</div>

<!-- The flexible grid (content) -->

<div class="row">

<div class="side">

<h2>About Me</h2>

<h5>Photo of me:</h5>

<div class="fakeimg" style="height:200px;">Image</div>

<p>Some text about me in culpa qui officia deserunt mollit anim..</p>

<h3>More Text</h3>

<p>Lorem ipsum dolor sit ame.</p>

<div class="fakeimg" style="height:60px;">Image</div><br>

<div class="fakeimg" style="height:60px;">Image</div><br>

<div class="fakeimg" style="height:60px;">Image</div>

</div>

<div class="main">

<h2>TITLE HEADING</h2>

<h5>Title description, Dec 7, 2017</h5>

<div class="fakeimg" style="height:200px;">Image</div>

<p>Some text..</p>

<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>

<br>

<h2>TITLE HEADING</h2>

<h5>Title description, Sep 2, 2017</h5>

<div class="fakeimg" style="height:200px;">Image</div>

<p>Some text..</p>

<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>

</div>

</div>

<!-- Footer -->

<div class="footer">

<h2>Footer</h2>

</div>

</body>

</html>

## 100.8 方向：人像 / 风景

媒体查询还可以用于根据浏览器的方向更改页面的布局。

您可以设置一组 CSS 属性，这些属性仅在浏览器窗口的宽度大于其高度时才适用，即所谓的横屏：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

background-color: lightgreen;

}

@media only screen and (orientation: landscape) {

body {

background-color: lightblue;

}

}

</style>

</head>

<body>

<p>请调整浏览器窗口的大小。如果此文档的宽度大于高度，背景色为“浅蓝色”，否则为“浅绿色”。</p>

</body>

</html>

## 100.9 最小宽度到最大宽度

您还可以使用 max-width 和 min-width 属性设置最小宽度和最大宽度。

例如，当浏览器的宽度在 600 到 900 像素之间时，更改 <div> 元素的外观：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

@media screen and (max-width: 900px) and (min-width: 600px) {

div.example {

font-size: 50px;

padding: 50px;

border: 8px solid black;

background: yellow;

}

}

</style>

</head>

<body>

<h1>在不同的屏幕尺寸上更改 DIV 的外观</h1>

<div class="example">Example DIV.</div>

<p>当浏览器的宽度在 600 到 900 像素之间时，更改 DIV 的外观。<b>请调整浏览器窗口大小以查看效果。</b></p>

</body>

</html>

使用附加值：在下面的例子中，我们使用逗号（类似 OR 运算符）将附加的媒体查询添加到已有媒体查询中：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

@media screen and (max-width: 900px) and (min-width: 600px), (min-width: 1100px) {

div.example{

font-size: 50px;

padding: 50px;

border: 8px solid black;

background: yellow;

}

}

</style>

</head>

<body>

<h1>在不同的屏幕尺寸上更改 DIV 的外观</h1>

<div class="example">Example DIV.</div>

<p>当浏览器的宽度介于 600 和 900 像素之间或大于 1100 像素时，请更改 DIV 的外观。<b>请调整浏览器窗口大小以查看效果。</b></p>

</body>

</html>

## 100.10 CSS @media 参考手册

有关所有媒体类型和特性/表达式的完整概述，请查看 CSS 参考中的 @media 规则。

提示：如需学习有关响应式 Web 设计（如何针对不同的设备和屏幕）的更多知识，以及使用媒体查询断点，请阅读我们的 响应式 Web 设计教程。

# 第101章 响应式网页设计 - 简介

## 101.1 什么是响应式网页设计？

响应式 web 设计会让您的网页在所有设备上看起来都不错。

响应式 web 设计仅使用 HTML 和 CSS。

响应式 web 设计并不是程序或 JavaScript。

## 101.2 为所有用户获得最佳体验的设计

可以使用许多不同的设备来查看网页：台式机、平板电脑和手机。无论使用哪种设备，您的网页都应该看起来美观且易用。

网页不应舍弃信息来适合较小的设备，而应使其内容适合任何设备：

如果您使用 CSS 和 HTML 调整大小、隐藏、缩小、放大或移动内容，以使其在任何屏幕上看起来都很好，则称为响应式 Web 设计。

如果您不理解下面的例子，请不要担心，我们将在下一章中一步一步地分解代码：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

.row::after {

content: "";

clear: both;

display: table;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

html {

font-family: "Lucida Sans", sans-serif;

}

.header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.menu li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.menu li:hover {

background-color: #0099cc;

}

.aside {

background-color: #33b5e5;

padding: 15px;

color: #ffffff;

text-align: center;

font-size: 14px;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.footer {

background-color: #0099cc;

color: #ffffff;

text-align: center;

font-size: 12px;

padding: 15px;

}

/\* For mobile phones: \*/

[class\*="col-"] {

width: 100%;

}

@media only screen and (min-width: 600px) {

/\* 针对平板电脑： \*/

.col-s-1 {width: 8.33%;}

.col-s-2 {width: 16.66%;}

.col-s-3 {width: 25%;}

.col-s-4 {width: 33.33%;}

.col-s-5 {width: 41.66%;}

.col-s-6 {width: 50%;}

.col-s-7 {width: 58.33%;}

.col-s-8 {width: 66.66%;}

.col-s-9 {width: 75%;}

.col-s-10 {width: 83.33%;}

.col-s-11 {width: 91.66%;}

.col-s-12 {width: 100%;}

}

@media only screen and (min-width: 768px) {

/\* 针对桌面： \*/

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

}

</style>

</head>

<body>

<div class="header">

<h1>Shanghai</h1>

</div>

<div class="row">

<div class="col-3 col-s-3 menu">

<ul>

<li>交通</li>

<li>文化</li>

<li>旅游</li>

<li>美食</li>

</ul>

</div>

<div class="col-6 col-s-9">

<h1>欢迎来到上海</h1>

<p>上海市，简称沪，别称申，是中华人民共和国直辖市，中国的经济、金融、贸易和航运中心，世界著名的港口城市，是中国人口第二多的城市。</p>

</div>

<div class="col-3 col-s-12">

<div class="aside">

<h2>历史</h2>

<p>最晚在新石器时代，上海地区已经有先民聚居。春秋时代，上海由吴国管辖，战国时代则是楚国领土 ...</p>

<h2>位置</h2>

<p>上海位于中国东部弧形海岸线的正中间，长江三角洲最东部，东临东海，南濒杭州湾，西与江苏、浙江两省相接 ...</p>

<h2>环境</h2>

<p>上海地处江南水乡，并位于长江入海口，亦不处于主要地震带上，因此如地震、洪水以及地质类灾害鲜有发生 ...</p>

</div>

</div>

</div>

<div class="footer">

<p>请调整浏览器窗口的大小，以查看内容如何响应调整大小。</p>

</div>

</body>

</html>

# 第102章 响应式网页设计 - 视口

## 102.1 什么是视口？

视口（viewport）是用户在网页上的可见区域。

视口随设备而异，在移动电话上会比在计算机屏幕上更小。

在平板电脑和手机之前，网页仅设计为用于计算机屏幕，并且网页拥有静态设计和固定大小是很常见的。

然后，当我们开始使用平板电脑和手机上网时，固定大小的网页太大了，无法适应视口。为了解决这个问题，这些设备上的浏览器会按比例缩小整个网页以适合屏幕大小。

这并不是完美的！勉强是一种快速的修正。

## 102.2 设置视口

HTML5 引入了一种方法，使 Web 设计者可以通过 <meta> 标签来控制视口。

您应该在所有网页中包含以下 <meta> 视口元素：

<meta name="viewport" content="width=device-width, initial-scale=1.0">

它为浏览器提供了关于如何控制页面尺寸和缩放比例的指令。

width=device-width 部分将页面的宽度设置为跟随设备的屏幕宽度（视设备而定）。

当浏览器首次加载页面时，initial-scale=1.0 部分设置初始缩放级别。

下面分别是不带视口 meta 标签的网页、以及带视口 meta 标签的网页的例子：

提示：如果您使用手机或平板电脑浏览这张页面，则可以单击下面的两个链接以查看不同之处。

## 102.3 把内容调整到视口的大小

用户习惯在台式机和移动设备上垂直滚动网站，而不是水平滚动！

因此，如果迫使用户水平滚动或缩小以查看整个网页，则会导致不佳的用户体验。

还需要遵循的一些附加规则：

1. 请勿使用较大的固定宽度元素 - 例如，如果图像的宽度大于视口的宽度，则可能导致视口水平滚动。请务必调整此内容以适合视口的宽度。

2. 不要让内容依赖于特定的视口宽度来呈现好的效果 - 由于以 CSS 像素计的屏幕尺寸和宽度在设备之间变化很大，因此内容不应依赖于特定的视口宽度来呈现良好的效果。

3. 使用 CSS 媒体查询为小屏幕和大屏幕应用不同的样式 - 为页面元素设置较大的 CSS 绝对宽度将导致该元素对于较小设备上的视口太宽。而是应该考虑使用相对宽度值，例如 width: 100%。另外，要小心使用较大的绝对定位值，这可能会导致元素滑落到小型设备的视口之外。

# 第103章 响应式网页设计 - 网格视图

## 103.1 什么是网格视图？

许多网页都基于网格视图（grid-view），这意味着页面被分割为几列：

在设计网页时，使用网格视图非常有帮助。这样可以更轻松地在页面上放置元素。

响应式网格视图通常有 12 列，总宽度为 100％，并且在调整浏览器窗口大小时会收缩和伸展。

## 103.2 构建响应式网格视图

让我们开始构建响应式网格视图。

首先，确保所有 HTML 元素的 box-sizing 属性设置为 border-box。这样可以确保元素的总宽度和高度中包括内边距（填充）和边框。

请在 CSS 中添加如下代码：

\* {

box-sizing: border-box;

}

请在我们的 CSS Box Sizing 一章中阅读有关 box-sizing 属性的更多内容。

下面的例子展示了一张简单的响应式网页，其中包含两列：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

.header {

border: 1px solid red;

padding: 15px;

}

.menu {

width: 25%;

float: left;

padding: 15px;

border: 1px solid red;

}

.main {

width: 75%;

float: left;

padding: 15px;

border: 1px solid red;

}

</style>

</head>

<body>

<div class="header">

<h1>Shanghai</h1>

</div>

<div class="menu">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</div>

<div class="main">

<h1>The City</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!

</p>

<p>请调整浏览器窗口的大小，来查看内容如何响应调整大小。</p>

</div>

</body>

</html>

若网页只包含两列，则上面的例子还不错。

但是，我们希望使用拥有 12 列的响应式网格视图，来更好地控制网页。

首先，我们必须计算一列的百分比：100% / 12 列 = 8.33%。

然后，我们为 12 列中的每一列创建一个类，即 class="col-" 和一个数字，该数字定义此节应跨越的列数：

CSS:

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

.header {

border: 1px solid red;

padding: 15px;

}

.row::after {

content: "";

clear: both;

display: table;

}

[class\*="col-"] {

float: left;

padding: 15px;

border: 1px solid red;

}

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

</style>

</head>

<body>

<div class="header">

<h1>Shanghai</h1>

</div>

<div class="row">

<div class="col-3">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</div>

<div class="col-9">

<h1>The City</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!

</p>

<p>请调整浏览器窗口的大小，以查看内容如何响应调整大小。</p>

</div>

</div>

</body>

</html>

所有这些列应向左浮动，并带有 15px 的内边距：

CSS:

[class\*="col-"] {

float: left;

padding: 15px;

border: 1px solid red;

}

每行都应被包围在 <div> 中。行内的列数总应总计为 12：

HTML:

<div class="row">

<div class="col-3">...</div> <!-- 25% -->

<div class="col-9">...</div> <!-- 75% -->

</div>

行内的所有列全部都向左浮动，因此会从页面流中移出，并将放置其他元素，就好像这些列不存在一样。为了防止这种情况，我们会添加清除流的样式：

CSS:

.row::after {

content: "";

clear: both;

display: table;

}

我们还想添加一些样式和颜色，使其看起来更美观：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

.row::after {

content: "";

clear: both;

display: table;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

html {

font-family: "Lucida Sans", sans-serif;

}

.header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.menu li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.menu li:hover {

background-color: #0099cc;

}

</style>

</head>

<body>

<div class="header">

<h1>Shanghai</h1>

</div>

<div class="row">

<div class="col-3 menu">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</div>

<div class="col-9">

<h1>The City</h1>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!

</p>

<p>请调整浏览器窗口的大小，来查看内容如何响应调整大小。</p>

</div>

</div>

</body>

</html>

# 第104章 响应式网页设计 - 媒体查询

## 104.1 什么是媒体查询？

媒体查询是 CSS3 中引入的一种 CSS 技术。

仅在满足特定条件时，它才会使用 @media 规则来引用 CSS 属性块。

实例

如果浏览器窗口是 600px 或更小，则背景颜色为浅蓝色：

@media only screen and (max-width: 600px) {

body {

background-color: lightblue;

}

}

## 104.2 添加断点

在本教程中稍早前，我们制作了一张包含行和列的网页，但是这张响应式网页在小屏幕上看起来效果并不好。

媒体查询可以帮助您。我们可以添加一个断点，其中设计的某些部分在断点的每一侧会表现得有所不同。

使用媒体查询在 768px 处添加断点：

实例

当屏幕（浏览器窗口）小于 768px 时，每列的宽度应为 100％：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

.row::after {

content: "";

clear: both;

display: block;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

html {

font-family: "Lucida Sans", sans-serif;

}

.header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.menu li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.menu li:hover {

background-color: #0099cc;

}

.aside {

background-color: #33b5e5;

padding: 15px;

color: #ffffff;

text-align: center;

font-size: 14px;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.footer {

background-color: #0099cc;

color: #ffffff;

text-align: center;

font-size: 12px;

padding: 15px;

}

/\* 针对桌面： \*/

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {

/\* 针对手机： \*/

[class\*="col-"] {

width: 100%;

}

}

</style>

</head>

<body>

<div class="header">

<h1>Shanghai</h1>

</div>

<div class="row">

<div class="col-3 menu">

<ul>

<li>交通</li>

<li>文化</li>

<li>旅游</li>

<li>美食</li>

</ul>

</div>

<div class="col-6">

<h1>欢迎来到上海</h1>

<p>上海市，简称沪，别称申，是中华人民共和国直辖市，中国的经济、金融、贸易和航运中心，世界著名的港口城市，是中国人口第二多的城市。</p>

</div>

<div class="col-3 right">

<div class="aside">

<h2>历史</h2>

<p>最晚在新石器时代，上海地区已经有先民聚居。春秋时代，上海由吴国管辖，战国时代则是楚国领土 ...</p>

<h2>位置</h2>

<p>上海位于中国东部弧形海岸线的正中间，长江三角洲最东部，东临东海，南濒杭州湾，西与江苏、浙江两省相接 ...</p>

<h2>环境</h2>

<p>上海地处江南水乡，并位于长江入海口，亦不处于主要地震带上，因此如地震、洪水以及地质类灾害鲜有发生 ...</p>

</div>

</div>

</div>

<div class="footer">

<p>请调整浏览器窗口的大小，以查看内容如何响应调整大小。</p>

</div>

</body>

</html>

## 104.3 始终移动优先设计

移动优先（Mobile First）指的是在对台式机或任何其他设备进行设计之前，优先针对移动设备进行设计（这将使页面在较小的设备上显示得更快）。

这意味着我们必须在 CSS 中做一些改进。

当宽度小于 768px 时，我们应该修改设计，而不是更改宽度。我们就这样进行了“移动优先”的设计：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

.row::after {

content: "";

clear: both;

display: table;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

html {

font-family: "Lucida Sans", sans-serif;

}

.header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.menu li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.menu li:hover {

background-color: #0099cc;

}

.aside {

background-color: #33b5e5;

padding: 15px;

color: #ffffff;

text-align: center;

font-size: 14px;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.footer {

background-color: #0099cc;

color: #ffffff;

text-align: center;

font-size: 12px;

padding: 15px;

}

/\* 针对手机： \*/

[class\*="col-"] {

width: 100%;

}

@media only screen and (min-width: 768px) {

/\* For desktop: \*/

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

}

</style>

</head>

<body>

<div class="header">

<h1>上海</h1>

</div>

<div class="row">

<div class="col-3 menu">

<ul>

<li>交通</li>

<li>文化</li>

<li>旅游</li>

<li>美食</li>

</ul>

</div>

<div class="col-6">

<h1>欢迎来到上海</h1>

<p>上海市，简称沪，别称申，是中华人民共和国直辖市，中国的经济、金融、贸易和航运中心，世界著名的港口城市，是中国人口第二多的城市。</p>

</div>

<div class="col-3 right">

<div class="aside">

<h2>历史</h2>

<p>最晚在新石器时代，上海地区已经有先民聚居。春秋时代，上海由吴国管辖，战国时代则是楚国领土 ...</p>

<h2>位置</h2>

<p>上海位于中国东部弧形海岸线的正中间，长江三角洲最东部，东临东海，南濒杭州湾，西与江苏、浙江两省相接 ...</p>

<h2>环境</h2>

<p>上海地处江南水乡，并位于长江入海口，亦不处于主要地震带上，因此如地震、洪水以及地质类灾害鲜有发生 ...</p>

</div>

</div>

</div>

<div class="footer">

<p>请调整浏览器窗口的大小，来查看内容如何响应调整大小。</p>

</div>

</body>

</html>

## 104.4 另一个断点

您可以添加任意多个断点。

我们还会在平板电脑和手机之间插入一个断点。

为此，我们添加了一个媒体查询（在 600 像素），并为大于 600 像素（但小于 768 像素）的设备添加了一组新类：

实例

请注意，两组类几乎相同，唯一的区别是名称（col- 和 col-s-）：

/\* 针对手机： \*/

[class\*="col-"] {

width: 100%;

}

@media only screen and (min-width: 600px) {

/\* 针对平板电脑： \*/

.col-s-1 {width: 8.33%;}

.col-s-2 {width: 16.66%;}

.col-s-3 {width: 25%;}

.col-s-4 {width: 33.33%;}

.col-s-5 {width: 41.66%;}

.col-s-6 {width: 50%;}

.col-s-7 {width: 58.33%;}

.col-s-8 {width: 66.66%;}

.col-s-9 {width: 75%;}

.col-s-10 {width: 83.33%;}

.col-s-11 {width: 91.66%;}

.col-s-12 {width: 100%;}

}

@media only screen and (min-width: 768px) {

/\* 针对桌面： \*/

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

}

有两组相同的类似乎很奇怪，但是它给了我们机会用 HTML 来决定在每个断点处的列会发生什么：

HTML 实例

对于台式机：

第一和第三部分都会跨越 3 列。中间部分将跨越 6 列。

对于平板电脑：

第一部分将跨越 3 列，第二部分将跨越 9 列，第三部分将显示在前两部分的下方，并将跨越 12 列：

<div class="row">

<div class="col-3 col-s-3">...</div>

<div class="col-6 col-s-9">...</div>

<div class="col-3 col-s-12">...</div>

</div>

详细代码：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

.row::after {

content: "";

clear: both;

display: table;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

html {

font-family: "Lucida Sans", sans-serif;

}

.header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.menu li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.menu li:hover {

background-color: #0099cc;

}

.aside {

background-color: #33b5e5;

padding: 15px;

color: #ffffff;

text-align: center;

font-size: 14px;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.footer {

background-color: #0099cc;

color: #ffffff;

text-align: center;

font-size: 12px;

padding: 15px;

}

/\* For mobile phones: \*/

[class\*="col-"] {

width: 100%;

}

@media only screen and (min-width: 600px) {

/\* 针对平板电脑： \*/

.col-s-1 {width: 8.33%;}

.col-s-2 {width: 16.66%;}

.col-s-3 {width: 25%;}

.col-s-4 {width: 33.33%;}

.col-s-5 {width: 41.66%;}

.col-s-6 {width: 50%;}

.col-s-7 {width: 58.33%;}

.col-s-8 {width: 66.66%;}

.col-s-9 {width: 75%;}

.col-s-10 {width: 83.33%;}

.col-s-11 {width: 91.66%;}

.col-s-12 {width: 100%;}

}

@media only screen and (min-width: 768px) {

/\* 针对桌面： \*/

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

}

</style>

</head>

<body>

<div class="header">

<h1>Shanghai</h1>

</div>

<div class="row">

<div class="col-3 col-s-3 menu">

<ul>

<li>交通</li>

<li>文化</li>

<li>旅游</li>

<li>美食</li>

</ul>

</div>

<div class="col-6 col-s-9">

<h1>欢迎来到上海</h1>

<p>上海市，简称沪，别称申，是中华人民共和国直辖市，中国的经济、金融、贸易和航运中心，世界著名的港口城市，是中国人口第二多的城市。</p>

</div>

<div class="col-3 col-s-12">

<div class="aside">

<h2>历史</h2>

<p>最晚在新石器时代，上海地区已经有先民聚居。春秋时代，上海由吴国管辖，战国时代则是楚国领土 ...</p>

<h2>位置</h2>

<p>上海位于中国东部弧形海岸线的正中间，长江三角洲最东部，东临东海，南濒杭州湾，西与江苏、浙江两省相接 ...</p>

<h2>环境</h2>

<p>上海地处江南水乡，并位于长江入海口，亦不处于主要地震带上，因此如地震、洪水以及地质类灾害鲜有发生 ...</p>

</div>

</div>

</div>

<div class="footer">

<p>请调整浏览器窗口的大小，以查看内容如何响应调整大小。</p>

</div>

</body>

</html>

## 104.5典型的设备断点

高度和宽度不同的屏幕和设备不计其数，因此很难为每个设备创建精确的断点。为了简单起见，您可以瞄准这五组：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

.example {

padding: 20px;

color: white;

}

/\* 超小型设备（600px及以下的手机） \*/

@media only screen and (max-width: 600px) {

.example {background: red;}

}

/\* 小型设备（平板电脑竖屏模式和大屏手机，600 像素及以上） \*/

@media only screen and (min-width: 600px) {

.example {background: green;}

}

/\* 中型设备（平板电脑横屏模式，768 像素及以上） \*/

@media only screen and (min-width: 768px) {

.example {background: blue;}

}

/\* 大型设备（笔电、台式机，992 像素及以上） \*/

@media only screen and (min-width: 992px) {

.example {background: orange;}

}

/\* 超大型设备（大屏笔电、台式机，1200 像素及以上） \*/

@media only screen and (min-width: 1200px) {

.example {background: pink;}

}

</style>

</head>

<body>

<h1>典型的媒体查询断点</h1>

<p class="example">请调整浏览器窗口的大小，来查看该段落的背景色在不同屏幕尺寸下如何变化。</p>

</body>

</html>

## 104.6 方向：人像 / 风景

媒体查询还可用于根据浏览器的方向来更改页面的布局。

您可以设置一组 CSS 属性，这些属性仅在浏览器窗口的宽度大于其高度时才适用，即所谓的“横屏”方向：

实例

如果方向为横向模式（landscape mode），则网页背景为浅蓝色：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

background-color: lightgreen;

}

@media only screen and (orientation: landscape) {

body {

background-color: lightblue;

}

}

</style>

</head>

<body>

<p>请调整浏览器窗口的大小。如果此文档的宽度大于高度，背景色为“浅蓝色”，否则为“浅绿色”。</p>

</body>

</html>

## 104.7 用媒体查询隐藏元素

媒体查询的另一种常见用法是在不同屏幕尺寸上对元素进行隐藏：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

div.example {

background-color: yellow;

padding: 20px;

}

@media screen and (max-width: 600px) {

div.example {

display: none;

}

}

</style>

</head>

<body>

<h1>隐藏不同屏幕尺寸的元素</h1>

<div class="example">Example DIV.</div>

<p>当浏览器的宽度为 600 像素或更小时，隐藏 div 元素。请调整浏览器窗口的大小以查看效果。</p>

</body>

</html>

## 104.8 用媒体查询修改字体

您还可以使用媒体查询来更改不同屏幕尺寸上的元素的字体大小：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

div.example {

background-color: lightgrey;

padding: 20px;

}

@media screen and (min-width: 600px) {

div.example {

font-size: 80px;

}

}

@media screen and (max-width: 600px) {

div.example {

font-size: 30px;

}

}

</style>

</head>

<body>

<h1>在不同的屏幕尺寸上更改元素的字体大小</h1>

<div class="example">Example DIV.</div>

<p>当浏览器的宽度为 600 像素或更小时，将 DIV 的字体大小设置为 30px。当它是 601 像素或更宽时，将字体大小设置为 80 像素。请调整浏览器窗口的大小以查看效果。</p>

</body>

</html>

## 104.9 CSS @media 参考手册

# 第105章 响应式网页设计 - 图像

## 105.1 使用 width 属性

如果 width 属性设置为百分比，且高度设置为 "auto"，则图像将进行响应来放大或缩小：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

img {

width: 100%;

height: auto;

}

</style>

</head>

<body>

<img src="/i/photo/shanghai.jpg" width="460" height="306">

<p>调整浏览器窗口的大小，来查看图像如何缩放。</p>

</body>

</html>

## 105.2 使用 max-width 属性

如果将 max-width 属性设置为 100％，则图像将按需缩小，但绝不会放大到大于其原始大小

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

img {

max-width: 100%;

height: auto;

}

</style>

</head>

<body>

<img src="/i/photo/shanghai.jpg" width="460" height="306">

<p>请调整浏览器窗口的大小，来查看宽度小于 460 像素时图像如何缩放。</p>

</body>

</html>

## 105.3 向实例网页添加图像

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

img {

width: 100%;

height: auto;

}

.row:after {

content: "";

clear: both;

display: table;

}

[class\*="col-"] {

float: left;

padding: 15px;

width: 100%;

}

@media only screen and (min-width: 600px) {

.col-s-1 {width: 8.33%;}

.col-s-2 {width: 16.66%;}

.col-s-3 {width: 25%;}

.col-s-4 {width: 33.33%;}

.col-s-5 {width: 41.66%;}

.col-s-6 {width: 50%;}

.col-s-7 {width: 58.33%;}

.col-s-8 {width: 66.66%;}

.col-s-9 {width: 75%;}

.col-s-10 {width: 83.33%;}

.col-s-11 {width: 91.66%;}

.col-s-12 {width: 100%;}

}

@media only screen and (min-width: 768px) {

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

}

html {

font-family: "Lucida Sans", sans-serif;

}

.header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.menu li {

padding: 8px;

margin-bottom: 7px;

background-color :#33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.menu li:hover {

background-color: #0099cc;

}

.aside {

background-color: #33b5e5;

padding: 15px;

color: #ffffff;

text-align: center;

font-size: 14px;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.footer {

background-color: #0099cc;

color: #ffffff;

text-align: center;

font-size: 12px;

padding: 15px;

}

</style>

</head>

<body>

<div class="header">

<h1>Shanghai</h1>

</div>

<div class="row">

<div class="col-3 col-s-3 menu">

<ul>

<li>交通</li>

<li>文化</li>

<li>旅游</li>

<li>美食</li>

</ul>

</div>

<div class="col-6 col-s-9">

<h1>欢迎来到上海</h1>

<p>上海市，简称沪，别称申，是中华人民共和国直辖市，中国的经济、金融、贸易和航运中心，世界著名的港口城市，是中国人口第二多的城市。</p>

<img src="/i/photo/shanghai.jpg" width="460" height="306">

</div>

<div class="col-3 col-s-12">

<div class="aside">

<h2>历史</h2>

<p>最晚在新石器时代，上海地区已经有先民聚居。春秋时代，上海由吴国管辖，战国时代则是楚国领土 ...</p>

<h2>位置</h2>

<p>上海位于中国东部弧形海岸线的正中间，长江三角洲最东部，东临东海，南濒杭州湾，西与江苏、浙江两省相接 ...</p>

<h2>环境</h2>

<p>上海地处江南水乡，并位于长江入海口，亦不处于主要地震带上，因此如地震、洪水以及地质类灾害鲜有发生 ...</p>

</div>

</div>

</div>

<div class="footer">

<p>请调整浏览器窗口的大小，来查看内容如何响应调整大小。</p>

</div>

</body>

</html>

## 105.4 背景图像

背景图像也可以响应调整大小和缩放比例。

这是我们展示的三种不同方法：

1. 如果将 background-size 属性设置为 "contain"，则背景图像将缩放，并尝试匹配内容区域。不过图像将保持其长宽比（图像宽度与高度之间的比例关系）：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

div {

width: 100%;

height: 400px;

background-image: url('/i/photo/flower-5.jpg');

background-repeat: no-repeat;

background-size: contain;

border: 1px solid red;

}

</style>

</head>

<body>

<p>请调整浏览器窗口大小来查看效果。</p>

<div></div>

</body>

</html>

1. 如果将 background-size 属性设置为 "100% 100%"，则背景图像将拉伸以覆盖整个内容区域：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

div {

width: 100%;

height: 400px;

background-image: url('/i/photo/flower-5.jpg');

background-size: 100% 100%;

border: 1px solid red;

}

</style>

</head>

<body>

<p>请调整浏览器窗口大小来查看效果。</p>

<div></div>

</body>

</html>

1. 如果 background-size 属性设置为 "cover"，则背景图像将缩放以覆盖整个内容区域。请注意，"cover" 值保持长宽比，且可能会裁剪背景图像的某部分：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

div {

width: 100%;

height: 400px;

background-image: url('/i/photo/flower-5.jpg');

background-size: cover;

border: 1px solid red;

}

</style>

</head>

<body>

<p>请调整浏览器窗口大小来查看效果。</p>

<div></div>

</body>

</html>

## 105.5 为不同设备准备不同图像

大幅的图像在大型计算机屏幕上可以完美显示，但在小型设备上就没用了。为什么在不得不缩小图像时又加载大图像呢？为了减少负载或出于任何其他原因，您可以使用媒体查询在不同的设备上显示不同的图像。

这是一幅大图像和一幅小图像，会在不同的设备上显示：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

/\* 针对宽度小于 400 像素的设备： \*/

body {

background-repeat: no-repeat;

background-image: url('/i/photo/tulip\_peach\_blossom\_s.jpg');

}

/\* 针对宽度大于等于 400 像素的设备： \*/

@media only screen and (min-width: 400px) {

body {

background-image: url('/i/photo/flower-5.jpg');

}

}

</style>

</head>

<body>

<p style="margin-top:660px;">请调整浏览器宽度，背景图像会在 400 像素（这个断点）发生变化。</p>

</body>

</html>

您可以使用媒体查询 min-device-width 而不是 min-width 来检查设备宽度，而不是浏览器宽度。然后，当您调整浏览器窗口的大小时，图像将不会变化：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

/\* 针对宽度小于 400 像素的设备： \*/

body {

background-repeat: no-repeat;

background-image: url('/i/photo/tulip\_peach\_blossom\_s.jpg');

}

/\* 针对宽度大于等于 400 像素的设备： \*/

@media only screen and (min-device-width: 400px) {

body {

background-image: url('/i/photo/flower-5.jpg');

}

}

</style>

</head>

<body>

</body>

</html>

## 105.6 HTML5 <picture> 元素

HTML5 引入了 <picture> 元素，该元素使您可以定义多幅图像。

<picture> 元素的作用类似于 <video> 和 <audio> 元素。我们设置了不同的来源，而匹配优先权的第一个来源是正在使用的来源：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<picture>

<source srcset="/i/photo/tulip\_peach\_blossom\_s.jpg" media="(max-width: 400px)">

<source srcset="/i/photo/flower-5.jpg">

<img src="/i/photo/flower-5.jpg" alt="Flowers" style="width:auto;">

</picture>

<p>请调整浏览器宽度，图像会在 400 像素（这个断点）发生变化。</p>

</body>

</html>

srcset 属性是必需的，它定义图像的来源。

media 属性是可选的，它接受可在 CSS @media 规则 中找到的媒体查询。

提示：您还应该为不支持 <picture> 元素的浏览器定义 <img> 元素。

# 第106章 响应式网页设计 - 视频

## 106.1 使用 width 属性

如果 width 属性设置为100％，则视频播放器将响应并放大和缩小：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

video {

width: 100%;

height: auto;

}

</style>

</head>

<body>

<video width="400" controls>

<source src="/i/photo/shanghai.mp4" type="video/mp4">

Your browser does not support HTML5 video.

</video>

<p>请调整浏览器窗口的大小，来查看视频播放器如何缩放。</p>

</body>

</html>

## 106.2 使用 max-width 属性

如果将 max-width 属性设置为 100％，则视频播放器将按比例缩小，但绝不会放大到大于其原始尺寸：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

video {

max-width: 100%;

height: auto;

}

</style>

</head>

<body>

<video width="400" controls>

<source src="/i/photo/shanghai.mp4" type="video/mp4">

Your browser does not support HTML5 video.

</video>

<p>请调整浏览器窗口的大小，查看宽度小于 400 像素时视频播放器如何缩放。</p>

</body>

</html>

## 106.3 在实例网页中添加视频

我们希望在实例网页中添加视频。视频将被调整大小以便始终占据所有可用空间：

实例

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

video {

width: 100%;

height: auto;

}

.row:after {

content: "";

clear: both;

display: table;

}

[class\*="col-"] {

float: left;

padding: 15px;

width: 100%;

}

@media only screen and (min-width: 600px) {

.col-s-1 {width: 8.33%;}

.col-s-2 {width: 16.66%;}

.col-s-3 {width: 25%;}

.col-s-4 {width: 33.33%;}

.col-s-5 {width: 41.66%;}

.col-s-6 {width: 50%;}

.col-s-7 {width: 58.33%;}

.col-s-8 {width: 66.66%;}

.col-s-9 {width: 75%;}

.col-s-10 {width: 83.33%;}

.col-s-11 {width: 91.66%;}

.col-s-12 {width: 100%;}

}

@media only screen and (min-width: 768px) {

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

}

html {

font-family: "Lucida Sans", sans-serif;

}

.header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

}

.menu li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.menu li:hover {

background-color: #0099cc;

}

.aside {

background-color: #33b5e5;

padding: 15px;

color: #ffffff;

text-align: center;

font-size: 14px;

box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);

}

.footer {

background-color: #0099cc;

color: #ffffff;

text-align: center;

font-size: 12px;

padding: 15px;

}

</style>

</head>

<body>

<div class="header">

<h1>上海</h1>

</div>

<div class="row">

<div class="col-3 col-s-3 menu">

<ul>

<li>交通</li>

<li>文化</li>

<li>旅游</li>

<li>美食</li>

</ul>

</div>

<div class="col-6 col-s-9">

<h1>欢迎来到上海</h1>

<p>上海市，简称沪，别称申，是中华人民共和国直辖市，中国的经济、金融、贸易和航运中心，世界著名的港口城市，是中国人口第二多的城市。</p>

<video width="400" controls>

<source src="/i/photo/shanghai.mp4" type="video/mp4">

Your browser does not support HTML5 video.

</video>

</div>

<div class="col-3 col-s-12">

<div class="aside">

<h2>历史</h2>

<p>最晚在新石器时代，上海地区已经有先民聚居。春秋时代，上海由吴国管辖，战国时代则是楚国领土 ...</p>

<h2>位置</h2>

<p>上海位于中国东部弧形海岸线的正中间，长江三角洲最东部，东临东海，南濒杭州湾，西与江苏、浙江两省相接 ...</p>

<h2>环境</h2>

<p>上海地处江南水乡，并位于长江入海口，亦不处于主要地震带上，因此如地震、洪水以及地质类灾害鲜有发生 ...</p>

</div>

</div>

</div>

<div class="footer">

<p>请调整浏览器窗口的大小，来查看内容如何响应调整大小。</p>

</div>

</body>

</html>

# 第107章 CSS 网格布局模块

## 107.1 网格布局

CSS 网格布局模块（CSS Grid Layout Module）提供了带有行和列的基于网格的布局系统，它使网页设计变得更加容易，而无需使用浮动和定位。

## 107.2 网格元素

网格布局由一个父元素以及一个或多个子元素组成。

实例

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>网格元素</h1>

<p>网格布局必须有一个 <em>display</em> 属性设置为 <em>grid</em> 或 <em>inline-grid</em> 的父元素。</p>

<p>网格容器的直接子元素自动成为网格项目。</p>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

</body>

</html>

## 107.3 Display 属性

当 HTML 元素的 display 属性设置为 grid 或 inline-grid 时，它就会成为网格容器。

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>display 属性：</h1>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

<p>请将 <em>display</em> 属性设置为 <em>grid</em>，以生成块级的网格容器。</p>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: inline-grid;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>display 属性：</h1>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

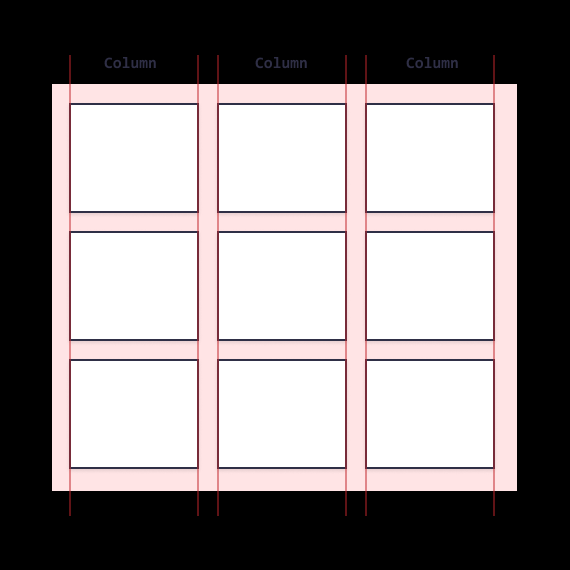
<p>请将 <em>display</em> 属性设置为 <em>inline-grid</em>，以生成行内的网格容器。</p>

</body>

</html>

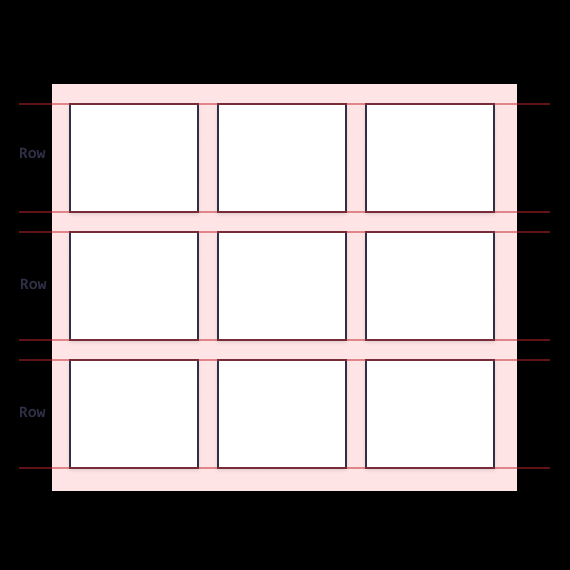
## 107.4 网格列（Grid Columns）

网格项的垂直线被称为列。



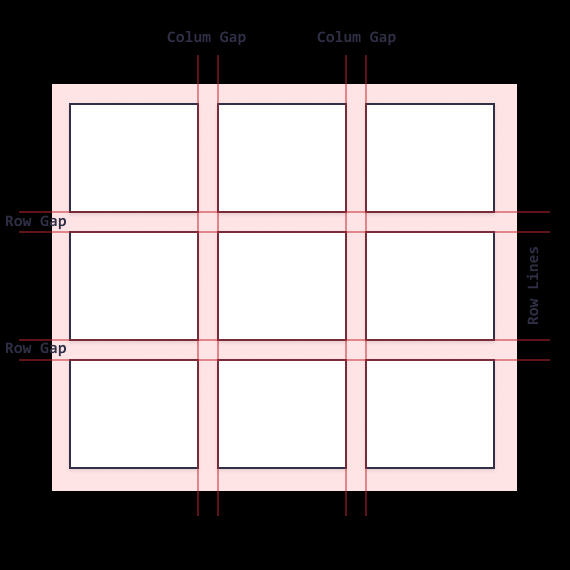
## 107.5 网隔行（Grid Rows）

网格项的水平线被称为行。



## 107.6 网格间隙（Grid Gaps）

每列/行之间的间隔称为间隙。



您可以通过使用以下属性之一来调整间隙大小：

grid-column-gap

grid-row-gap

grid-gap

实例

grid-column-gap 属性设置列之间的间隙：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-column-gap: 50px;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>grid-column-gap 属性：</h1>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

<p>请使用 <em>grid-column-gap</em> 属性调整列之间的空间。</p>

</body>

</html>

实例

grid-row-gap 属性设置行之间的间隙：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-row-gap: 50px;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>grid-row-gap 属性：</h1>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

<p>请使用 <em>grid-row-gap</em> 属性来调整行间距。</p>

</body>

</html>

实例

grid-gap 属性是 grid-row-gap 和 grid-column-gap 属性的简写属性：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-gap: 50px 100px;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>grid-gap 属性：</h1>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

<p>请使用 <em>grid-gap</em> 属性来调整列<em>和</em>行之间的空间。</p>

</body>

</html>

实例

grid-gap 属性还可用于将行间隙和列间隙设置为一个值：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-gap: 50px;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>grid-gap 属性：</h1>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

<p>请使用 <em>grid-gap</em> 属性来调整列<em>和</em>行之间的空间。</p>

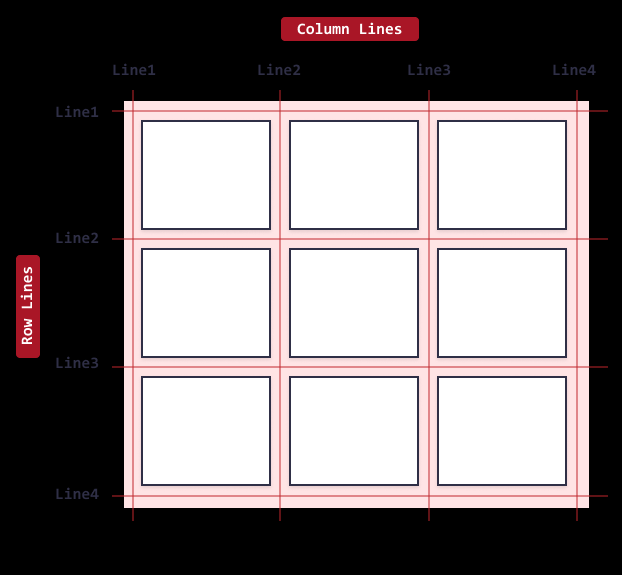
</body>

</html>

## 107.7 网格行（Grid Lines）

列之间的线称为列线（column lines）。

行之间的线称为行线（row lines）。



当把网格项目放在网格容器中时，请引用行号：

实例

把网格项目放在列线 1，并在列线 3 结束它：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item1 {

grid-column-start: 1;

grid-column-end: 3;

}

</style>

</head>

<body>

<h1>网格行</h1>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

</div>

<p>您可以在放置网格项时引用行号。</p>

</body>

</html>

实例

把网格项目放在行线 1，并在行线 3 结束它：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item1 {

grid-row-start: 1;

grid-row-end: 3;

}

</style>

</head>

<body>

<h1>网格行</h1>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

</div>

<p>您可以在放置网格项时引用行号。</p>

</body>

</html>

# 第108章 CSS 网格容器

## 108.1 网格容器

如需使 HTML 元素充当网格容器，您必须把 display 属性设置为 grid 或 inline-grid。

网格容器由放置在列和行内的网格项目组成。

## 108.2 grid-template-columns 属性

grid-template-columns 属性定义网格布局中的列数，并可定义每列的宽度。

该值是以空格分隔的列表，其中每个值定义相应列的长度。

如果您希望网格布局包含 4 列，请指定这 4 列的宽度；如果所有列都应当有相同的宽度，则设置为 "auto"。

实例

生成包含四列的网格：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-template-columns 属性：</h1>

<p>您可使用 <em>grid-template-columns</em> 属性来规定网格布局中的列数。</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

</div>

</body>

</html>

注意：如果在 4 列网格中有 4 个以上的项目，则网格会自动添加新行并将这些项目放入其中。

grid-template-columns 属性还可以用于指定列的尺寸（宽度）。

实例

设置这 4 列的尺寸：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: 80px 200px auto 30px;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-template-columns 属性：</h1>

<p>请使用 <em>grid-template-columns</em> 属性来规定每列的尺寸。</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

</div>

</body>

</html>

## 108.3 grid-template-rows 属性

grid-template-rows 属性定义每列的高度。

它的值是以空格分隔的列表，其中每个值定义相应行的高度：

实例

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

grid-template-rows: 80px 200px;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-template-rows 属性：</h1>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

<p>请使用 <em>grid-template-rows</em> 属性来规定每行的尺寸（高度）。</p>

</body>

</html>

## 108.4 justify-content 属性

justify-content 属性用于在容器内对齐整个网格。

注意：网格的总宽度必须小于容器的宽度，这样 justify-content 属性才能生效。

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

justify-content: space-evenly;

grid-template-columns: 50px 50px 50px; /\*Make the grid smaller than the container\*/

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>请使用 <em>justify-content</em> 属性在容器内对齐网格。</p>

<p>值 "space-evenly" 会在列之间以及列周围留出相等的空间：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

justify-content: space-around;

grid-template-columns: 50px 50px 50px; /\*Make the grid smaller than the container\*/

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>请使用 <em>justify-content</em> 属性在容器内对齐网格。</p>

<p>值 "space-around" 会在列周围留出相等的空间：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

justify-content: space-between;

grid-template-columns: 50px 50px 50px; /\*Make the grid smaller than the container\*/

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>请使用 <em>justify-content</em> 属性在容器内对齐网格。</p>

<p>值 "space-between" 会在列之间留出相等的空间：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

justify-content: center;

grid-template-columns: 50px 50px 50px; /\* 制作小于容器的网格 \*/

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>请使用 <em>justify-content</em> 属性在容器内对齐网格。</p>

<p>值 "center" 会在容器中间对齐网格：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

justify-content: start;

grid-template-columns: 50px 50px 50px; /\*Make the grid smaller than the container\*/

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>请使用 <em>justify-content</em> 属性在容器内对齐网格。</p>

<p>值 "start" 会在容器开头对齐网格：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

justify-content: end;

grid-template-columns: 50px 50px 50px; /\*Make the grid smaller than the container\*/

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>请使用 <em>justify-content</em> 属性在容器内对齐网格。</p>

<p>值 "end" 会在容器末端对齐网格：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

## 108.5 align-content 属性

align-content 属性用于垂直对齐容器内的整个网格。

注意：网格的总高度必须小于容器的高度，这样 align-content 属性才能生效。

实例

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

height: 400px;

align-content: center;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>请使用 <em>align-content</em> 属性垂直对齐容器内的网格。</p>

<p>值 "center" 会对齐容器中间的行：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

height: 400px;

align-content: space-evenly;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>请使用 <em>align-content</em> 属性垂直对齐容器内的网格。</p>

<p>值 "space-evenly" 将使行之间以及行周围具有相等的空间：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

height: 400px;

align-content: space-around;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>请使用 <em>align-content</em> 属性垂直对齐容器内的网格。</p>

<p>值 "space-around" 将使行周围具有相等的空间：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

height: 400px;

align-content: space-between;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>请使用 <em>align-content</em> 属性垂直对齐容器内的网格。</p>

<p>值 "space-between" 将使行之间具有相等的空间：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

height: 400px;

align-content: start;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>请使用 <em>align-content</em> 属性垂直对齐容器内的网格。</p>

<p>值 "start" 会对齐容器开头的行：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

height: 400px;

align-content: end;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<p>请使用 <em>align-content</em> 属性垂直对齐容器内的网格。</p>

<p>值 "end" 会对齐容器末端的行：</p>

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

</body>

</html>

# 第109章 CSS 网格项目

## 109.1 子元素（项目）

网格容器包含网格项目。

默认情况下，容器在每一行的每一列都有一个网格项目，但是您可以设置网格项目的样式，让它们跨越多个列和/或行。

## 109.2 grid-column 属性：

grid-column 属性定义将项目放置在哪一列上。

您可以定义项目的开始位置以及结束位置。

注释：grid-column 属性是 grid-column-start 和 grid-column-end 属性的简写属性。

如需放置某个项目，您可以引用行号（line numbers），或使用关键字 "span" 来定义该项目将跨越多少列。

实例

使 "item1" 从第 1 列开始并在第 5 列之前结束：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item1 {

grid-column: 1 / 5;

}

</style>

</head>

<body>

<h1>grid-column 属性：</h1>

<p>请使用 <em>grid-column</em> 属性规定在何处放置项目。</p>

<p>Item1 将在 column 1 开始，并在 column 5 之间结束：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

<div class="item10">10</div>

<div class="item11">11</div>

<div class="item12">12</div>

<div class="item13">13</div>

<div class="item14">14</div>

<div class="item15">15</div>

</div>

</body>

</html>

实例

使 "item1" 从第 1 列开始，并跨越 3 列：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item1 {

grid-column: 1 / span 3;

}

</style>

</head>

<body>

<h1>grid-column 属性：</h1>

<p>请使用 <em>grid-column</em> 属性规定在何处放置项目。</p>

<p>Item1 将在 column-line 1 开始，并横跨三列：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

<div class="item10">10</div>

<div class="item11">11</div>

<div class="item12">12</div>

<div class="item13">13</div>

<div class="item14">14</div>

<div class="item15">15</div>

<div class="item16">16</div>

</div>

</body>

</html>

实例

使 "item2" 从第 2 列开始，并跨越 3 列

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item2 {

grid-column: 2 / span 3;

}

</style>

</head>

<body>

<h1>grid-column 属性：</h1>

<p>请使用 <em>grid-column</em> 属性来规定在何处放置项目。</p>

<p>Item2 将在 column-line 2 开始，并横跨三列：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

<div class="item10">10</div>

<div class="item11">11</div>

<div class="item12">12</div>

<div class="item13">13</div>

<div class="item14">14</div>

<div class="item15">15</div>

<div class="item16">16</div>

</div>

</body>

</html>

## 109.3 grid-row 属性：

grid-row 属性定义了将项目放置在哪一行。

您可以定义项目的开始位置以及结束位置。

注释：grid-row 属性是 grid-row-start 和 grid-row-end 属性的简写属性。

如需放置项目，您可以引用行号，或使用关键字 "span" 定义该项目将跨越多少行：

实例

使 "item1" 在 row-line 1 开始，在 row-line 4 结束：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item1 {

grid-row: 1 / 4;

}

</style>

</head>

<body>

<h1>grid-row 属性：</h1>

<p>请使用 <em>grid-row</em> 属性来规定在何处放置项目。</p>

<p>Item1 将在 row-line 1 开始并在 row-line 4 结束：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

<div class="item10">10</div>

<div class="item11">11</div>

<div class="item12">12</div>

<div class="item13">13</div>

<div class="item14">14</div>

<div class="item15">15</div>

<div class="item16">16</div>

</div>

</body>

</html>

实例

使 "item1" 从第 1 行开始并跨越 2 行：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item1 {

grid-row: 1 / span 2;

}

</style>

</head>

<body>

<h1>grid-row 属性：</h1>

<p>请使用 <em>grid-row</em> 属性来规定在何处放置项目。</p>

<p>Item1 将在 row 1 开始并横跨两行：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

<div class="item10">10</div>

<div class="item11">11</div>

<div class="item12">12</div>

<div class="item13">13</div>

<div class="item14">14</div>

<div class="item15">15</div>

<div class="item16">16</div>

<div class="item17">17</div>

</div>

</body>

</html>

## 109.4 grid-area 属性

grid-area 属性可以用作 grid-row-start、grid-column-start、grid-row-end 和 grid-column-end 属性的简写属性。

实例

使 "item8" 从 row-line 1 和 column-line 2 开始，在 row-line 5 和 column line 6 结束：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item8 {

grid-area: 1 / 2 / 5 / 6;

}

</style>

</head>

<body>

<h1>grid-area 属性</h1>

<p>您可以使用 <em>grid-area</em> 属性来规定在何处放置项目。</p>

<p>语法是：</p>

<p>grid-row-start / grid-column-start / grid-row-end / grid-column-end.</p>

<p>Item8 将在 row-line 1 和 column-line 2 开始，并在 row-line 5 column-line 6 结束：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

<div class="item10">10</div>

<div class="item11">11</div>

<div class="item12">12</div>

<div class="item13">13</div>

<div class="item14">14</div>

<div class="item15">15</div>

</div>

</body>

</html>

实例

使 "item8" 从 row-line 2 和 column-line 开始，并跨越 2 行和 3 列：

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item8 {

grid-area: 2 / 1 / span 2 / span 3;

}

</style>

</head>

<body>

<h1>grid-area 属性</h1>

<p>您可以使用 <em>grid-area</em> 属性来规定在何处放置项目。</p>

<p>语法是：grid-row-start / grid-column-start / grid-row-end / grid-column-end.</p>

<p>Item8 将在 row-line 2 和 column-line 1，并横跨两行三列：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

<div class="item10">10</div>

<div class="item11">11</div>

<div class="item12">12</div>

<div class="item13">13</div>

</div>

</body>

</html>

## 109.5 命名网格项

grid-area 属性也可以用于为网格项目分配名称。

可以通过网格容器的 grid-template-areas 属性来引用命名的网格项目。

实例

item1 的名称是 "myArea"，并跨越五列网格布局中的所有五列：

<!DOCTYPE html>

<html>

<head>

<style>

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas: 'myArea myArea myArea myArea myArea';

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-area 属性</h1>

<p>您可使用 <em>grid-area</em> 属性来命名网格项目。</p>

<p>通过在网格容器上使用 <em>grid-template-areas</em>属性，您可以在设置网格布局时引用该名称。</p>

<p>Item1 的名为 "myArea"，并会占据全部五列：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

</div>

</body>

</html>

每行由撇号（' '）定义。

每行中的列都在撇号内定义，并以空格分隔。

注释：句号表示没有名称的网格项目。

实例

让 "myArea" 跨越五列网格布局中的两列（句号代表没有名称的项目）：

<!DOCTYPE html>

<html>

<head>

<style>

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas: 'myArea myArea . . .';

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-area 属性</h1>

<p>您可使用 <em>grid-area</em> 属性来命名网格项目。</p>

<p>通过在网格容器上使用 <em>grid-template-areas</em>属性，您可以在设置网格布局时引用该名称。</p>

<p>Item1 名为 "myArea"，并会占据（五列中的）两列的位置：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

</div>

</body>

</html>

每行由撇号（' '）定义。

每行中的列都在撇号内定义，并以空格分隔。

注释：句号表示没有名称的网格项目。

实例

让 "myArea" 跨越五列网格布局中的两列（句号代表没有名称的项目）：

<!DOCTYPE html>

<html>

<head>

<style>

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas: 'myArea myArea . . .';

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-area 属性</h1>

<p>您可使用 <em>grid-area</em> 属性来命名网格项目。</p>

<p>通过在网格容器上使用 <em>grid-template-areas</em>属性，您可以在设置网格布局时引用该名称。</p>

<p>Item1 名为 "myArea"，并会占据（五列中的）两列的位置：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

<div class="item8">8</div>

<div class="item9">9</div>

</div>

</body>

</html>

如需定义两行，请在另一组撇号内定义第二行的列：

实例

使 "item1" 跨越两列和两行：

<!DOCTYPE html>

<html>

<head>

<style>

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas: 'myArea myArea . . .' 'myArea myArea . . .';

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-area 属性</h1>

<p>您可使用 <em>grid-area</em> 属性来命名网格项目。</p>

<p>通过在网格容器上使用 <em>grid-template-areas</em>属性，您可以在设置网格布局时引用该名称。</p>

<p>Item1 名为 "myArea"，会占据（五列中的）两列的位置，并将横跨两行：</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

<div class="item7">7</div>

</div>

</body>

</html>

实例

命名所有项目，并制作一张随时可用的网页模板：

<!DOCTYPE html>

<html>

<head>

<style>

.item1 { grid-area: header; }

.item2 { grid-area: menu; }

.item3 { grid-area: main; }

.item4 { grid-area: right; }

.item5 { grid-area: footer; }

.grid-container {

display: grid;

grid-template-areas:

'header header header header header header'

'menu main main main right right'

'menu footer footer footer footer footer';

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

</style>

</head>

<body>

<h1>grid-area 属性</h1>

<p>您可使用 <em>grid-area</em> 属性来命名网格项目。</p>

<p>通过在网格容器上使用 <em>grid-template-areas</em>属性，您可以在设置网格布局时引用该名称。</p>

<p>此网格布局包含六列三行：</p>

<div class="grid-container">

<div class="item1">Header</div>

<div class="item2">Menu</div>

<div class="item3">Main</div>

<div class="item4">Right</div>

<div class="item5">Footer</div>

</div>

</body>

</html>

## 109.6 项目的顺序

网格布局允许我们将项目放置在我们喜欢的任意位置。

HTML 代码中的第一项不必显示为网格中的第一项。

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

.item1 { grid-area: 1 / 3 / 2 / 4; }

.item2 { grid-area: 2 / 3 / 3 / 4; }

.item3 { grid-area: 1 / 1 / 2 / 2; }

.item4 { grid-area: 1 / 2 / 2 / 3; }

.item5 { grid-area: 2 / 1 / 3 / 2; }

.item6 { grid-area: 2 / 2 / 3 / 3; }

</style>

</head>

<body>

<h1>对项目排序：</h1>

<p>这些项目不必以与它们在 HTML 代码中编写的顺序相同的顺序显示。</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

</div>

</body>

</html>

您可以通过使用媒体查询来重新排列某些屏幕尺寸的顺序：

实例

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

grid-gap: 10px;

background-color: #2196F3;

padding: 10px;

}

.grid-container > div {

background-color: rgba(255, 255, 255, 0.8);

text-align: center;

padding: 20px 0;

font-size: 30px;

}

@media only screen and (max-width: 500px) {

.item1 { grid-area: 1 / span 3 / 2 / 4; }

.item2 { grid-area: 3 / 3 / 4 / 4; }

.item3 { grid-area: 2 / 1 / 3 / 2; }

.item4 { grid-area: 2 / 2 / span 2 / 3; }

.item5 { grid-area: 3 / 1 / 4 / 2; }

.item6 { grid-area: 2 / 3 / 3 / 4; }

}

</style>

</head>

<body>

<h1>在小型设备上重新排序</h1>

<p>请将窗口调整为 500 像素来查看效果。</p>

<div class="grid-container">

<div class="item1">1</div>

<div class="item2">2</div>

<div class="item3">3</div>

<div class="item4">4</div>

<div class="item5">5</div>

<div class="item6">6</div>

</div>

</body>

</html>

# 第110章 CSS 参考手册

## 110.1 CSS 属性

A:

|  |  |
| --- | --- |
| align-content | 规定弹性容器内的行之间的对齐方式，当项目不使用所有可用空间时。 |
| align-items | 规定弹性容器内项目的对齐方式。 |
| align-self | 规定弹性容器内所选项目的对齐方式。 |
| all | 重置所有属性（除了 unicode-bidi 和 direction）。 |
| animation | 所有 animation-\* 属性的简写属性。 |
| animation-delay | 规定开始动画的延迟。 |
| animation-direction | 规定动画是向前播放、向后播放还是交替播放。 |
| animation-duration | 规定动画完成一个周期应花费的时间。 |
| animation-fill-mode | 规定元素在不播放动画时（在开始之前、结束之后、或同时）的样式。 |
| animation-iteration-count | 规定动画的播放次数。 |
| animation-name | 规定 @keyframes 动画的名称。 |
| animation-play-state | 规定动画是播放还是暂停。 |
| animation-timing-function | 规定动画的速度曲线。 |

B:

|  |  |
| --- | --- |
| backface-visibility | 定义当面对用户时元素的背面是否应可见。 |
| background | 所有 background-\* 属性的简写属性。 |
| background-attachment | 设置背景图像是与页面的其余部分一起滚动还是固定的。 |
| background-blend-mode | 规定每个背景图层（颜色/图像）的混合模式。 |
| background-clip | 定义背景（颜色或图像）应在元素内延伸的距离。 |
| background-color | 规定元素的背景色。 |
| background-image | 规定元素的一幅或多幅背景图像。 |
| background-origin | 规定背景图像的初始位置。 |
| background-position | 规定背景图像的位置。 |
| background-repeat | 设置是否以及如何重复背景图像。 |
| background-size | 规定背景图像的尺寸。 |
| border | border-width、border-style 以及 border-color 的简写属性。 |
| border-bottom | border-bottom-width、border-bottom-style 以及 border-bottom-color 的简写属性。 |
| border-bottom-color | 设置下边框的颜色。 |
| border-bottom-left-radius | 定义左下角的边框圆角。 |
| border-bottom-right-radius | 定义右下角的边框圆角。 |
| border-bottom-style | 设置下边框的样式。 |
| border-bottom-width | 设置下边框的宽度。 |
| border-collapse | 设置表格边框是折叠为单一边框还是分开的。 |
| border-color | 设置四条边框的颜色。 |
| border-image | border-image-\* 属性的简写属性。 |
| border-image-outset | 规定边框图像区域超出边框的量。 |
| border-image-repeat | 规定边框图像应重复、圆角、还是拉伸。 |
| border-image-slice | 规定如何裁切边框图像。 |
| border-image-source | 规定用作边框的图像的路径。 |
| border-image-width | 规定边框图像的宽度。 |
| border-left | 所有 border-left-\* 属性的简写属性。 |
| border-left-color | 设置左边框的颜色。 |
| border-left-style | 设置左边框的样式。 |
| border-left-width | 设置左边框的宽度。 |
| border-radius | 四个 border-\*-radius 属性的简写属性。 |
| border-right | 所有 border-right-\* 属性的简写属性。 |
| border-right-color | 设置右边框的颜色。 |
| border-right-style | 设置右边框的样式。 |
| border-right-width | 设置右边框的宽度。 |
| border-spacing | 设置相邻单元格边框之间的距离。 |
| border-style | 设置四条边框的样式。 |
| border-top | border-top-width、border-top-style 以及 border-top-color 的简写属性。 |
| border-top-color | 设置上边框的颜色。 |
| border-top-left-radius | 定义左上角的边框圆角。 |
| border-top-right-radius | 定义右上角的边框圆角。 |
| border-top-style | 设置上边框的样式。 |
| border-top-width | 设置上边框的宽度。 |
| border-width | 设置四条边框的宽度。 |
| bottom | 设置元素相对于其父元素底部的位置。 |
| box-decoration-break | 设置元素在分页符处的背景和边框的行为，或对于行内元素在换行符处的行为。 |
| box-shadow | 将一个或多个阴影附加到元素。 |
| box-sizing | 定义元素的宽度和高度的计算方式：它们是否应包含内边距和边框。 |
| break-after | 规定指定元素之后是否应出现 page-、column- 或 region-break。 |
| break-before | 规定指定元素之前是否应出现 page-、column- 或 region-break。 |
| break-inside | 规定指定元素内部是否应出现 page-、column- 或 region-break。 |

C:

|  |  |
| --- | --- |
| caption-side | 规定表格标题的放置方式。 |
| caret-color | 规定光标在 input、textarea 或任何可编辑元素中的颜色。 |
| @charset | 规定样式表中使用的字符编码。 |
| clear | 规定不允许在元素的哪一侧浮动元素 |
| clip | 剪裁绝对定位的元素。 |
| clip-path | 将元素裁剪为基本形状或 SVG 源。 |
| color | 设置文本的颜色。 |
| column-count | 规定元素应分为的列数。 |
| column-fill | 指定如何填充列（是否 balanced）。 |
| column-gap | 规定列间隙。 |
| column-rule | 所有 column-rule-\* 属性的简写属性。 |
| column-rule-color | 规定列之间规则的颜色。 |
| column-rule-style | 规定列之间的规则样式。 |
| column-rule-width | 规定列之间的规则宽度。 |
| column-span | 规定元素应该跨越多少列。 |
| column-width | 规定列宽度。 |
| columns | column-width 和 column-count 的简写属性。 |
| content | 与 :before 和 :after 伪元素一起使用，来插入生成的内容。 |
| counter-increment | 增加或减少一个或多个 CSS 计数器的值。 |
| counter-reset | 创建或重置一个或多个 CSS 计数器。 |
| cursor | 规定当指向元素时要显示的鼠标光标。 |

D:

|  |  |
| --- | --- |
| direction | 规定文本方向/书写方向。 |
| display | 规定如何显示某个 HTML 元素。 |

E:

|  |  |
| --- | --- |
| empty-cells | 规定是否在表格中的空白单元格上显示边框和背景。 |

F:

|  |  |
| --- | --- |
| filter | 定义元素显示之前的效果（例如，模糊或颜色偏移）。 |
| flex | flex-grow、flex-shrink 以及 flex-basis 的简写属性。 |
| flex-basis | 规定弹性项目的初始长度。 |
| flex-direction | 规定弹性项目的方向。 |
| flex-flow | flex-direction 和 flex-wrap 的简写属性。 |
| flex-grow | 规定项目相对于其余项目的增量。 |
| flex-shrink | 规定项目相对于其余项目的减量。 |
| flex-wrap | 规定弹性项目是否应该换行。 |
| float | 规定是否应该对盒（box）进行浮动。 |
| font | font-style、font-variant、font-weight、font-size/line-height 以及 font-family 的简写属性。 |
| @font-face | 允许网站下载和使用 "web-safe" 字体以外的其他字体的规则。 |
| font-family | 规定文本的字体族（字体系列）。 |
| font-feature-settings | 允许控制 OpenType 字体中的高级印刷特性。 |
| @font-feature-values | 允许创作者使用 font-variant-alternate 中的通用名来实现在 OpenType 中以不同方式激活的特性。 |
| font-kerning | 控制字距调整信息的使用（字母间距）。 |
| font-language-override | 控制特定语言的字形在字体的使用。 |
| font-size | 规定文本的字体大小。 |
| font-size-adjust | 保持发生字体回退时的可读性。 |
| font-stretch | 从字体系列中选择一个普通的、压缩的或扩展的字体。 |
| font-style | 规定文本的字体样式。 |
| font-synthesis | 控制哪些缺失的字体（粗体或斜体）可以由浏览器合成。 |
| font-variant | 规定是否应该以小型大写字体显示文本。 |
| font-variant-alternates | 控制与 @font-feature-values 中定义的备用名称关联的备用字形的使用。 |
| font-variant-caps | 控制大写字母的备用字形的使用。 |
| font-variant-east-asian | 控制东亚文字（例如中文和日语）的备用字形的使用。 |
| font-variant-ligatures | 控制在适用于元素的文本内容中使用哪些连字和上下文形式。 |
| font-variant-numeric | 控制数字、分数和序号标记的备用字形的使用。 |
| font-variant-position | 控制较小字体的替代字形的使用，这些字形相对于字体基线定位为上标或下标。 |
| font-weight | 规定字体的粗细。 |

G:

|  |  |
| --- | --- |
| grid | grid-template-rows、grid-template-columns、grid-template-areas、grid-auto-rows、grid-auto-columns 以及 grid-auto-flow 属性的简写属性。 |
| grid-area | 即可规定网格项的名称，也可以是 grid-row-start、grid-column-start、grid-row-end 以及 grid-column-end 属性的简写属性。 |
| grid-auto-columns | 规定默认的列尺寸。 |
| grid-auto-flow | 规定如何在网格中插入自动放置的项目。 |
| grid-auto-rows | 规定默认的行尺寸。 |
| grid-column | grid-column-start 和 grid-column-end 属性的简写属性。 |
| grid-column-end | 规定如何结束网格项目。 |
| grid-column-gap | 规定列间隙的尺寸。 |
| grid-column-start | 规定网格项目从何处开始。 |
| grid-gap | grid-row-gap 和 grid-column-gap 的简写属性。 |
| grid-row | grid-row-start 和 grid-row-end 属性的简写属性。 |
| grid-row-end | 规定网格项目在何处结束。 |
| grid-row-gap | 规定列间隙的尺寸。 |
| grid-row-start | 规定网格项目从何处开始。 |
| grid-template | grid-template-rows、grid-template-columns 以及 grid-areas 属性的简写属性。 |
| grid-template-areas | 规定如何使用命名的网格项显示列和行。 |
| grid-template-columns | 指定列的尺寸以及网格布局中的列数。 |
| grid-template-rows | 指定网格布局中的行的尺寸。 |

H:

|  |  |
| --- | --- |
| hanging-punctuation | 规定是否可以在行框外放置标点符号。 |
| height | 设置元素的高度。 |
| hyphens | 设置如何分割单词以改善段落的布局。 |

I:

|  |  |
| --- | --- |
| image-rendering | 当图像被缩放时，向浏览器提供关于保留图像的哪些最重要的方面的信息。 |
| @import | 允许您将样式表导入另一张样式表。 |
| isolation | 定义元素是否必须创建新的堆叠内容。 |

J:

|  |  |
| --- | --- |
| justify-content | 规定项目在弹性容器内的对齐方式，当项目未用到所有可用空间时。 |

K:

|  |  |
| --- | --- |
| @keyframes | 规定动画代码。 |

L:

|  |  |
| --- | --- |
| left | 规定定位元素的左侧位置。 |
| letter-spacing | 增加或减少文本中的字符间距。 |
| line-break | 如何如何/是否换行。 |
| line-height | 设置行高。 |
| list-style | 在一条声明中设置所有列表属性。 |
| list-style-image | 把图像指定为列表项标记。 |
| list-style-position | 规定列表项标记的位置。 |
| list-style-type | 规定列表项标记的类型。 |

M:

|  |  |
| --- | --- |
| margin | 在一条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。 |
| mask | 通过在特定位置遮罩或剪切图像来隐藏元素。 |
| mask-type | 规定将遮罩元素用作亮度或 Alpha 遮罩。 |
| max-height | 设置元素的最大高度。 |
| max-width | 设置元素的最大宽度。 |
| @media | 为不同的媒体类型、设备、尺寸设置样式规则。 |
| min-height | 设置元素的最小高度。 |
| min-width | 设置元素的最小宽度。 |
| mix-blend-mode | 规定元素内容应如何与其直接父的背景相混合。 |

O:

|  |  |
| --- | --- |
| object-fit | 规定替换元素的内容应如何适合其所用高度和宽度建立的框。 |
| object-position | 指定替换元素在其框内的对齐方式。 |
| opacity | 设置元素的不透明等级。 |
| order | 设置弹性项目相对于其余项目的顺序。 |
| orphans | 设置在元素内发生分页时必须保留在页面底部的最小行数。 |
| outline | outline-width、outline-style 以及 outline-color 属性的简写属性。 |
| outline-color | 设置轮廓的颜色。 |
| outline-offset | 对轮廓进行偏移，并将其绘制到边框边缘之外。 |
| outline-style | 设置轮廓的样式。 |
| outline-width | 设置轮廓的宽度。 |
| overflow | 规定如果内容溢出元素框会发生什么情况。 |
| overflow-wrap | 规定浏览器是否可能为了防止溢出而在单词内折行（当字符串太长而无法适应其包含框时）。 |
| overflow-x | 规定是否剪裁内容的左右边缘，如果它溢出了元素的内容区域。 |
| overflow-y | 规定是否剪裁内容的上下边缘，如果它溢出了元素的内容区域。 |

P:

|  |  |
| --- | --- |
| padding | 所有 padding-\* 属性的简写属性。 |
| 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。 |
| page-break-after | 设置元素之后的分页（page-break）行为。 |
| page-break-before | 设置元素之前的分页（page-break）行为。 |
| page-break-inside | 设置元素内的分页（page-break）行为。 |
| perspective | 为 3D 定位元素提供透视。 |
| perspective-origin | 定义用户观看 3D 定位元素的位置。 |
| pointer-events | 定义元素是否对指针事件做出反应。 |
| position | 规定用于元素的定位方法的类型（静态、相对、绝对或固定）。 |

Q:

|  |  |
| --- | --- |
| quotes | 设置引号类型。 |

R:

|  |  |
| --- | --- |
| resize | 定义用户是否以及如何调整元素的尺寸。 |
| right | 规定定位元素的左侧位置。 |

S:

|  |  |
| --- | --- |
| scroll-behavior | 规定可滚动框中是否平滑地滚动，而不是直接跳跃。 |

T:

|  |  |
| --- | --- |
| tab-size | 规定制表符的宽度。 |
| table-layout | 定义用于对单元格、行和列进行布局的算法。 |
| text-align | 规定文本的水平对齐方式。 |
| text-align-last | 描述当 text-align 为 "justify" 时，如何在强制换行之前对齐块或行的最后一行。 |
| text-combine-upright | 将多个字符组合到到单个字符的空间中。 |
| text-decoration | 规定文本装饰。 |
| text-decoration-color | 规定文本装饰（text-decoration）的颜色。 |
| text-decoration-line | 规定文本装饰（text-decoration）中的的行类型。 |
| text-decoration-style | 规定文本装饰（text-decoration）中的行样式。 |
| text-indent | 规定文本块（text-block）中的的首行缩进。 |
| text-justify | 规定当 text-align 为 "justify" 时使用的对齐方法。 |
| text-orientation | 定义行中的文本方向。 |
| text-overflow | 规定当文本溢出包含元素时应该发生的情况。 |
| text-shadow | 添加文本阴影。 |
| text-transform | 控制文本的大写。 |
| text-underline-position | 规定使用 text-decoration 属性设置的下划线的位置。 |
| top | 规定定位元素的顶端位置。 |
| transform | 向元素应用 2D 或 3D 转换。 |
| transform-origin | 允许您更改转换元素的位置。 |
| transform-style | 规定如何在 3D 空间中渲染嵌套的元素。 |
| transition | 所有 transition-\* 属性的简写属性。 |
| transition-delay | 规定合适开始过渡效果。 |
| transition-duration | 规定完成过渡效果所需的秒或毫秒数。 |
| transition-property | 规定过渡效果对应的 CSS 属性的名称。 |
| transition-timing-function | 规定过渡效果的速度曲线。 |

U:

|  |  |
| --- | --- |
| unicode-bidi | 与 direction 属性一起使用，设置或返回是否应覆写文本来支持同一文档中的多种语言。 |
| user-select | 规定是否能选取元素的文本。 |

V:

|  |  |
| --- | --- |
| vertical-align | 设置元素的垂直对齐方式。 |
| visibility | 规定元素是否可见。 |

W:

|  |  |
| --- | --- |
| white-space | 规定如何处理元素内的空白字符。 |
| widows | 设置如果元素内发生分页，必须在页面顶部保留的最小行数。 |
| width | 设置元素的宽度。 |
| word-break | 规定单词到达行末后如何换行。 |
| word-spacing | 增加或减少文本中的单词间距。 |
| word-wrap | 允许长的、不能折行的单词换到下一行。 |
| writing-mode | 规定文本行是水平还是垂直布局。 |

Z:

|  |  |
| --- | --- |
| z-index | 设置定位元素的堆叠顺序。 |

# 第111章 CSS 选择器参考手册

## 111.1 CSS 选择器

在 CSS 中，选择器是选取需设置样式的元素的模式。

请使用我们的 CSS 选择器测试工具，它可为您演示不同的选择器。

|  |  |  |
| --- | --- | --- |
| 选择器 | 例子 | 例子描述 |
| .class | .intro | 选择 class="intro" 的所有元素。 |
| .class1.class2 | .name1.name2 | 选择 class 属性中同时有 name1 和 name2 的所有元素。 |
| .class1 .class2 | .name1 .name2 | 选择作为类名 name1 元素后代的所有类名 name2 元素。 |
| #id | #firstname | 选择 id="firstname" 的元素。 |
| \* | \* | 选择所有元素。 |
| element | p | 选择所有 <p> 元素。 |
| element.class | p.intro | 选择 class="intro" 的所有 <p> 元素。 |
| element,element | div, p | 选择所有 <div> 元素和所有 <p> 元素。 |
| element element | div p | 选择 <div> 元素内的所有 <p> 元素。 |
| element>element | div > p | 选择父元素是 <div> 的所有 <p> 元素。 |
| element+element | div + p | 选择紧跟 <div> 元素的首个 <p> 元素。 |
| element1~element2 | p ~ ul | 选择前面有 <p> 元素的每个 <ul> 元素。 |
| [attribute] | [target] | 选择带有 target 属性的所有元素。 |
| [attribute=value] | [target=\_blank] | 选择带有 target="\_blank" 属性的所有元素。 |
| [attribute~=value] | [title~=flower] | 选择 title 属性包含单词 "flower" 的所有元素。 |
| [attribute|=value] | [lang|=en] | 选择 lang 属性值以 "en" 开头的所有元素。 |
| [attribute^=value] | a[href^="https"] | 选择其 src 属性值以 "https" 开头的每个 <a> 元素。 |
| [attribute$=value] | a[href$=".pdf"] | 选择其 src 属性以 ".pdf" 结尾的所有 <a> 元素。 |
| [attribute\*=value] | a[href\*="w3schools"] | 选择其 href 属性值中包含 "abc" 子串的每个 <a> 元素。 |
| :active | a:active | 选择活动链接。 |
| ::after | p::after | 在每个 <p> 的内容之后插入内容。 |
| ::before | p::before | 在每个 <p> 的内容之前插入内容。 |
| :checked | input:checked | 选择每个被选中的 <input> 元素。 |
| :default | input:default | 选择默认的 <input> 元素。 |
| :disabled | input:disabled | 选择每个被禁用的 <input> 元素。 |
| :empty | p:empty | 选择没有子元素的每个 <p> 元素（包括文本节点）。 |
| :enabled | input:enabled | 选择每个启用的 <input> 元素。 |
| :first-child | p:first-child | 选择属于父元素的第一个子元素的每个 <p> 元素。 |
| ::first-letter | p::first-letter | 选择每个 <p> 元素的首字母。 |
| ::first-line | p::first-line | 选择每个 <p> 元素的首行。 |
| :first-of-type | p:first-of-type | 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。 |
| :focus | input:focus | 选择获得焦点的 input 元素。 |
| :fullscreen | :fullscreen | 选择处于全屏模式的元素。 |
| :hover | a:hover | 选择鼠标指针位于其上的链接。 |
| :in-range | input:in-range | 选择其值在指定范围内的 input 元素。 |
| :indeterminate | input:indeterminate | 选择处于不确定状态的 input 元素。 |
| :invalid | input:invalid | 选择具有无效值的所有 input 元素。 |
| :lang(language) | p:lang(it) | 选择 lang 属性等于 "it"（意大利）的每个 <p> 元素。 |
| :last-child | p:last-child | 选择属于其父元素最后一个子元素每个 <p> 元素。 |
| :last-of-type | p:last-of-type | 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。 |
| :link | a:link | 选择所有未访问过的链接。 |
| :not(selector) | :not(p) | 选择非 <p> 元素的每个元素。 |
| :nth-child(n) | p:nth-child(2) | 选择属于其父元素的第二个子元素的每个 <p> 元素。 |
| :nth-last-child(n) | p:nth-last-child(2) | 同上，从最后一个子元素开始计数。 |
| :nth-of-type(n) | p:nth-of-type(2) | 选择属于其父元素第二个 <p> 元素的每个 <p> 元素。 |
| :nth-last-of-type(n) | p:nth-last-of-type(2) | 同上，但是从最后一个子元素开始计数。 |
| :only-of-type | p:only-of-type | 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。 |
| :only-child | p:only-child | 选择属于其父元素的唯一子元素的每个 <p> 元素。 |
| :optional | input:optional | 选择不带 "required" 属性的 input 元素。 |
| :out-of-range | input:out-of-range | 选择值超出指定范围的 input 元素。 |
| ::placeholder | input::placeholder | 选择已规定 "placeholder" 属性的 input 元素。 |
| :read-only | input:read-only | 选择已规定 "readonly" 属性的 input 元素。 |
| :read-write | input:read-write | 选择未规定 "readonly" 属性的 input 元素。 |
| :required | input:required | 选择已规定 "required" 属性的 input 元素。 |
| :root | :root | 选择文档的根元素。 |
| ::selection | ::selection | 选择用户已选取的元素部分。 |
| :target | #news:target | 选择当前活动的 #news 元素。 |
| :valid | input:valid | 选择带有有效值的所有 input 元素。 |
| :visited | a:visited | 选择所有已访问的链接。 |

# 第112章 CSS 函数参考手册

## 112.1 CSS 函数

CSS 函数用作各种CSS属性的值。

|  |  |
| --- | --- |
| 函数 | 描述 |
| attr() | 返回所选元素的属性值。 |
| calc() | 允许您执行计算来确定 CSS 属性值。 |
| cubic-bezier() | 定义三次贝塞尔曲线。 |
| hsl() | 使用色相-饱和度-亮度模型（HSL）定义颜色。 |
| hsla() | 使用色相-饱和度-亮度-阿尔法模型（HSLA）定义颜色。 |
| linear-gradient() | 将线性渐变设置为背景图像。定义至少两种颜色（从上到下）。 |
| radial-gradient() | 将径向渐变设置为背景图像。定义至少两种颜色（从中心到边缘）。 |
| repeating-linear-gradient() | 重复线性渐变。 |
| repeating-radial-gradient() | 重复径向渐变。 |
| rgb() | 使用红-绿-蓝模型（RGB）定义颜色。 |
| rgba() | 使用红-绿-蓝-阿尔法模型（RGB）定义颜色。 |
| var() | 插入自定义属性的值。 |

# 第113章 CSS 动画相关属性

## 113.1 定义和用法

一些 CSS 属性可用于动画制作，这意味着它们可用于过渡等效果中。

可设置动画的属性可以从一个值逐渐更改为另一个值，例如尺寸、数字、百分比和颜色。

## 113.2 浏览器支持

表格中的数字注明了完全支持 CSS 动画的首个浏览器版本。

-webkit-、-moz- 或 -o- 后面的数字注明了使用前缀的第一个版本。

## 113.3 动画相关属性

下面的表格中列出了 CSS 中的动画相关属性：

|  |  |
| --- | --- |
| 属性 | TIY |
| background | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  background: red url('smiley.gif') no-repeat top left/5px 5px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {background: blue bottom right/50px 50px;}  }  </style>  </head>  <body>  <h1>background 的动画效果：</h1>  <p>逐渐更改背景属性：<p>  <div id="myDIV"></div>  <p>background属性是所有背景属性的简写属性。</p>  <p>在 CSS 总，background-color、background-position 以及 background-size 属性<em>支持动画制作</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| background-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  background: red;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {background-color: blue;}  }  </style>  </head>  <body>  <h1>background-color 的动画效果：</h1>  <p>Gradually change the background-color from red to blue, and back to red:<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| background-position | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  background-image: url('smiley.gif');  background-position: top left;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {background-position: center;}  }  </style>  </head>  <body>  <h1>background-position 的动画效果：</h1>  <p>逐渐将背景位置从“左上方”更改为“中心”，然后再改回为“左上方”：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| background-size | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  background-image: url('/i/logo/w3logo-1.png');  background-size: 20px 20px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {background-size: 125px 125px;}  }  </style>  </head>  <body>  <h1>background-size 的动画效果：</h1>  <p>逐渐将背景尺寸从“10px 10px”更改为“125px 125px”，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border: 15px solid lightblue;}  }  </style>  </head>  <body>  <h1>border 的动画效果：</h1>  <p>逐渐更改 border 属性：<p>  <div id="myDIV"></div>  <p>border 属性是所有边框属性的简写属性。</p>  <p>请参阅单个边框属性，来查看哪个属性<em>支持动画制作</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-bottom | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-bottom: 15px solid lightblue;}  }  </style>  </head>  <body>  <h1>border-bottom 的动画效果：</h1>  <p>逐渐更改 border-bottom 属性：<p>  <div id="myDIV"></div>  <p>border-bottom 是所有 border-bottom 属性的简写属性。</p>  <p>请查看单个的 border-bottom 的属性，来了解哪些是<em>支持动画制作的</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-bottom-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 15px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-bottom-color: lightblue;}  }  </style>  </head>  <body>  <h1>border-bottom-color 的动画效果：</h1>  <p>逐渐将 border-bottom-color 属性从黑色更改为蓝色：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-bottom-left-radius | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-bottom-left-radius: 150px;}  }  </style>  </head>  <body>  <h1>border-bottom-left-radius 的动画效果：</h1>  <p>逐渐将 border-bottom-left-radius 属性从 0 改为 150px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-bottom-right-radius | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-bottom-right-radius: 150px;}  }  </style>  </head>  <body>  <h1>border-bottom-right-radius 的动画效果：</h1>  <p>逐渐将 border-bottom-right-radius 属性从 0 改为 150px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-bottom-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-bottom-width: 50px;}  }  </style>  </head>  <body>  <h1>border-bottom-width 的动画效果：</h1>  <p>逐渐将 border-bottom-width 属性从 1px 改为 50px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 15px solid coral;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-color: lightblue;}  }  </style>  </head>  <body>  <h1>border-color 的动画效果：</h1>  <p>逐渐将border-color属性从珊瑚更改为淡蓝色，然后返回：<p>  <div id="myDIV"></div>  逐渐将 border-color 属性从珊瑚色更改为淡蓝色，然后再：  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-left | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-left: 15px solid lightblue;}  }  </style>  </head>  <body>  <h1>border-left 的动画效果：</h1>  <p>逐渐更改 border-left 属性：<p>  <div id="myDIV"></div>  <p>border-left 是所有 border-left 属性的简写属性。</p>  <p>请查看单个的 border-left 的属性，来了解哪些是<em>支持动画制作的</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-left-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 15px solid coral;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-left-color: lightblue;}  }  </style>  </head>  <body>  <h1>border-left-color 的动画效果：</h1>  <p>逐渐将 border-left-color 属性从珊瑚色更改为淡蓝色，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-left-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-left-width: 15px;}  }  </style>  </head>  <body>  <h1>border-left-width 的动画效果：</h1>  <p>逐渐将 border-left-width 属性从 1px 更改为 15px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-right | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-right: 15px solid lightblue;}  }  </style>  </head>  <body>  <h1>border-right 的动画效果：</h1>  <p>逐渐更改 border-right 属性：<p>  <div id="myDIV"></div>  <p>border-right 是所有 border-right 属性的简写属性。</p>  <p>请查看单个的 border-right 的属性，来了解哪些是<em>支持动画制作的</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-right-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 15px solid coral;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-right-color: lightblue;}  }  </style>  </head>  <body>  <h1>border-right-color 的动画效果：</h1>  <p>逐渐将 border-right-color 属性从珊瑚色更改为浅蓝色，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-right-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-right-width: 15px;}  }  </style>  </head>  <body>  <h1>border-right-width 的动画效果：</h1>  <p>逐渐将 border-right-width 属性从 1px 改为 15px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-spacing | <!DOCTYPE html>  <html>  <head>  <style>  table, th, td {  border: 1px solid black;  }  #myTABLE {  border-spacing: 2px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-spacing: 20px;}  }  </style>  </head>  <body>  <h1>border-spacing 的动画效果：</h1>  <p>逐渐将 border-spacing 属性从 2px 更改为 20px，然后再改回：<p>  <table id="myTABLE">  <tr>  <th>Name</th>  <th>Age</th>  </tr>  <tr>  <td>Emil</td>  <td>9</td>  </tr>  <tr>  <td>Tobias</td>  <td>7</td>  </tr>  <tr>  <td>Linus</td>  <td>2</td>  </tr>  </table>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-top | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-top: 15px solid lightblue;}  }  </style>  </head>  <body>  <h1>border-top 的动画效果：</h1>  <p>逐渐更改border-top属性：<p>  <div id="myDIV"></div>  <p>border-top 是所有 border-top 属性的简写属性。</p>  <p>请查看单个的 border-top 的属性，来了解哪些是<em>支持动画制作的</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-top-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 15px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-top-color: lightblue;}  }  </style>  </head>  <body>  <h1>border-top-color 的动画效果：</h1>  <p>逐渐将 border-top-color 属性从黑色更改为蓝色，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-top-left-radius | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-top-left-radius: 150px;}  }  </style>  </head>  <body>  <h1>border-top-left-radius 的动画效果：</h1>  <p>逐渐将 border-top-left-radius 属性从 1px 更改为 150px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-top-right-radius | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-top-right-radius: 150px;}  }  </style>  </head>  <body>  <h1>border-top-right-radius 的动画效果：</h1>  <p>逐渐将 border-top-right-radius 属性从 1px 更改为 150px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| border-top-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {border-top-width: 15px;}  }  </style>  </head>  <body>  <h1>border-top-width 的动画效果：</h1>  <p>逐渐将 border-top-width 属性从 1px 更改为 15px，然后再改回：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| bottom | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  position: absolute;  width: 100px;  height: 100px;  background-color: coral;  color: white;  bottom: 0px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {bottom: 300px;}  }  </style>  </head>  <body>  <h1>bottom 的动画效果：</h1>  <p>逐渐将 bottom 属性从 0px 更改为 300px，然后再改回：<p>  <div id="myDIV">  <h1>myDIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| box-shadow | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 100px;  height: 100px;  background-color: coral;  color: white;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {box-shadow: 10px 20px 30px blue;}  }  </style>  </head>  <body>  <h1>box-shadow 的动画效果：</h1>  <p>逐渐更改 box-shadow 属性：<p>  <div id="myDIV">  <h1>myDIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| clip | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 100px;  height: 100px;  background-color: coral;  color: green;  position: absolute;  animation: mymove 5s infinite;  clip: rect(0,100px,100px,0);  }  @keyframes mymove {  50% {clip: rect(0,50px,50px,0);}  }  </style>  </head>  <body>  <h1>clip 的动画效果：</h1>  <div id="myDIV">  <h1>myDIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  width: 300px;  color: red;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {color: blue;}  }  </style>  </head>  <body>  <h1>color 的动画效果：</h1>  <p>逐渐将文本的颜色从红色更改为蓝色，然后再改回：<p>  <div id="myDIV">  <h1>这是一个标题</h1>  <p>这是一个段落。</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| column-count | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  width: 500px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {column-count: 5;}  }  </style>  </head>  <body>  <h1>column-count 的动画效果：</h1>  <p>将列数从 1 改为 5，然后再改回：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>Edge 不支持 column-count 属性上的动画。</p>  </body>  </html> |
| column-gap | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  column-count: 3;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {column-gap: 100px;}  }  </style>  </head>  <body>  <h1>column-gap 的动画效果：</h1>  <p>将列之间的间隔从正常更改为 100px，然后再恢复为正常：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>Edge 不支持 column-gap 属性上的动画。</p>  </body>  </html> |
| column-rule | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  column-count: 3;  column-rule: 1px outset red;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {  column-rule: 50px outset blue;  }  }  </style>  </head>  <body>  <h1>column-rule 的动画效果：</h1>  <p>逐渐更改列间规则：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p>column-rule 属性是所有 column-rule 属性的简写属性。</p>  <p>在 CSS 中，column-rule-color 和 column-rule-width 属性<em>支持动画制作</em>。</p>  <p><b>注释：</b>Edge 不支持 column-rule 属性上的动画。</p>  </body>  </html> |
| column-rule-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  column-count: 3;  column-rule: 15px outset red;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {  column-rule-color: blue;  }  }  </style>  </head>  <body>  <h1>column-rule-color 的动画效果：</h1>  <p>逐渐将列间规则的颜色从红色更改为蓝色，然后再改回为红色：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>Edge 不支持 column-rule-color 属性上的动画。</p>  </body>  </html> |
| column-rule-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  column-count: 3;  column-rule: 1px outset red;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {  column-rule-width: 50px;  }  }  </style>  </head>  <body>  <h1>column-rule-width 的动画效果：</h1>  <p>逐渐将列间规则的宽度从 1px 更改为 50px，然后再改回 1px：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>Edge 不支持 column-rule-width 属性上的动画。</p>  </body>  </html> |
| column-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  column-width: 300px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {  column-width: 100px;  }  }  </style>  </head>  <body>  <h1>column-width 的动画效果：</h1>  <p>逐渐将列的宽度从 300px 更改为 100px，然后再改回 300px：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>Edge 不支持 column-width 属性上的动画。</p>  </body>  </html>  一试 |
| columns | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {  columns: 100px 5;  }  }  </style>  </head>  <body>  <h1>columns 的动画效果：</h1>  <p>逐渐更改 columns 属性：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p>columns 属性是用于设置 column-count 和 column-width 属性的简写属性。</p>  <p><b>注释：</b>Edge 不支持 column 属性上的动画。</p>  </body>  </html> |
| filter | <!DOCTYPE html>  <html>  <head>  <style>  img {  animation: mymove 7s infinite;  }  @keyframes mymove {  50% {  -webkit-filter: grayscale(100%);  filter: grayscale(100%);  }  }  </style>  </head>  <body>  <h1>filter 的动画效果：</h1>  <p>逐渐将图像的颜色更改为黑白（100％ 灰度），然后恢复为原始颜色：<p>  <img src="/i/photo/tulip.jpg" alt="tulip" width="300" height="300">  <p><b>注释：</b>Edge 12 或 Internet Explorer 不支持 filter 属性。</p>  </body>  </html> |
| flex | <!DOCTYPE html>  <html>  <head>  <style>  #main {  width: 270px;  height: 300px;  border: 1px solid black;  display: flex;  }  #main div {  -ms-flex: 1; /\* IE 10 \*/  flex: 1;  }  #myBlueDiv {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {flex: 3;}  }  </style>  </head>  <body>  <h1>flex 的动画效果：</h1>  <p>Gradually change the flex property of the BLUE div element from 1 to 3, and back:<p>  <div id="main">  <div style="background-color:coral;">RED</div>  <div style="background-color:lightblue;" id="myBlueDiv">BLUE</div>  <div style="background-color:lightgreen;">GREEN</div>  </div>  <p><b>注释：</b> Internet Explorer 10 supports an alternate, the -ms-flex property. IE11 and newer versions fully support the flex property (do not need the -ms- prefix).</p>  </body>  </html> |
| flex-basis | <!DOCTYPE html>  <html>  <head>  <style>  #main {  width: 250px;  height: 100px;  border: 1px solid #c3c3c3;  display: flex;  }  #main div {  flex-grow: 0;  flex-shrink: 0;  flex-basis: 50px;  }  #myBlueDiv {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {flex-basis: 200px;}  }  </style>  </head>  <body>  <h1>flex-basis 的动画效果：</h1>  <p>Gradually change the flex-basis property of the BLUE div from 50px to 200px, and back:<p>  <div id="main">  <div style="background-color:coral;">RED</div>  <div style="background-color:lightblue;" id="myBlueDiv">BLUE</div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| flex-grow | <!DOCTYPE html>  <html>  <head>  <style>  #main {  width: 350px;  height: 100px;  border: 1px solid #c3c3c3;  display: flex;  }  #main div:nth-of-type(1) {flex-grow: 1;}  #main div:nth-of-type(2) {flex-grow: 1;}  #main div:nth-of-type(3) {flex-grow: 1;}  #myBlueDiv {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {flex-grow: 8;}  }  </style>  </head>  <body>  <h1>flex-grow 的动画效果：</h1>  <p>Gradually change the flex-grow property of the "blue DIV" from 1 to 8, and back to 1:<p>  <div id="main">  <div style="background-color:coral;"></div>  <div style="background-color:lightblue;" id="myBlueDiv"></div>  <div style="background-color:khaki;"></div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| flex-shrink | <!DOCTYPE html>  <html>  <head>  <style>  #main {  width: 350px;  height: 100px;  border: 1px solid #c3c3c3;  display: flex;  }  #main div {  flex-grow: 1;  flex-shrink: 1;  flex-basis: 300px;  }  #myBlueDiv {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {flex-shrink: 8;}  }  </style>  </head>  <body>  <h1>flex-shrink 的动画效果：</h1>  <p>Gradually change the flex-shrink property of the "blue DIV" from 1 to 8, and back to 1:<p>  <div id="main">  <div style="background-color:coral;">Red DIV</div>  <div style="background-color:lightblue;" id="myBlueDiv">Blue DIV</div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| font | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  width: 400px;  height: 100px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {font: 40px bold;}  }  </style>  </head>  <body>  <h1>font 的动画效果：</h1>  <p>逐渐更改字体属性：<p>  <div id="myDIV">  <p>This is a paragraph</p>  </div>  <p>font 属性是所有字体属性的简写属性。</p>  <p>在 CSS 中，font-size、font-weight、font-stretch 以及 line-height <em>支持动画制作</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| font-size | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  width: 400px;  height: 100px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {font-size: 40px;}  }  </style>  </head>  <body>  <h1>font-size 的动画效果：</h1>  <p>逐渐更改文本的字体大小：<p>  <div id="myDIV">  <p>This is a paragraph</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| font-size-adjust |  |
| font-stretch |  |
| font-weight | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  width: 400px;  height: 100px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {font-weight: bold;}  }  </style>  </head>  <body>  <h1>font-weight 的动画效果：</h1>  <p>逐渐将字体粗细从正常更改为粗体，然后再恢复为正常：<p>  <div id="myDIV">  <p>This is a paragraph</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| grid | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid: auto / auto auto auto auto;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid: 100px / auto auto;}  }  </style>  </head>  <body>  <h1>grid 的动画效果：</h1>  <p>该动画将更改网格布局。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  <div class="item8">8</div>  </div>  </body>  </html> |
| grid-area | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  grid-area: 2 / 1 / span 2 / span 3;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-area: 1 / 2 / span 3 / span 2;}  }  </style>  </head>  <body>  <h1>grid-area 的动画效果：</h1>  <p>该动画将更改 "item1" 的外观。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-auto-columns | <!DOCTYPE html>  <html>  <head>  <style>  .item1 { grid-area: 1 / 1 / 2 / 2; }  .item2 { grid-area: 1 / 2 / 2 / 3; }  .item3 { grid-area: 1 / 3 / 2 / 4; }  .item4 { grid-area: 2 / 1 / 3 / 2; }  .item5 { grid-area: 2 / 2 / 3 / 3; }  .item6 { grid-area: 2 / 3 / 3 / 4; }  .grid-container {  display: grid;  grid-auto-columns: 50px;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-auto-columns: 100px;}  }  </style>  </head>  <body>  <h1>grid-auto-columns 的动画效果：</h1>  <p>动画会将列的尺寸从 50px 改为 100px。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  </div>  </body>  </html> |
| grid-auto-flow | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto;  grid-template-rows: auto auto;  grid-auto-flow: row;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-auto-flow: column;}  }  </style>  </head>  <body>  <h1>grid-auto-flow 的动画效果：</h1>  <p>该动画将 grid-auto-flow 从行改为列。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  </div>  </body>  </html> |
| grid-auto-rows | <!DOCTYPE html>  <html>  <head>  <style>  .item1 { grid-area: 1 / 1 / 2 / 2; }  .item2 { grid-area: 1 / 2 / 2 / 3; }  .item3 { grid-area: 1 / 3 / 2 / 4; }  .item4 { grid-area: 2 / 1 / 3 / 2; }  .item5 { grid-area: 2 / 2 / 3 / 3; }  .item6 { grid-area: 2 / 3 / 3 / 4; }  .grid-container {  display: grid;  grid-auto-rows: 150px;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-auto-rows: 100px;}  }  </style>  </head>  <body>  <h1>grid-auto-rows 的动画效果：</h1>  <p>该动画会将行的尺寸从 150px 改为 100px。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  </div>  </body>  </html> |
| grid-column | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  grid-column: 1 / span 2;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-column: 3 / span 3;}  }  </style>  </head>  <body>  <h1>grid-column 的动画效果：</h1>  <p>该动画将更改 "item1" 的外观。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-column-end | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  grid-column-end: span 2;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-column-end: span 3;}  }  </style>  </head>  <body>  <h1>grid-column-end 的动画效果：</h1>  <p>该动画将更改 "item1" 的外观。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-column-gap | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-column-gap: 100px;}  }  </style>  </head>  <body>  <h1>grid-column-gap 的动画效果：</h1>  <p>该动画会将列间隔从 10px 改为 100px。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-column-start | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-column-start: 3;}  }  </style>  </head>  <body>  <h1>grid-column-start 的动画效果：</h1>  <p>该动画将更改 "item1" 的位置。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-gap | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-gap: 100px;}  }  </style>  </head>  <body>  <h1>grid-gap 的动画效果：</h1>  <p>该动画会把行间隙和列间隙从 10px 改为 100px。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-row | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  grid-row: 1 / span 2;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-row: 1 / span 3;}  }  </style>  </head>  <body>  <h1>grid-row 的动画效果：</h1>  <p>该动画将更改 "item1" 的外观。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html>  <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  grid-row: 1 / span 2;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-row: 1 / span 3;}  }  </style>  </head>  <body>  <h1>grid-row 的动画效果：</h1>  <p>该动画将更改 "item1" 的外观。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-row-end | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  grid-row-end: span 2;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-row-end: span 3;}  }  </style>  </head>  <body>  <h1>grid-row-end 的动画效果：</h1>  <p>该动画将更改 "item1" 的外观。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-row-gap | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-row-gap: 100px;}  }  </style>  </head>  <body>  <h1>grid-row-gap 的动画效果：</h1>  <p>该动画会把行间隔从 10px 更改为 100px。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-row-start | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  .item1 {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {grid-row-start: 3;}  }  </style>  </head>  <body>  <h1>grid-row-start 的动画效果：</h1>  <p>该动画将更改 "item1" 的外观。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  </div>  </body>  </html> |
| grid-template | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template: auto / auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-template: 100px / auto auto;}  }  </style>  </head>  <body>  <h1>grid-template 的动画效果：</h1>  <p>该动画将更改网格布局。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  <div class="item8">8</div>  </div>  </body>  </html> |
| grid-template-areas | <!DOCTYPE html>  <html>  <head>  <style>  .item1 {  grid-area: myArea;  }  .grid-container {  display: grid;  grid-template-areas: 'myArea myArea . . .';  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-template-areas: '. . . myArea myArea';}  }  </style>  </head>  <body>  <h1>grid-template-areas 的动画效果：</h1>  <p>该动画将更改 "item1" 的位置。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  <div class="item8">8</div>  <div class="item9">9</div>  </div>  </body>  </html> |
| grid-template-columns | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-template-columns: auto auto;}  }  </style>  </head>  <body>  <h1>grid-template-columns 的动画效果：</h1>  <p>动画将列数从四改为二。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  <div class="item8">8</div>  </div>  </body>  </html> |
| grid-template-rows | <!DOCTYPE html>  <html>  <head>  <style>  .grid-container {  display: grid;  grid-template-columns: auto auto auto auto;  grid-gap: 10px;  background-color: #2196F3;  padding: 10px;  animation: mymove 5s infinite;  }  .grid-container > div {  background-color: rgba(255, 255, 255, 0.8);  text-align: center;  padding: 20px 0;  font-size: 30px;  }  @keyframes mymove {  50% {grid-template-rows: 100px;}  }  </style>  </head>  <body>  <h1>grid-template-rows 的动画效果：</h1>  <p>该动画会将第一行的尺寸改为 100px，然后再改回。</p>  <div class="grid-container">  <div class="item1">1</div>  <div class="item2">2</div>  <div class="item3">3</div>  <div class="item4">4</div>  <div class="item5">5</div>  <div class="item6">6</div>  <div class="item7">7</div>  <div class="item8">8</div>  </div>  </body>  </html> |
| height | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  height: 100px;  background-color: coral;  color: white;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {height: 500px;}  }  </style>  </head>  <body>  <h1>height 的动画效果：</h1>  <p>逐步将 div 元素的高度从 100px 改为 500px，然后再改回 100px：<p>  <div id="myDIV">  <h1>My DIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| left | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  position: absolute;  left: 0;  width: 100px;  height: 100px;  background-color: coral;  color: white;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {left: 500px;}  }  </style>  </head>  <body>  <h1>left 的动画效果：</h1>  <p>逐渐把 div 元素的左侧位置从 0 改500px，然后再改回 0：<p>  <div id="myDIV">  <h1>My DIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| letter-spacing | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  letter-spacing: 2px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {letter-spacing: 50px;}  }  </style>  </head>  <body>  <h1>letter-spacing 的动画效果：</h1>  <p>逐渐将 div 元素的字母间距属性从 2px 改为 50px，然后再改回 2px：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| line-height | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  line-height: 16px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {line-height: 50px;}  }  </style>  </head>  <body>  <h1>line-height 的动画效果：</h1>  <p>逐渐将 div 元素的 line-height 属性从 16px 改为 50px，然后再改回 16px：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| margin | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {margin: 50px;}  }  </style>  </head>  <body>  <h1>margin 的动画效果：</h1>  <p>逐渐将外边距从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  This is my DIV element.  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| margin-bottom | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {margin-bottom: 50px;}  }  </style>  </head>  <body>  <h1>margin-bottom 的动画效果：</h1>  <p>逐渐将下外边距从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  This is my DIV element.  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| margin-left | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {margin-left: 50px;}  }  </style>  </head>  <body>  <h1>margin-left 的动画效果：</h1>  <p>逐渐将左外边距从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  This is my DIV element.  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| margin-right | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {margin-right: 50px;}  }  </style>  </head>  <body>  <h1>margin-right 的动画效果：</h1>  <p>逐渐将右外边距从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  This is my DIV element.  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| margin-top | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {margin-top: 50px;}  }  </style>  </head>  <body>  <h1>margin-top 的动画效果：</h1>  <p>逐渐将上外边距从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  This is my DIV element.  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| max-height | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 500px;  background-color: lightblue;  overflow: auto;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {max-height: 100px;}  }  </style>  </head>  <body>  <h1>max-height 的动画效果：</h1>  <p>将最大高度从 "none" 更改为 "100px"，然后再改回 "none"：<p>  <div id="myDIV">  <p>该 div 元素没有预定义的高度。</p>  <p>动画会将最大高度从 "none" 更改为 "100px"。</p>  <p>该 div 元素的高度将不超过 100 个像素。</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| max-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  overflow: auto;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {max-width: 600px;}  }  </style>  </head>  <body>  <h1>max-width 的动画效果：</h1>  <p>Change the max-width from "none" to "600px", and back to "none":<p>  <div id="myDIV">  <p>该 div 元素没有预定义的宽度。</p>  <p>尝试调整浏览器窗口的大小，来查看这个 div 元素的宽度将随之扩展或收缩，但宽度永远不会超过 600 像素。</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| min-height | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 500px;  background-color: lightblue;  overflow: auto;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {min-height: 400px;}  }  </style>  </head>  <body>  <h1>min-height 的动画效果：</h1>  <p>将最小高度从 "none" 更改为 "400px"，然后再改会 "none"：<p>  <div id="myDIV">  <p>该 div 元素没有预定义的高度。</p>  <p>动画会将最小高度从 "none" 更改为 "400px"。</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| min-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 50%;  background-color: lightblue;  overflow: auto;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {min-width: 800px;}  }  </style>  </head>  <body>  <h1>min-width 的动画效果：</h1>  <p>将最小宽度从 "none" 改为 800px，然后再改回 "none"：<p>  <p><b>注释：</b>min-width 属性会覆盖 width 属性。</p>  <div id="myDIV">  <p>该 div 元素的预定义宽度是：50％。</p>  <p>动画将逐渐将最小宽度更改为 800 像素。</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| object-position | <!DOCTYPE html>  <html>  <head>  <style>  #myImg {  width: 200px;  height: 400px;  object-fit: cover;  object-position: 0% 0%;  animation: mymove 5s infinite;  }  @keyframes mymove {  0% {  object-position: 0% 0%;  }  25% {  object-position: 20% 0%;  }  100% {  object-position: 100% 100%;  }  }  </style>  </head>  <body>  <h1>object-position 的动画效果：</h1>  <p>逐渐更改图像在其内容框中的位置：<p>  <img src="/i/photo/tulip.jpg" alt="Tulip" id="myImg" width="400" height="300">  <p>注释：Internet Explorer/Edge 15 或更早的版本不支持 object-position 和 object-fit 属性。</p>  </body>  </html> |
| opacity | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  position: absolute;  top: 200px;  width: 300px;  height: 150px;  background-color: lightblue;  border: 1px solid black;  animation: mymove 5s infinite;  }  #DIV2 {  position: absolute;  top: 180px;  left: 30px;  width: 300px;  height: 150px;  background-color: coral;  border: 1px solid black;  }  @keyframes mymove {  50% {opacity: 0;}  }  </style>  </head>  <body>  <h1>opacity 的动画效果：</h1>  <p>将蓝色 div 元素的不透明度从 1 更改为 0，然后再改回 1：<p>  <div id="DIV2">  <h1>Voila!</h1>  </div>  <div id="myDIV">  </div>  <p>在 CSS 中，opacity 属性<em>支持动画支持</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| order | <!DOCTYPE html>  <html>  <head>  <style>  #main {  width: 400px;  height: 150px;  border: 1px solid #000000;  display: flex;  }  #main div {  width: 70px;  height: 70px;  }  #myGreenDIV {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {order: 6;}  }  </style>  </head>  <body>  <h1>order 的动画效果：</h1>  <p>将绿色 div 元素的顺序从 1 更改为 6，然后改回 1：<p>  <div id="main">  <div style="background-color:lightgreen;order:1;" id="myGreenDIV"></div>  <div style="background-color:coral;order:2;"></div>  <div style="background-color:lightblue;order:3;"></div>  <div style="background-color:pink;order:4;"></div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| outline | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  outline: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {outline: 15px solid lightblue;}  }  </style>  </head>  <body>  <h1>outline 的动画效果：</h1>  <p>逐渐更改 outline 属性：<p>  <div id="myDIV"></div>  <p>outline 属性是所有轮廓属性的简写属性。</p>  <p>请参阅单个轮廓属性，来查看哪个属性<em>支持动画制作</em>。</p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| outline-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 300px;  height: 200px;  outline: 15px solid red;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {outline-color: blue;}  }  </style>  </head>  <body>  <h1>outline-color 的动画效果：</h1>  <p>逐渐将 outline-color 属性从红色改为蓝色，然后再改回红色：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| outline-offset | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  margin: auto;  border: 1px solid black;  outline: coral solid 5px;  width: 300px;  height: 300px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {outline-offset: 50px;}  }  </style>  </head>  <body>  <h1>outline-offset 的动画效果：</h1>  <p>逐渐将 outline-offset 属性从 0 改为 50px，然后再改回 0：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| outline-width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  margin: auto;  border: 1px solid black;  outline: coral solid 5px;  width: 300px;  height: 300px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {outline-width: 50px;}  }  </style>  </head>  <body>  <h1>outline-width 的动画效果：</h1>  <p>逐渐将 outline-width 属性从 5px 更改为 50px，然后再改回 5px：<p>  <div id="myDIV"></div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| padding | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {padding: 50px;}  }  </style>  </head>  <body>  <h1>padding 的动画效果：</h1>  <p>逐渐将内边距从 0px 更改为 50px，然后再改回 0px：<p>  <div id="myDIV">  Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai! ...  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| padding-bottom | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {padding-bottom: 50px;}  }  </style>  </head>  <body>  <h1>padding-bottom 的动画效果：</h1>  <p>逐渐将下内边距从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai! ...  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| padding-left | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {padding-left: 50px;}  }  </style>  </head>  <body>  <h1>padding-left 的动画效果：</h1>  <p>逐渐将 padding-left 从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!...  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| padding-right | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {padding-right: 150px;}  }  </style>  </head>  <body>  <h1>padding-right 的动画效果：</h1>  <p>逐渐将右内边距从 0px 改为 150px，然后再改回 0px：<p>  <div id="myDIV">  Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai! ...  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| padding-top | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {padding-top: 50px;}  }  </style>  </head>  <body>  <h1>padding-top 的动画效果：</h1>  <p>逐渐将上内边距从 0px 改为 50px，然后再改回 0px：<p>  <div id="myDIV">  Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai! ...  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| perspective | <!DOCTYPE html>  <html>  <head>  <style>  #div1 {  position: relative;  margin: auto;  height: 150px;  width: 250px;  padding: 10px;  border: 1px solid black;  animation: mymove 5s infinite;  perspective: 200px;  }  @keyframes mymove {  50% {perspective: 100px;}  }  #div2 {  padding: 50px;  position: absolute;  border: 1px solid black;  background-color: red;  transform: rotateX(45deg);  }  </style>  </head>  <body>  <h1>perspective 的动画效果：</h1>  <div id="div1">DIV1  <div id="div2">请想象您正在墙上向下看。看看逐渐改变视角会发生什么！</div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| perspective-origin | <!DOCTYPE html>  <html>  <head>  <style>  #div1 {  position: relative;  margin: auto;  height: 150px;  width: 250px;  padding: 10px;  border: 1px solid black;  perspective: 125px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {perspective-origin: 10px 50%;}  }  #div2 {  padding: 50px;  position: absolute;  border: 1px solid black;  background-color: coral;  transform: rotateX(45deg);  }  </style>  </head>  <body>  <h1>perspective-origin 的动画效果：</h1>  <div id="div1">DIV1  <div id="div2">DIV2</div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| right | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  position: absolute;  right: 0;  width: 100px;  height: 100px;  background-color: coral;  color: white;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {right: 500px;}  }  </style>  </head>  <body>  <h1>right 的动画效果：</h1>  <p>逐渐将 DIV 元素的右侧位置从 0 改为 500px，然后再改回 0：<p>  <div id="myDIV">  <h1>My DIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| text-decoration-color | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  text-decoration: underline;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {  text-decoration-color: red;  }  }  </style>  </head>  <body>  <h1>text-decoration-color 的动画效果：</h1>  <p>逐渐将下划线的颜色从黑色更改为红色，然后再更回为：<p>  <div id="myDIV">  <p>This is a paragraph</p>  </div>  <p><b>注释：</b>text-decoration-color 在版本 79 之前的 Edge 中无效。</p>  </body>  </html> |
| text-indent | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  background-color: lightblue;  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {text-indent: 150px;}  }  </style>  </head>  <body>  <h1>text-indent 的动画效果：</h1>  <p>逐渐将文本缩进从 0 改为 150px，然后再改回 0：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| text-shadow | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {text-shadow: 10px 20px 30px blue;}  }  </style>  </head>  <body>  <h1>text-shadow 的动画效果：</h1>  <p>逐渐更改 text-shadow 属性：<p>  <div id="myDIV">  <h1>This is a header</h1>  <p>This is a paragraph</p>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| top | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  position: absolute;  width: 100px;  height: 100px;  background-color: coral;  color: white;  top: 0;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {top: 300px;}  }  </style>  </head>  <body>  <h1>top 的动画效果：</h1>  <p>逐渐将 top 属性从 0 改为 300px，然后再改回 0：<p>  <div id="myDIV">  <h1>myDIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| transform | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  margin: auto;  border: 1px solid black;  width: 200px;  height: 100px;  background-color: coral;  color: white;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {transform: rotate(180deg);}  }  </style>  </head>  <body>  <h1>transform 的动画效果：</h1>  <p>逐渐将 div 元素旋转 180 度，然后转回来：<p>  <div id="myDIV">  <h1>myDIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| transform-origin | <!DOCTYPE html>  <html>  <head>  <style>  #DIV1 {  height: 200px;  width: 200px;  margin: auto;  border: 1px solid black;  }  #DIV2 {  width: 150px;  height: 150px;  border: 1px solid black;  background-color: coral;  -ms-transform: rotate(45deg); /\* IE 9 \*/  transform: rotate(45deg);  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {transform-origin: 0 0 0;}  }  #DIV2original {  position: absolute;  width: 150px;  height: 150px;  border: 1px dashed grey;  background-color: lightgrey;  opacity: 0.5;  }  </style>  </head>  <body>  <h1>transform-origin 的动画效果：</h1>  <p>逐渐将 transform-origin 从 "50% 50% 0" 改为 "0 0 0"，然后再改回：<p>  <p><b>注释：</b>这个灰色的 DIV 元素指示 DIV2 元素在不进行转换的情况下的位置。</p>  <div id="DIV1">DIV1  <div id="DIV2original">DIV2</div> <div id="DIV2">DIV2</div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| vertical-align | <!DOCTYPE html>  <html>  <head>  <style>  #myIMG {  vertical-align: 50px;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {vertical-align: 100px;}  }  </style>  </head>  <body>  <h1>vertical-align 的动画效果：</h1>  <p>逐渐将图像元素的 vertical-align 属性从 0 改为 100px，然后再改回 0：<p>  <p><img id="myIMG" src="smiley.gif" width="32" height="32">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </p>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| visibility |  |
| width | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  width: 120px;  background-color: coral;  color: white;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {width: 500px;}  }  </style>  </head>  <body>  <h1>width 的动画效果：</h1>  <p>逐渐将 DIV 元素的宽度从 120px 更改为 500px，然后再改回 120px：<p>  <div id="myDIV">  <h1>My DIV</h1>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| word-spacing | <!DOCTYPE html>  <html>  <head>  <style>  #myDIV {  border: 1px solid black;  animation: mymove 5s infinite;  }  @keyframes mymove {  50% {word-spacing: 50px;}  }  </style>  </head>  <body>  <h1>word-spacing 的动画效果：</h1>  <p>逐渐将 div 元素的字距从“普通”更改为 50px，然后再改回“普通”：<p>  <div id="myDIV">  话说天下大势，分久必合，合久必分：周末七国分争，并入于秦；及秦灭之后，楚、汉分争，又并入于汉；汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，传至献帝，遂分为三国。推其致乱之由，殆始于桓、灵二帝。桓帝禁锢善类，崇信宦官。及桓帝崩，灵帝即位，大将军窦武、太傅陈蕃，共相辅佐；时有宦官曹节等弄权，窦武、陈蕃谋诛之，机事不密，反为所害，中涓自此愈横。建宁二年四月望日，帝御温德殿。方升座，殿角狂风骤起，只见一条大青蛇，从梁上飞将下来，蟠于椅上。帝惊倒，左右急救入宫，百官俱奔避。须臾，蛇不见了。忽然大雷大雨，加以冰雹，落到半夜方止，坏却房屋无数。建宁四年二月，洛阳地震；又海水泛溢，沿海居民，尽被大浪卷入海中。光和元年，雌鸡化雄。六月朔，黑气十馀丈，飞入温德殿中。秋七月，有虹现于玉堂，五原山岸，尽皆崩裂。种种不祥，非止一端。帝下诏问群臣以灾异之由，议郎蔡邕上疏，以为蜺堕鸡化，乃妇寺干政之所致，言颇切直。帝览奏叹息，因起更衣。曹节在后窃视，悉宣告左右；遂以他事陷邕于罪，放归田里。后张让、赵忠、封谞、段珪、曹节、侯览、蹇硕、程旷、夏恽、郭胜十人朋比为奸，号为“十常侍”。帝尊信张让，呼为“阿父”。朝政日非，以致天下人心思乱，盗贼蜂起。  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |
| z-index | <!DOCTYPE html>  <html>  <head>  <style>  div {position: absolute;}  #container div {  background-color: lightblue;  border: 1px solid #333333;  width: 100px;  height: 100px;  opacity: 0.5;  }  div#myBox {  opacity: 1;  background-color: coral;  z-index: 1;  animation: mymove 5s infinite linear;  }  @keyframes mymove {  50% {z-index: 5;}  }  </style>  </head>  <body style="position:absolute">  <h1>z-index 的动画效果：</h1>  <p>逐渐将 "myBox" 的 z-index 属性从 1 改为 5，然后再改回 1：<p>  <div id="container">  <div id="myBox">myBox</div>  <div style="top:20px;left:20px;z-index:1;">z-index 1</div>  <div style="top:40px;left:40px;z-index:2;">z-index 2</div>  <div style="top:60px;left:60px;z-index:3;">z-index 3</div>  <div style="top:80px;left:80px;z-index:4;">z-index 4</div>  </div>  <p><b>注释：</b>CSS 动画在 Internet Explorer 9 以及更早版本中不起作用。</p>  </body>  </html> |

# 第114章 CSS 网络安全字体

## 114.1 适用于 HTML 和 CSS 的最佳 Web 安全字体

下面列出了适用于 HTM L和 CSS 的最佳 Web 安全字体：

Arial (sans-serif)

Verdana (sans-serif)

Helvetica (sans-serif)

Tahoma (sans-serif)

Trebuchet MS (sans-serif)

Times New Roman (serif)

Georgia (serif)

Garamond (serif)

Courier New (monospace)

Brush Script MT (cursive)

注意：发布网站之前，请您始终检查字体在不同浏览器和设备上的显示效果，并始终使用后备字体！

# 第115章 CSS 字体回退

常用的后备字体

以下是按 5 个通用字体系列组织的一些常用字体后备：

Serif

Sans-serif

Monospace

Cursive

Fantasy

# 第116章 CSS 单位

## 116.1 CSS 单位

CSS 有几种表示长度的不同单位。

许多 CSS 属性接受“长度”值，诸如 width、margin、padding、font-size 等。

长度是一个后面跟着长度单位的数字，诸如 10px、2em 等。

实例

使用 px（像素）设置不同的长度值：

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

font-size: 60px;

}

p {

font-size: 25px;

line-height: 50px;

}

</style>

</head>

<body>

<h1>这是标题 1</h1>

<h2>这是标题 2</h2>

<p>这是一个段落。</p>

<p>这是另一个段落。</p>

</body>

</html>

数字和单位之间不能出现空格。但是，如果值为 0，则可以省略单位。

对于某些 CSS 属性，允许使用负的长度。

长度单位有两种类型：绝对单位和相对单位。

## 116.2 绝对长度

绝对长度单位是固定的，用任何一个绝对长度表示的长度都将恰好显示为这个尺寸。

不建议在屏幕上使用绝对长度单位，因为屏幕尺寸变化很大。但是，如果已知输出介质，则可以使用它们，例如用于打印布局（print layout）。

|  |  |  |
| --- | --- | --- |
| 单位 | 描述 | TIY |
| cm | 厘米 | 试一试 |
| mm | 毫米 | 试一试 |
| in | 英寸 (1in = 96px = 2.54cm) | 试一试 |
| px \* | 像素 (1px = 1/96th of 1in) | 试一试 |
| pt | 点 (1pt = 1/72 of 1in) | 试一试 |
| pc | 派卡 (1pc = 12 pt) | 试一试 |

\* 像素（px）是相对于观看设备的。对于低 dpi 的设备，1px 是显示器的一个设备像素（点）。对于打印机和高分辨率屏幕，1px 表示多个设备像素。

## 116.3 相对长度

相对长度单位规定相对于另一个长度属性的长度。相对长度单位在不同渲染介质之间缩放表现得更好。

|  |  |  |
| --- | --- | --- |
| 单位 | 描述 | TIY |
| em | 相对于元素的字体大小（font-size）（2em 表示当前字体大小的 2 倍） | 试一试 |
| ex | 相对于当前字体的 x-height(极少使用) | 试一试 |
| ch | 相对于 "0"（零）的宽度 | 试一试 |
| rem | 相对于根元素的字体大小（font-size） | 试一试 |
| vw | 相对于视口\*宽度的 1% | 试一试 |
| vh | 相对于视口\*高度的 1% | 试一试 |
| vmin | 相对于视口\*较小尺寸的 1％ | 试一试 |
| vmax | 相对于视口\*较大尺寸的 1％ | 试一试 |
| % | 相对于父元素 | 试一试 |

提示：em 和 rem 单元可用于创建完美的可扩展布局！

\* 视口（Viewport）= 浏览器窗口的尺寸。如果视口宽 50 里面，则 1vw = 0.5cm。

# 第117章 CSS 合法颜色值

## 117.1 CSS 颜色

CSS 中的颜色可以通过以下方法指定：

十六进制颜色

带透明度的十六进制颜色

RGB 颜色

RGBA 颜色

HSL 颜色

HSLA 颜色

预定义/跨浏览器的颜色名称

使用 currentcolor 关键字

## 117.2 十六进制颜色

用 #RRGGBB 规定十六进制颜色，其中 RR（红色）、GG（绿色）和 BB（蓝色）十六进制整数指定颜色的成分（分量）。所有值必须在 00 到 FF 之间。

例如，＃0000ff 值呈现为蓝色，因为蓝色分量设置为最高值（ff），其他分量设置为 00。

实例

定义不同的 HEX 颜色：

#p1 {background-color: #ff0000;} /\* 红色 \*/

#p2 {background-color: #00ff00;} /\* 绿色 \*/

#p3 {background-color: #0000ff;} /\* 蓝色 \*/

## 117.3 带透明度的十六进制颜色

用 #RRGGBB 规定十六进制颜色。如需增加透明度，请在 00 和 FF 之间添加两个额外的数字。

实例

定义带透明度的 HEX 颜色：

#p1a {background-color: #ff000080;} /\* 带透明度的红色 \*/

#p2a {background-color: #00ff0080;} /\* 带透明度的绿色 \*/

#p3a {background-color: #0000ff80;} /\* 带透明度的蓝色 \*/

## 117.4 RGB 颜色

RGB 颜色值由 rgb() 函数规定，语法如下：

rgb(red, green, blue)

每个参数（red, green, blue）定义颜色的强度，可以是 0 到 255 之间的整数或百分比值（从 0％ 到 100％）。

例如，值 rgb(0,0,255) 呈现为蓝色，因为 blue 参数设置为其最高值（255），其他参数设置为 0。

此外，以下值定义相同的颜色：rgb(0,0,255) 和 rgb(0%,0%,100%)。

实例

定义不同的 RGB 颜色：

#p1 {background-color: rgb(255, 0, 0);} /\* 红色 \*/

#p2 {background-color: rgb(0, 255, 0);} /\* 绿色 \*/

#p3 {background-color: rgb(0, 0, 255);} /\* 蓝色 \*/

## 117.5 RGBA 颜色

RGBA 颜色值是 RGB 颜色值的扩展，它带有 Alpha 通道 - 指定对象的不透明度。

RGBA 颜色通过 rgba() 函数规定，语法如下：

rgba(red, green, blue, alpha)

alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字。

实例

定义带有不透明度的不同 RGB 颜色：

#p1 {background-color: rgba(255, 0, 0, 0.3);} /\* 带不透明度的红色 \*/

#p2 {background-color: rgba(0, 255, 0, 0.3);} /\* 带不透明度的绿色 \*/

#p3 {background-color: rgba(0, 0, 255, 0.3);} /\* 带不透明度的蓝色 \*/

## 117.6 HSL 颜色

HSL 指的是色相（hue）、饱和度（saturation）和亮度（lightness）- 代表颜色的圆柱坐标表示。

使用 hsl() 函数指定 HSL 颜色值，该函数的语法如下：

hsl(hue, saturation, lightness)

色相是色轮上的度数（从 0 到 360）- 0（或 360）是红色，120 是绿色，240 是蓝色。

饱和度是一个百分比值； 0％ 表示灰色阴影，而 100％ 是全彩色。

亮度也是一个百分比； 0％ 是黑色，100％ 是白色。

实例

定义不同的 HSL 颜色：

#p1 {background-color: hsl(120, 100%, 50%);} /\* 绿色 \*/

#p2 {background-color: hsl(120, 100%, 75%);} /\* 浅绿色 \*/

#p3 {background-color: hsl(120, 100%, 25%);} /\* 深绿色 \*/

#p4 {background-color: hsl(120, 60%, 70%);} /\* 柔和的绿色 \*/

## 117.7 HSLA 颜色

HSLA 颜色值是 HSL 颜色值的扩展，它带有 Alpha 通道 - 指定对象的不透明度。

HSLA 颜色值由 hsla() 函数指定，该函数的语法如下：

hsla(hue, saturation, lightness, alpha)

alpha 参数是介于 0.0（完全透明）和 1.0（完全不透明）之间的数字。

实例

定义带有不透明度的不同 HSL 颜色：

#p1 {background-color: hsla(120, 100%, 50%, 0.3);} /\* 带不透明度的绿色 \*/

#p2 {background-color: hsla(120, 100%, 75%, 0.3);} /\* 带不透明度的浅绿色 \*/

#p3 {background-color: hsla(120, 100%, 25%, 0.3);} /\* 带不透明度的深绿色 \*/

#p4 {background-color: hsla(120, 60%, 70%, 0.3);} /\* 带不透明度的柔绿色 \*/

## 117.8 预定义/跨浏览器的颜色名称

HTML 和 CSS 颜色规范中预定义了 140 个颜色名称。

例如：blue、red、coral、brown 等：

实例

定义不同的颜色名：

#p1 {background-color: blue;}

#p2 {background-color: red;}

#p3 {background-color: coral;}

#p4 {background-color: brown;}

所有预定义名称的列表都可以在我们的 颜色名称参考手册 中找到。

## 117.9 currentcolor 关键字

currentcolor 关键字引用元素的 color 属性值。

实例

以下 <div> 元素的边框颜色将为蓝色，因为 <div> 元素的文本颜色为蓝色：

#myDIV {

color: blue; /\* 蓝色文本色 \*/

border: 10px solid currentcolor; /\* 蓝色边框色 \*/

}

# 第118章 CSS 默认值参考手册

## 118.1 HTML 元素的 CSS 默认值

下表显示了所有 HTML 元素的默认 CSS 浏览器值。

|  |  |
| --- | --- |
| 元素 | 默认的 CSS 值 |
| a:link | color: (internal value);  text-decoration: underline;  cursor: auto; |
| a:visited | color: (internal value);  text-decoration: underline;  cursor: auto; |
| a:link:active | color: (internal value); |
| a:visited:active | color: (internal value); |
| abbr | None. |
| address | display: block;  font-style: italic; |
| area | display: none; |
| article | display: block; |
| aside | display: block; |
| audio | None. |
| ase | None. |
| bdi | None. |
| bdo | unicode-bidi: bidi-override; |
| blockquote | display: block;  margin-top: 1em;  margin-bottom: 1em;  margin-left: 40px;  margin-right: 40px; |
| body | display: block;  margin: 8px; |
| body:focus | outline: none; |
| br | None. |
| button | None |
| canvas | None. |
| caption | display: table-caption  text-align: center; |
| cite | font-style: italic; |
| code | font-family: monospace; |
| col | display: table-column; |
| colgroup | display: table-column-group |
| datalist | display: none; |
| dd | display: block;  margin-left: 40px; |
| el | text-decoration: line-through; |
| details | display: block; |
| dfn | font-style: italic; |
| dialog | None. |
| div | display: block; |
| dl | display: block;  margin-top: 1em;  margin-bottom: 1em;  margin-left: 0;  margin-right: 0; |
| dt | display: block; |
| em | font-style: italic; |
| embed:focus | outline: none; |
| fieldset | display: block;  margin-left: 2px;  margin-right: 2px;  padding-top: 0.35em;  padding-bottom: 0.625em;  padding-left: 0.75em;  padding-right: 0.75em;  border: 2px groove (internal value); |
| figcaption | display: block; |
| figure | display: block;  margin-top: 1em;  margin-bottom: 1em;  margin-left: 40px;  margin-right: 40px; |
| footer | display: block; |
| form | display: block;  margin-top: 0em; |
| h1 | display: block;  font-size: 2em;  margin-top: 0.67em;  margin-bottom: 0.67em;  margin-left: 0;  margin-right: 0;  font-weight: bold; |
| h2 | display: block;  font-size: 1.5em;  margin-top: 0.83em;  margin-bottom: 0.83em;  margin-left: 0;  margin-right: 0;  font-weight: bold; |
| h3 | display: block;  font-size: 1.17em;  margin-top: 1em;  margin-bottom: 1em;  margin-left: 0;  margin-right: 0;  font-weight: bold; |
| h4 | display: block;  margin-top: 1.33em;  margin-bottom: 1.33em;  margin-left: 0;  margin-right: 0;  font-weight: bold; |
| h5 | display: block;  font-size: .83em;  margin-top: 1.67em;  margin-bottom: 1.67em;  margin-left: 0;  margin-right: 0;  font-weight: bold; |
| h6 | display: block;  font-size: .67em;  margin-top: 2.33em;  margin-bottom: 2.33em;  margin-left: 0;  margin-right: 0;  font-weight: bold; |
| head | display: none; |
| header | display: block; |
| hr | display: block;  margin-top: 0.5em;  margin-bottom: 0.5em;  margin-left: auto;  margin-right: auto;  border-style: inset;  border-width: 1px; |
| html | display: block;  html:focus  outline: none; |
| i | font-style: italic; |
| iframe:focus | outline: none; |
| iframe[seamless] | display: block; |
| img | display: inline-block; |
| input | None. |
| ins | text-decoration: underline; |
| kbd | font-family: monospace; |
| label | cursor: default; |
| legend | display: block;  padding-left: 2px;  padding-right: 2px;  border: none; |
| li | display: list-item; |
| link | display: none; |
| main | None. |
| map | display: inline; |
| mark | background-color: yellow;  color: black; |
| menu | display: block;  list-style-type: disc;  margin-top: 1em;  margin-bottom: 1em;  margin-left: 0;  margin-right: 0;  padding-left: 40px; |
| menuitem | None. |
| meta | None. |
| meter | None. |
| nav | display: block; |
| noscript | None. |
| object:focus | outline: none; |
| ol | display: block;  list-style-type: decimal;  margin-top: 1em;  margin-bottom: 1em;  margin-left: 0;  margin-right: 0;  padding-left: 40px; |
| optgroup | None. |
| option | None. |
| output | display: inline; |
| p | display: block;  margin-top: 1em;  margin-bottom: 1em;  margin-left: 0;  margin-right: 0; |
| param | display: none; |
| picture | None. |
| pre | display: block;  font-family: monospace;  white-space: pre;  margin: 1em 0; |
| progress | None. |
| q | display: inline; |
| q::before | content: open-quote; |
| q::after | content: close-quote; |
| rp | None. |
| rt | line-height: normal; |
| ruby | None. |
| s | text-decoration: line-through; |
| samp | font-family: monospace; |
| script | display: none; |
| section | display: block; |
| select | None. |
| small | font-size: smaller; |
| source | None. |
| span | None. |
| strike | text-decoration: line-through; |
| strong | font-weight: bold; |
| style | display: none; |
| sub | vertical-align: sub;  font-size: smaller; |
| summary | display: block; |
| sup | vertical-align: super;  font-size: smaller; |
| table | display: table;  border-collapse: separate;  border-spacing: 2px;  border-color: gray; |
| tbody | display: table-row-group;  vertical-align: middle;  border-color: inherit; |
| td | display: table-cell;  vertical-align: inherit; |
| template | None. |
| textarea | None. |
| tfoot | display: table-footer-group;  vertical-align: middle;  border-color: inherit; |
| th | display: table-cell;  vertical-align: inherit;  font-weight: bold;  text-align: center; |
| thead | display: table-header-group;  vertical-align: middle;  border-color: inherit; |
| time | None. |
| title | display: none; |
| tr | display: table-row;  vertical-align: inherit;  border-color: inherit; |
| track | None. |
| u | text-decoration: underline; |
| ul | display: block;  list-style-type: disc;  margin-top: 1em;  margin-bottom: 1 em;  margin-left: 0;  margin-right: 0;  padding-left: 40px; |
| var | font-style: italic; |
| video | None. |
| wbr | None. |

# 第119章 CSS 实体

## 119.1 完整的 CSS 实体参考手册

如果希望在 HTML 中使用 CSS 显示以下任何字符，您可以使用下表中的 CSS 实体。

实例

<!DOCTYPE html>

<html>

<style>

h1:after {

content: ' \00A7';

}

</style>

<body>

<h1>我将显示：</h1>

<p>所有 h1 元素都将在末尾添加一个额外的字符。</p>

</body>

</html>

所有 <h1> 元素都将在结尾显示该字符：

字符 CSS 实体 名称

0020 SPACE

! 0021 EXCLAMATION MARK

" 0022 QUOTATION MARK

# 0023 NUMBER SIGN

$ 0024 DOLLAR SIGN

% 0025 PERCENT SIGN

& 0026 AMPERSAND

' 0027 APOSTROPHE

( 0028 LEFT PARENTHESIS

) 0029 RIGHT PARENTHESIS

\* 002A ASTERISK

+ 002B PLUS SIGN

, 002C COMMA

- 002D HYPHEN-MINUS

. 002E FULL STOP

/ 002F SOLIDUS

0 0030 DIGIT ZERO

1 0031 DIGIT ONE

2 0032 DIGIT TWO

3 0033 DIGIT THREE

4 0034 DIGIT FOUR

5 0035 DIGIT FIVE

6 0036 DIGIT SIX

7 0037 DIGIT SEVEN

8 0038 DIGIT EIGHT

9 0039 DIGIT NINE

: 003A COLON

; 003B SEMICOLON

< 003C LESS-THAN SIGN

= 003D EQUALS SIGN

> 003E GREATER-THAN SIGN

? 003F QUESTION MARK

@ 0040 COMMERCIAL AT

A 0041 LATIN CAPITAL LETTER A

B 0042 LATIN CAPITAL LETTER B

C 0043 LATIN CAPITAL LETTER C

D 0044 LATIN CAPITAL LETTER D

E 0045 LATIN CAPITAL LETTER E

F 0046 LATIN CAPITAL LETTER F

G 0047 LATIN CAPITAL LETTER G

H 0048 LATIN CAPITAL LETTER H

I 0049 LATIN CAPITAL LETTER I

J 004A LATIN CAPITAL LETTER J

K 004B LATIN CAPITAL LETTER K

L 004C LATIN CAPITAL LETTER L

M 004D LATIN CAPITAL LETTER M

N 004E LATIN CAPITAL LETTER N

O 004F LATIN CAPITAL LETTER O

P 0050 LATIN CAPITAL LETTER P

Q 0051 LATIN CAPITAL LETTER Q

R 0052 LATIN CAPITAL LETTER R

S 0053 LATIN CAPITAL LETTER S

T 0054 LATIN CAPITAL LETTER T

U 0055 LATIN CAPITAL LETTER U

V 0056 LATIN CAPITAL LETTER V

W 0057 LATIN CAPITAL LETTER W

X 0058 LATIN CAPITAL LETTER X

Y 0059 LATIN CAPITAL LETTER Y

Z 005A LATIN CAPITAL LETTER Z

[ 005B LEFT SQUARE BRACKET

\ 005C REVERSE SOLIDUS

] 005D RIGHT SQUARE BRACKET

^ 005E CIRCUMFLEX ACCENT

\_ 005F LOW LINE

` 0060 GRAVE ACCENT

a 0061 LATIN SMALL LETTER A

b 0062 LATIN SMALL LETTER B

c 0063 LATIN SMALL LETTER C

d 0064 LATIN SMALL LETTER D

e 0065 LATIN SMALL LETTER E

f 0066 LATIN SMALL LETTER F

g 0067 LATIN SMALL LETTER G

h 0068 LATIN SMALL LETTER H

i 0069 LATIN SMALL LETTER I

j 006A LATIN SMALL LETTER J

k 006B LATIN SMALL LETTER K

l 006C LATIN SMALL LETTER L

m 006D LATIN SMALL LETTER M

n 006E LATIN SMALL LETTER N

o 006F LATIN SMALL LETTER O

p 0070 LATIN SMALL LETTER P

q 0071 LATIN SMALL LETTER Q

r 0072 LATIN SMALL LETTER R

s 0073 LATIN SMALL LETTER S

t 0074 LATIN SMALL LETTER T

u 0075 LATIN SMALL LETTER U

v 0076 LATIN SMALL LETTER V

w 0077 LATIN SMALL LETTER W

x 0078 LATIN SMALL LETTER X

y 0079 LATIN SMALL LETTER Y

z 007A LATIN SMALL LETTER Z

{ 007B LEFT CURLY BRACKET

| 007C VERTICAL LINE

} 007D RIGHT CURLY BRACKET

~ 007E TILDE

00A0 NO-BREAK SPACE

¡ 00A1 INVERTED EXCLAMATION MARK

¢ 00A2 CENT SIGN

£ 00A3 POUND SIGN

¤ 00A4 CURRENCY SIGN

¥ 00A5 YEN SIGN

¦ 00A6 BROKEN BAR

§ 00A7 SECTION SIGN

¨ 00A8 DIAERESIS

© 00A9 COPYRIGHT SIGN

ª 00AA FEMININE ORDINAL INDICATOR

« 00AB LEFT-POINTING DOUBLE ANGLE QUOTATION MARK

¬ 00AC NOT SIGN

­ 00AD SOFT HYPHEN

® 00AE REGISTERED SIGN

¯ 00AF MACRON

° 00B0 DEGREE SIGN

± 00B1 PLUS-MINUS SIGN

² 00B2 SUPERSCRIPT TWO

³ 00B3 SUPERSCRIPT THREE

´ 00B4 ACUTE ACCENT

µ 00B5 MICRO SIGN

¶ 00B6 PILCROW SIGN

· 00B7 MIDDLE DOT

¸ 00B8 CEDILLA

¹ 00B9 SUPERSCRIPT ONE

º 00BA MASCULINE ORDINAL INDICATOR

» 00BB RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK

¼ 00BC VULGAR FRACTION ONE QUARTER

½ 00BD VULGAR FRACTION ONE HALF

¾ 00BE VULGAR FRACTION THREE QUARTERS

¿ 00BF INVERTED QUESTION MARK

À 00C0 LATIN CAPITAL LETTER A WITH GRAVE

Á 00C1 LATIN CAPITAL LETTER A WITH ACUTE

Â 00C2 LATIN CAPITAL LETTER A WITH CIRCUMFLEX

Ã 00C3 LATIN CAPITAL LETTER A WITH TILDE

Ä 00C4 LATIN CAPITAL LETTER A WITH DIAERESIS

Å 00C5 LATIN CAPITAL LETTER A WITH RING ABOVE

Æ 00C6 LATIN CAPITAL LETTER AE

Ç 00C7 LATIN CAPITAL LETTER C WITH CEDILLA

È 00C8 LATIN CAPITAL LETTER E WITH GRAVE

É 00C9 LATIN CAPITAL LETTER E WITH ACUTE

Ê 00CA LATIN CAPITAL LETTER E WITH CIRCUMFLEX

Ë 00CB LATIN CAPITAL LETTER E WITH DIAERESIS

Ì 00CC LATIN CAPITAL LETTER I WITH GRAVE

Í 00CD LATIN CAPITAL LETTER I WITH ACUTE

Î 00CE LATIN CAPITAL LETTER I WITH CIRCUMFLEX

Ï 00CF LATIN CAPITAL LETTER I WITH DIAERESIS

Ð 00D0 LATIN CAPITAL LETTER ETH

Ñ 00D1 LATIN CAPITAL LETTER N WITH TILDE

Ò 00D2 LATIN CAPITAL LETTER O WITH GRAVE

Ó 00D3 LATIN CAPITAL LETTER O WITH ACUTE

Ô 00D4 LATIN CAPITAL LETTER O WITH CIRCUMFLEX

Õ 00D5 LATIN CAPITAL LETTER O WITH TILDE

Ö 00D6 LATIN CAPITAL LETTER O WITH DIAERESIS

× 00D7 MULTIPLICATION SIGN

Ø 00D8 LATIN CAPITAL LETTER O WITH STROKE

Ù 00D9 LATIN CAPITAL LETTER U WITH GRAVE

Ú 00DA LATIN CAPITAL LETTER U WITH ACUTE

Û 00DB LATIN CAPITAL LETTER U WITH CIRCUMFLEX

Ü 00DC LATIN CAPITAL LETTER U WITH DIAERESIS

Ý 00DD LATIN CAPITAL LETTER Y WITH ACUTE

Þ 00DE LATIN CAPITAL LETTER THORN

ß 00DF LATIN SMALL LETTER SHARP S

à 00E0 LATIN SMALL LETTER A WITH GRAVE

á 00E1 LATIN SMALL LETTER A WITH ACUTE

â 00E2 LATIN SMALL LETTER A WITH CIRCUMFLEX

ã 00E3 LATIN SMALL LETTER A WITH TILDE

ä 00E4 LATIN SMALL LETTER A WITH DIAERESIS

å 00E5 LATIN SMALL LETTER A WITH RING ABOVE

æ 00E6 LATIN SMALL LETTER AE

ç 00E7 LATIN SMALL LETTER C WITH CEDILLA

è 00E8 LATIN SMALL LETTER E WITH GRAVE

é 00E9 LATIN SMALL LETTER E WITH ACUTE

ê 00EA LATIN SMALL LETTER E WITH CIRCUMFLEX

ë 00EB LATIN SMALL LETTER E WITH DIAERESIS

ì 00EC LATIN SMALL LETTER I WITH GRAVE

í 00ED LATIN SMALL LETTER I WITH ACUTE

î 00EE LATIN SMALL LETTER I WITH CIRCUMFLEX

ï 00EF LATIN SMALL LETTER I WITH DIAERESIS

ð 00F0 LATIN SMALL LETTER ETH

ñ 00F1 LATIN SMALL LETTER N WITH TILDE

ò 00F2 LATIN SMALL LETTER O WITH GRAVE

ó 00F3 LATIN SMALL LETTER O WITH ACUTE

ô 00F4 LATIN SMALL LETTER O WITH CIRCUMFLEX

õ 00F5 LATIN SMALL LETTER O WITH TILDE

ö 00F6 LATIN SMALL LETTER O WITH DIAERESIS

÷ 00F7 DIVISION SIGN

ø 00F8 LATIN SMALL LETTER O WITH STROKE

ù 00F9 LATIN SMALL LETTER U WITH GRAVE

ú 00FA LATIN SMALL LETTER U WITH ACUTE

û 00FB LATIN SMALL LETTER U WITH CIRCUMFLEX

ü 00FC LATIN SMALL LETTER U WITH DIAERESIS

ý 00FD LATIN SMALL LETTER Y WITH ACUTE

þ 00FE LATIN SMALL LETTER THORN

ÿ 00FF LATIN SMALL LETTER Y WITH DIAERESIS

Ā 0100 LATIN CAPITAL LETTER A WITH MACRON

ā 0101 LATIN SMALL LETTER A WITH MACRON

Ă 0102 LATIN CAPITAL LETTER A WITH BREVE

ă 0103 LATIN SMALL LETTER A WITH BREVE

Ą 0104 LATIN CAPITAL LETTER A WITH OGONEK

ą 0105 LATIN SMALL LETTER A WITH OGONEK

Ć 0106 LATIN CAPITAL LETTER C WITH ACUTE

ć 0107 LATIN SMALL LETTER C WITH ACUTE

Ĉ 0108 LATIN CAPITAL LETTER C WITH CIRCUMFLEX

ĉ 0109 LATIN SMALL LETTER C WITH CIRCUMFLEX

Ċ 010A LATIN CAPITAL LETTER C WITH DOT ABOVE

ċ 010B LATIN SMALL LETTER C WITH DOT ABOVE

Č 010C LATIN CAPITAL LETTER C WITH CARON

č 010D LATIN SMALL LETTER C WITH CARON

Ď 010E LATIN CAPITAL LETTER D WITH CARON

ď 010F LATIN SMALL LETTER D WITH CARON

Đ 0110 LATIN CAPITAL LETTER D WITH STROKE

đ 0111 LATIN SMALL LETTER D WITH STROKE

Ē 0112 LATIN CAPITAL LETTER E WITH MACRON

ē 0113 LATIN SMALL LETTER E WITH MACRON

Ĕ 0114 LATIN CAPITAL LETTER E WITH BREVE

ĕ 0115 LATIN SMALL LETTER E WITH BREVE

Ė 0116 LATIN CAPITAL LETTER E WITH DOT ABOVE

ė 0117 LATIN SMALL LETTER E WITH DOT ABOVE

Ę 0118 LATIN CAPITAL LETTER E WITH OGONEK

ę 0119 LATIN SMALL LETTER E WITH OGONEK

Ě 011A LATIN CAPITAL LETTER E WITH CARON

ě 011B LATIN SMALL LETTER E WITH CARON

Ĝ 011C LATIN CAPITAL LETTER G WITH CIRCUMFLEX

ĝ 011D LATIN SMALL LETTER G WITH CIRCUMFLEX

Ğ 011E LATIN CAPITAL LETTER G WITH BREVE

ğ 011F LATIN SMALL LETTER G WITH BREVE

Ġ 0120 LATIN CAPITAL LETTER G WITH DOT ABOVE

ġ 0121 LATIN SMALL LETTER G WITH DOT ABOVE

Ģ 0122 LATIN CAPITAL LETTER G WITH CEDILLA

ģ 0123 LATIN SMALL LETTER G WITH CEDILLA

Ĥ 0124 LATIN CAPITAL LETTER H WITH CIRCUMFLEX

ĥ 0125 LATIN SMALL LETTER H WITH CIRCUMFLEX

Ħ 0126 LATIN CAPITAL LETTER H WITH STROKE

ħ 0127 LATIN SMALL LETTER H WITH STROKE

Ĩ 0128 LATIN CAPITAL LETTER I WITH TILDE

ĩ 0129 LATIN SMALL LETTER I WITH TILDE

Ī 012A LATIN CAPITAL LETTER I WITH MACRON

ī 012B LATIN SMALL LETTER I WITH MACRON

Ĭ 012C LATIN CAPITAL LETTER I WITH BREVE

ĭ 012D LATIN SMALL LETTER I WITH BREVE

Į 012E LATIN CAPITAL LETTER I WITH OGONEK

į 012F LATIN SMALL LETTER I WITH OGONEK

İ 0130 LATIN CAPITAL LETTER I WITH DOT ABOVE

ı 0131 LATIN SMALL LETTER DOTLESS I

Ĳ 0132 LATIN CAPITAL LIGATURE IJ

ĳ 0133 LATIN SMALL LIGATURE IJ

Ĵ 0134 LATIN CAPITAL LETTER J WITH CIRCUMFLEX

ĵ 0135 LATIN SMALL LETTER J WITH CIRCUMFLEX

Ķ 0136 LATIN CAPITAL LETTER K WITH CEDILLA

ķ 0137 LATIN SMALL LETTER K WITH CEDILLA

ĸ 0138 LATIN SMALL LETTER KRA

Ĺ 0139 LATIN CAPITAL LETTER L WITH ACUTE

ĺ 013A LATIN SMALL LETTER L WITH ACUTE

Ļ 013B LATIN CAPITAL LETTER L WITH CEDILLA

ļ 013C LATIN SMALL LETTER L WITH CEDILLA

Ľ 013D LATIN CAPITAL LETTER L WITH CARON

ľ 013E LATIN SMALL LETTER L WITH CARON

Ŀ 013F LATIN CAPITAL LETTER L WITH MIDDLE DOT

ŀ 0140 LATIN SMALL LETTER L WITH MIDDLE DOT

Ł 0141 LATIN CAPITAL LETTER L WITH STROKE

ł 0142 LATIN SMALL LETTER L WITH STROKE

Ń 0143 LATIN CAPITAL LETTER N WITH ACUTE

ń 0144 LATIN SMALL LETTER N WITH ACUTE

Ņ 0145 LATIN CAPITAL LETTER N WITH CEDILLA

ņ 0146 LATIN SMALL LETTER N WITH CEDILLA

Ň 0147 LATIN CAPITAL LETTER N WITH CARON

ň 0148 LATIN SMALL LETTER N WITH CARON

ŉ 0149 LATIN SMALL LETTER N PRECEDED BY APOSTROPHE

Ŋ 014A LATIN CAPITAL LETTER ENG

ŋ 014B LATIN SMALL LETTER ENG

Ō 014C LATIN CAPITAL LETTER O WITH MACRON

ō 014D LATIN SMALL LETTER O WITH MACRON

Ŏ 014E LATIN CAPITAL LETTER O WITH BREVE

ŏ 014F LATIN SMALL LETTER O WITH BREVE

Ő 0150 LATIN CAPITAL LETTER O WITH DOUBLE ACUTE

ő 0151 LATIN SMALL LETTER O WITH DOUBLE ACUTE

Œ 0152 LATIN CAPITAL LIGATURE OE

œ 0153 LATIN SMALL LIGATURE OE

Ŕ 0154 LATIN CAPITAL LETTER R WITH ACUTE

ŕ 0155 LATIN SMALL LETTER R WITH ACUTE

Ŗ 0156 LATIN CAPITAL LETTER R WITH CEDILLA

ŗ 0157 LATIN SMALL LETTER R WITH CEDILLA

Ř 0158 LATIN CAPITAL LETTER R WITH CARON

ř 0159 LATIN SMALL LETTER R WITH CARON

Ś 015A LATIN CAPITAL LETTER S WITH ACUTE

ś 015B LATIN SMALL LETTER S WITH ACUTE

Ŝ 015C LATIN CAPITAL LETTER S WITH CIRCUMFLEX

ŝ 015D LATIN SMALL LETTER S WITH CIRCUMFLEX

Ş 015E LATIN CAPITAL LETTER S WITH CEDILLA

ş 015F LATIN SMALL LETTER S WITH CEDILLA

Š 0160 LATIN CAPITAL LETTER S WITH CARON

š 0161 LATIN SMALL LETTER S WITH CARON

Ţ 0162 LATIN CAPITAL LETTER T WITH CEDILLA

ţ 0163 LATIN SMALL LETTER T WITH CEDILLA

Ť 0164 LATIN CAPITAL LETTER T WITH CARON

ť 0165 LATIN SMALL LETTER T WITH CARON

Ŧ 0166 LATIN CAPITAL LETTER T WITH STROKE

ŧ 0167 LATIN SMALL LETTER T WITH STROKE

Ũ 0168 LATIN CAPITAL LETTER U WITH TILDE

ũ 0169 LATIN SMALL LETTER U WITH TILDE

Ū 016A LATIN CAPITAL LETTER U WITH MACRON

ū 016B LATIN SMALL LETTER U WITH MACRON

Ŭ 016C LATIN CAPITAL LETTER U WITH BREVE

ŭ 016D LATIN SMALL LETTER U WITH BREVE

Ů 016E LATIN CAPITAL LETTER U WITH RING ABOVE

ů 016F LATIN SMALL LETTER U WITH RING ABOVE

Ű 0170 LATIN CAPITAL LETTER U WITH DOUBLE ACUTE

ű 0171 LATIN SMALL LETTER U WITH DOUBLE ACUTE

Ų 0172 LATIN CAPITAL LETTER U WITH OGONEK

ų 0173 LATIN SMALL LETTER U WITH OGONEK

Ŵ 0174 LATIN CAPITAL LETTER W WITH CIRCUMFLEX

ŵ 0175 LATIN SMALL LETTER W WITH CIRCUMFLEX

Ŷ 0176 LATIN CAPITAL LETTER Y WITH CIRCUMFLEX

ŷ 0177 LATIN SMALL LETTER Y WITH CIRCUMFLEX

Ÿ 0178 LATIN CAPITAL LETTER Y WITH DIAERESIS

Ź 0179 LATIN CAPITAL LETTER Z WITH ACUTE

ź 017A LATIN SMALL LETTER Z WITH ACUTE

Ż 017B LATIN CAPITAL LETTER Z WITH DOT ABOVE

ż 017C LATIN SMALL LETTER Z WITH DOT ABOVE

Ž 017D LATIN CAPITAL LETTER Z WITH CARON

ž 017E LATIN SMALL LETTER Z WITH CARON

ſ 017F LATIN SMALL LETTER LONG S

ƀ 0180 LATIN SMALL LETTER B WITH STROKE

Ɓ 0181 LATIN CAPITAL LETTER B WITH HOOK

Ƃ 0182 LATIN CAPITAL LETTER B WITH TOPBAR

ƃ 0183 LATIN SMALL LETTER B WITH TOPBAR

Ƅ 0184 LATIN CAPITAL LETTER TONE SIX

ƅ 0185 LATIN SMALL LETTER TONE SIX

Ɔ 0186 LATIN CAPITAL LETTER OPEN O

Ƈ 0187 LATIN CAPITAL LETTER C WITH HOOK

ƈ 0188 LATIN SMALL LETTER C WITH HOOK

Ɖ 0189 LATIN CAPITAL LETTER AFRICAN D

Ɗ 018A LATIN CAPITAL LETTER D WITH HOOK

Ƌ 018B LATIN CAPITAL LETTER D WITH TOPBAR

ƌ 018C LATIN SMALL LETTER D WITH TOPBAR

ƍ 018D LATIN SMALL LETTER TURNED DELTA

Ǝ 018E LATIN CAPITAL LETTER REVERSED E

Ə 018F LATIN CAPITAL LETTER SCHWA

Ɛ 0190 LATIN CAPITAL LETTER OPEN E

Ƒ 0191 LATIN CAPITAL LETTER F WITH HOOK

ƒ 0192 LATIN SMALL LETTER F WITH HOOK

Ɠ 0193 LATIN CAPITAL LETTER G WITH HOOK

Ɣ 0194 LATIN CAPITAL LETTER GAMMA

ƕ 0195 LATIN SMALL LETTER HV

Ɩ 0196 LATIN CAPITAL LETTER IOTA

Ɨ 0197 LATIN CAPITAL LETTER I WITH STROKE

Ƙ 0198 LATIN CAPITAL LETTER K WITH HOOK

ƙ 0199 LATIN SMALL LETTER K WITH HOOK

ƚ 019A LATIN SMALL LETTER L WITH BAR

ƛ 019B LATIN SMALL LETTER LAMBDA WITH STROKE

Ɯ 019C LATIN CAPITAL LETTER TURNED M

Ɲ 019D LATIN CAPITAL LETTER N WITH LEFT HOOK

ƞ 019E LATIN SMALL LETTER N WITH LONG RIGHT LEG

Ɵ 019F LATIN CAPITAL LETTER O WITH MIDDLE TILDE

Ơ 01A0 LATIN CAPITAL LETTER O WITH HORN

ơ 01A1 LATIN SMALL LETTER O WITH HORN

Ƣ 01A2 LATIN CAPITAL LETTER OI

ƣ 01A3 LATIN SMALL LETTER OI

Ƥ 01A4 LATIN CAPITAL LETTER P WITH HOOK

ƥ 01A5 LATIN SMALL LETTER P WITH HOOK

Ʀ 01A6 LATIN LETTER YR

Ƨ 01A7 LATIN CAPITAL LETTER TONE TWO

ƨ 01A8 LATIN SMALL LETTER TONE TWO

Ʃ 01A9 LATIN CAPITAL LETTER ESH

ƪ 01AA LATIN LETTER REVERSED ESH LOOP

ƫ 01AB LATIN SMALL LETTER T WITH PALATAL HOOK

Ƭ 01AC LATIN CAPITAL LETTER T WITH HOOK

ƭ 01AD LATIN SMALL LETTER T WITH HOOK

Ʈ 01AE LATIN CAPITAL LETTER T WITH RETROFLEX HOOK

Ư 01AF LATIN CAPITAL LETTER U WITH HORN

ư 01B0 LATIN SMALL LETTER U WITH HORN

Ʊ 01B1 LATIN CAPITAL LETTER UPSILON

Ʋ 01B2 LATIN CAPITAL LETTER V WITH HOOK

Ƴ 01B3 LATIN CAPITAL LETTER Y WITH HOOK

ƴ 01B4 LATIN SMALL LETTER Y WITH HOOK

Ƶ 01B5 LATIN CAPITAL LETTER Z WITH STROKE

ƶ 01B6 LATIN SMALL LETTER Z WITH STROKE

Ʒ 01B7 LATIN CAPITAL LETTER EZH

Ƹ 01B8 LATIN CAPITAL LETTER EZH REVERSED

ƹ 01B9 LATIN SMALL LETTER EZH REVERSED

ƺ 01BA LATIN SMALL LETTER EZH WITH TAIL

ƻ 01BB LATIN LETTER TWO WITH STROKE

Ƽ 01BC LATIN CAPITAL LETTER TONE FIVE

ƽ 01BD LATIN SMALL LETTER TONE FIVE

ƾ 01BE LATIN LETTER INVERTED GLOTTAL STOP WITH STROKE

ƿ 01BF LATIN LETTER WYNN

ǀ 01C0 LATIN LETTER DENTAL CLICK

ǁ 01C1 LATIN LETTER LATERAL CLICK

ǂ 01C2 LATIN LETTER ALVEOLAR CLICK

ǃ 01C3 LATIN LETTER RETROFLEX CLICK

Ǆ 01C4 LATIN CAPITAL LETTER DZ WITH CARON

ǅ 01C5 LATIN CAPITAL LETTER D WITH SMALL LETTER Z WITH CARON

ǆ 01C6 LATIN SMALL LETTER DZ WITH CARON

Ǉ 01C7 LATIN CAPITAL LETTER LJ

ǈ 01C8 LATIN CAPITAL LETTER L WITH SMALL LETTER J

ǉ 01C9 LATIN SMALL LETTER LJ

Ǌ 01CA LATIN CAPITAL LETTER NJ

ǋ 01CB LATIN CAPITAL LETTER N WITH SMALL LETTER J

ǌ 01CC LATIN SMALL LETTER NJ

Ǎ 01CD LATIN CAPITAL LETTER A WITH CARON

ǎ 01CE LATIN SMALL LETTER A WITH CARON

Ǐ 01CF LATIN CAPITAL LETTER I WITH CARON

ǐ 01D0 LATIN SMALL LETTER I WITH CARON

Ǒ 01D1 LATIN CAPITAL LETTER O WITH CARON

ǒ 01D2 LATIN SMALL LETTER O WITH CARON

Ǔ 01D3 LATIN CAPITAL LETTER U WITH CARON

ǔ 01D4 LATIN SMALL LETTER U WITH CARON

Ǖ 01D5 LATIN CAPITAL LETTER U WITH DIAERESIS AND MACRON

ǖ 01D6 LATIN SMALL LETTER U WITH DIAERESIS AND MACRON

Ǘ 01D7 LATIN CAPITAL LETTER U WITH DIAERESIS AND ACUTE

ǘ 01D8 LATIN SMALL LETTER U WITH DIAERESIS AND ACUTE

Ǚ 01D9 LATIN CAPITAL LETTER U WITH DIAERESIS AND CARON

ǚ 01DA LATIN SMALL LETTER U WITH DIAERESIS AND CARON

Ǜ 01DB LATIN CAPITAL LETTER U WITH DIAERESIS AND GRAVE

ǜ 01DC LATIN SMALL LETTER U WITH DIAERESIS AND GRAVE

ǝ 01DD LATIN SMALL LETTER TURNED E

Ǟ 01DE LATIN CAPITAL LETTER A WITH DIAERESIS AND MACRON

ǟ 01DF LATIN SMALL LETTER A WITH DIAERESIS AND MACRON

Ǡ 01E0 LATIN CAPITAL LETTER A WITH DOT ABOVE AND MACRON

ǡ 01E1 LATIN SMALL LETTER A WITH DOT ABOVE AND MACRON

Ǣ 01E2 LATIN CAPITAL LETTER AE WITH MACRON

ǣ 01E3 LATIN SMALL LETTER AE WITH MACRON

Ǥ 01E4 LATIN CAPITAL LETTER G WITH STROKE

ǥ 01E5 LATIN SMALL LETTER G WITH STROKE

Ǧ 01E6 LATIN CAPITAL LETTER G WITH CARON

ǧ 01E7 LATIN SMALL LETTER G WITH CARON

Ǩ 01E8 LATIN CAPITAL LETTER K WITH CARON

ǩ 01E9 LATIN SMALL LETTER K WITH CARON

Ǫ 01EA LATIN CAPITAL LETTER O WITH OGONEK

ǫ 01EB LATIN SMALL LETTER O WITH OGONEK

Ǭ 01EC LATIN CAPITAL LETTER O WITH OGONEK AND MACRON

ǭ 01ED LATIN SMALL LETTER O WITH OGONEK AND MACRON

Ǯ 01EE LATIN CAPITAL LETTER EZH WITH CARON

ǯ 01EF LATIN SMALL LETTER EZH WITH CARON

ǰ 01F0 LATIN SMALL LETTER J WITH CARON

Ǳ 01F1 LATIN CAPITAL LETTER DZ

ǲ 01F2 LATIN CAPITAL LETTER D WITH SMALL LETTER Z

ǳ 01F3 LATIN SMALL LETTER DZ

Ǵ 01F4 LATIN CAPITAL LETTER G WITH ACUTE

ǵ 01F5 LATIN SMALL LETTER G WITH ACUTE

Ƕ 01F6 LATIN CAPITAL LETTER HWAIR

Ƿ 01F7 LATIN CAPITAL LETTER WYNN

Ǹ 01F8 LATIN CAPITAL LETTER N WITH GRAVE

ǹ 01F9 LATIN SMALL LETTER N WITH GRAVE

Ǻ 01FA LATIN CAPITAL LETTER A WITH RING ABOVE AND ACUTE

ǻ 01FB LATIN SMALL LETTER A WITH RING ABOVE AND ACUTE

Ǽ 01FC LATIN CAPITAL LETTER AE WITH ACUTE

ǽ 01FD LATIN SMALL LETTER AE WITH ACUTE

Ǿ 01FE LATIN CAPITAL LETTER O WITH STROKE AND ACUTE

ǿ 01FF LATIN SMALL LETTER O WITH STROKE AND ACUTE

Ȁ 0200 LATIN CAPITAL LETTER A WITH DOUBLE GRAVE

ȁ 0201 LATIN SMALL LETTER A WITH DOUBLE GRAVE

Ȃ 0202 LATIN CAPITAL LETTER A WITH INVERTED BREVE

ȃ 0203 LATIN SMALL LETTER A WITH INVERTED BREVE

Ȅ 0204 LATIN CAPITAL LETTER E WITH DOUBLE GRAVE

ȅ 0205 LATIN SMALL LETTER E WITH DOUBLE GRAVE

Ȇ 0206 LATIN CAPITAL LETTER E WITH INVERTED BREVE

ȇ 0207 LATIN SMALL LETTER E WITH INVERTED BREVE

Ȉ 0208 LATIN CAPITAL LETTER I WITH DOUBLE GRAVE

ȉ 0209 LATIN SMALL LETTER I WITH DOUBLE GRAVE

Ȋ 020A LATIN CAPITAL LETTER I WITH INVERTED BREVE

ȋ 020B LATIN SMALL LETTER I WITH INVERTED BREVE

Ȍ 020C LATIN CAPITAL LETTER O WITH DOUBLE GRAVE

ȍ 020D LATIN SMALL LETTER O WITH DOUBLE GRAVE

Ȏ 020E LATIN CAPITAL LETTER O WITH INVERTED BREVE

ȏ 020F LATIN SMALL LETTER O WITH INVERTED BREVE

Ȑ 0210 LATIN CAPITAL LETTER R WITH DOUBLE GRAVE

ȑ 0211 LATIN SMALL LETTER R WITH DOUBLE GRAVE

Ȓ 0212 LATIN CAPITAL LETTER R WITH INVERTED BREVE

ȓ 0213 LATIN SMALL LETTER R WITH INVERTED BREVE

Ȕ 0214 LATIN CAPITAL LETTER U WITH DOUBLE GRAVE

ȕ 0215 LATIN SMALL LETTER U WITH DOUBLE GRAVE

Ȗ 0216 LATIN CAPITAL LETTER U WITH INVERTED BREVE

ȗ 0217 LATIN SMALL LETTER U WITH INVERTED BREVE

Ș 0218 LATIN CAPITAL LETTER S WITH COMMA BELOW

ș 0219 LATIN SMALL LETTER S WITH COMMA BELOW

Ț 021A LATIN CAPITAL LETTER T WITH COMMA BELOW

ț 021B LATIN SMALL LETTER T WITH COMMA BELOW

Ȝ 021C LATIN CAPITAL LETTER YOGH

ȝ 021D LATIN SMALL LETTER YOGH

Ȟ 021E LATIN CAPITAL LETTER H WITH CARON

ȟ 021F LATIN SMALL LETTER H WITH CARON

Ƞ 0220 LATIN CAPITAL LETTER N WITH LONG RIGHT LEG

ȡ 0221 LATIN SMALL LETTER D WITH CURL

Ȣ 0222 LATIN CAPITAL LETTER OU

ȣ 0223 LATIN SMALL LETTER OU

Ȥ 0224 LATIN CAPITAL LETTER Z WITH HOOK

ȥ 0225 LATIN SMALL LETTER Z WITH HOOK

Ȧ 0226 LATIN CAPITAL LETTER A WITH DOT ABOVE

ȧ 0227 LATIN SMALL LETTER A WITH DOT ABOVE

Ȩ 0228 LATIN CAPITAL LETTER E WITH CEDILLA

ȩ 0229 LATIN SMALL LETTER E WITH CEDILLA

Ȫ 022A LATIN CAPITAL LETTER O WITH DIAERESIS AND MACRON

ȫ 022B LATIN SMALL LETTER O WITH DIAERESIS AND MACRON

Ȭ 022C LATIN CAPITAL LETTER O WITH TILDE AND MACRON

ȭ 022D LATIN SMALL LETTER O WITH TILDE AND MACRON

Ȯ 022E LATIN CAPITAL LETTER O WITH DOT ABOVE

ȯ 022F LATIN SMALL LETTER O WITH DOT ABOVE

Ȱ 0230 LATIN CAPITAL LETTER O WITH DOT ABOVE AND MACRON

ȱ 0231 LATIN SMALL LETTER O WITH DOT ABOVE AND MACRON

Ȳ 0232 LATIN CAPITAL LETTER Y WITH MACRON

ȳ 0233 LATIN SMALL LETTER Y WITH MACRON

ȴ 0234 LATIN SMALL LETTER L WITH CURL

ȵ 0235 LATIN SMALL LETTER N WITH CURL

ȶ 0236 LATIN SMALL LETTER T WITH CURL

ȷ 0237 LATIN SMALL LETTER DOTLESS J

ȸ 0238 LATIN SMALL LETTER DB DIGRAPH

ȹ 0239 LATIN SMALL LETTER QP DIGRAPH

Ⱥ 023A LATIN CAPITAL LETTER A WITH STROKE

Ȼ 023B LATIN CAPITAL LETTER C WITH STROKE

ȼ 023C LATIN SMALL LETTER C WITH STROKE

Ƚ 023D LATIN CAPITAL LETTER L WITH BAR

Ⱦ 023E LATIN CAPITAL LETTER T WITH DIAGONAL STROKE

ȿ 023F LATIN SMALL LETTER S WITH SWASH TAIL

ɀ 0240 LATIN SMALL LETTER Z WITH SWASH TAIL

Ɂ 0241 LATIN CAPITAL LETTER GLOTTAL STOP

ɂ 0242 LATIN SMALL LETTER GLOTTAL STOP

Ƀ 0243 LATIN CAPITAL LETTER B WITH STROKE

Ʉ 0244 LATIN CAPITAL LETTER U BAR

Ʌ 0245 LATIN CAPITAL LETTER TURNED V

Ɇ 0246 LATIN CAPITAL LETTER E WITH STROKE

ɇ 0247 LATIN SMALL LETTER E WITH STROKE

Ɉ 0248 LATIN CAPITAL LETTER J WITH STROKE

ɉ 0249 LATIN SMALL LETTER J WITH STROKE

Ɋ 024A LATIN CAPITAL LETTER SMALL Q WITH HOOK TAIL

ɋ 024B LATIN SMALL LETTER Q WITH HOOK TAIL

Ɍ 024C LATIN CAPITAL LETTER R WITH STROKE

ɍ 024D LATIN SMALL LETTER R WITH STROKE

Ɏ 024E LATIN CAPITAL LETTER Y WITH STROKE

ɏ 024F LATIN SMALL LETTER Y WITH STROKE

ʰ 02B0 MODIFIER LETTER SMALL H

ʱ 02B1 MODIFIER LETTER SMALL H WITH HOOK

ʲ 02B2 MODIFIER LETTER SMALL J

ʳ 02B3 MODIFIER LETTER SMALL R

ʴ 02B4 MODIFIER LETTER SMALL TURNED R

ʵ 02B5 MODIFIER LETTER SMALL TURNED R WITH HOOK

ʶ 02B6 MODIFIER LETTER SMALL CAPITAL INVERTED R

ʷ 02B7 MODIFIER LETTER SMALL W

ʸ 02B8 MODIFIER LETTER SMALL Y

ʹ 02B9 MODIFIER LETTER PRIME

ʺ 02BA MODIFIER LETTER DOUBLE PRIME

ʻ 02BB MODIFIER LETTER TURNED COMMA

ʼ 02BC MODIFIER LETTER APOSTROPHE

ʽ 02BD MODIFIER LETTER REVERSED COMMA

ʾ 02BE MODIFIER LETTER RIGHT HALF RING

ʿ 02BF MODIFIER LETTER LEFT HALF RING

ˀ 02C0 MODIFIER LETTER GLOTTAL STOP

ˁ 02C1 MODIFIER LETTER REVERSED GLOTTAL STOP

˂ 02C2 MODIFIER LETTER LEFT ARROWHEAD

˃ 02C3 MODIFIER LETTER RIGHT ARROWHEAD

˄ 02C4 MODIFIER LETTER UP ARROWHEAD

˅ 02C5 MODIFIER LETTER DOWN ARROWHEAD

ˆ 02C6 MODIFIER LETTER CIRCUMFLEX ACCENT

ˇ 02C7 CARON

ˈ 02C8 MODIFIER LETTER VERTICAL LINE

ˉ 02C9 MODIFIER LETTER MACRON

ˊ 02CA MODIFIER LETTER ACUTE ACCENT

ˋ 02CB MODIFIER LETTER GRAVE ACCENT

ˌ 02CC MODIFIER LETTER LOW VERTICAL LINE

ˍ 02CD MODIFIER LETTER LOW MACRON

ˎ 02CE MODIFIER LETTER LOW GRAVE ACCENT

ˏ 02CF MODIFIER LETTER LOW ACUTE ACCENT

ː 02D0 MODIFIER LETTER TRIANGULAR COLON

ˑ 02D1 MODIFIER LETTER HALF TRIANGULAR COLON

˒ 02D2 MODIFIER LETTER CENTRED RIGHT HALF RING

˓ 02D3 MODIFIER LETTER CENTRED LEFT HALF RING

˔ 02D4 MODIFIER LETTER UP TACK

˕ 02D5 MODIFIER LETTER DOWN TACK

˖ 02D6 MODIFIER LETTER PLUS SIGN

˗ 02D7 MODIFIER LETTER MINUS SIGN

˘ 02D8 BREVE

˙ 02D9 DOT ABOVE

˚ 02DA RING ABOVE

˛ 02DB OGONEK

˜ 02DC SMALL TILDE

˝ 02DD DOUBLE ACUTE ACCENT

˞ 02DE MODIFIER LETTER RHOTIC HOOK

˟ 02DF MODIFIER LETTER CROSS ACCENT

ˠ 02E0 MODIFIER LETTER SMALL GAMMA

ˡ 02E1 MODIFIER LETTER SMALL L

ˢ 02E2 MODIFIER LETTER SMALL S

ˣ 02E3 MODIFIER LETTER SMALL X

ˤ 02E4 MODIFIER LETTER SMALL REVERSED GLOTTAL STOP

˥ 02E5 MODIFIER LETTER EXTRA-HIGH TONE BAR

˦ 02E6 MODIFIER LETTER HIGH TONE BAR

˧ 02E7 MODIFIER LETTER MID TONE BAR

˨ 02E8 MODIFIER LETTER LOW TONE BAR

˩ 02E9 MODIFIER LETTER EXTRA-LOW TONE BAR

˪ 02EA MODIFIER LETTER YIN DEPARTING TONE MARK

˫ 02EB MODIFIER LETTER YANG DEPARTING TONE MARK

ˬ 02EC MODIFIER LETTER VOICING

˭ 02ED MODIFIER LETTER UNASPIRATED

ˮ 02EE MODIFIER LETTER DOUBLE APOSTROPHE

˯ 02EF MODIFIER LETTER LOW DOWN ARROWHEAD

˰ 02F0 MODIFIER LETTER LOW UP ARROWHEAD

˱ 02F1 MODIFIER LETTER LOW LEFT ARROWHEAD

˲ 02F2 MODIFIER LETTER LOW RIGHT ARROWHEAD

˳ 02F3 MODIFIER LETTER LOW RING

˴ 02F4 MODIFIER LETTER MIDDLE GRAVE ACCENT

˵ 02F5 MODIFIER LETTER MIDDLE DOUBLE GRAVE ACCENT

˶ 02F6 MODIFIER LETTER MIDDLE DOUBLE ACUTE ACCENT

˷ 02F7 MODIFIER LETTER LOW TILDE

˸ 02F8 MODIFIER LETTER RAISED COLON

˹ 02F9 MODIFIER LETTER BEGIN HIGH TONE

˺ 02FA MODIFIER LETTER END HIGH TONE

˻ 02FB MODIFIER LETTER BEGIN LOW TONE

˼ 02FC MODIFIER LETTER END LOW TONE

˽ 02FD MODIFIER LETTER SHELF

˾ 02FE MODIFIER LETTER OPEN SHELF

˿ 02FF MODIFIER LETTER LOW LEFT ARROW

ò 0300 GRAVE ACCENT

ó 0301 ACUTE ACCENT

ô 0302 CIRCUMFLEX ACCENT

õ 0303 TILDE

ō 0304 MACRON

o̅ 0305 OVERLINE

ŏ 0306 BREVE

ȯ 0307 DOT ABOVE

ö 0308 DIAERESIS

ỏ 0309 HOOK ABOVE

o̊ 030A RING ABOVE

ő 030B DOUBLE ACUTE ACCENT

ǒ 030C CARON

o̍ 030D VERTICAL LINE ABOVE

o̎ 030E DOUBLE VERTICAL LINE ABOVE

ȍ 030F DOUBLE GRAVE ACCENT

o̐ 0310 CANDRABINDU

ȏ 0311 INVERTED BREVE

o̒ 0312 TURNED COMMA ABOVE

o̓ 0313 COMMA ABOVE

o̔ 0314 REVERSED COMMA ABOVE

o̕ 0315 COMMA ABOVE RIGHT

o̖ 0316 GRAVE ACCENT BELOW

o̗ 0317 ACUTE ACCENT BELOW

o̘ 0318 LEFT TACK BELOW

o̙ 0319 RIGHT TACK BELOW

o̚ 031A LEFT ANGLE ABOVE

ơ 031B HORN

o̜ 031C LEFT HALF RING BELOW

o̝ 031D UP TACK BELOW

o̞ 031E DOWN TACK BELOW

o̟ 031F PLUS SIGN BELOW

o̠ 0320 MINUS SIGN BELOW

o̡ 0321 PALATALIZED HOOK BELOW

o̢ 0322 RETROFLEX HOOK BELOW

ọ 0323 DOT BELOW

o̤ 0324 DIAERESIS BELOW

o̥ 0325 RING BELOW

o̦ 0326 COMMA BELOW

o̧ 0327 CEDILLA

ǫ 0328 OGONEK

o̩ 0329 VERTICAL LINE BELOW

o̪ 032A BRIDGE BELOW

o̫ 032B INVERTED DOUBLE ARCH BELOW

o̬ 032C CARON BELOW

o̭ 032D CIRCUMFLEX ACCENT BELOW

o̮ 032E BREVE BELOW

o̯ 032F INVERTED BREVE BELOW

o̰ 0330 TILDE BELOW

o̱ 0331 MACRON BELOW

o̲ 0332 LOW LINE

o̳ 0333 DOUBLE LOW LINE

o̴ 0334 TILDE OVERLAY

o̵ 0335 SHORT STROKE OVERLAY

o̶ 0336 LONG STROKE OVERLAY

o̷ 0337 SHORT SOLIDUS OVERLAY

o̸ 0338 LONG SOLIDUS OVERLAY

o̹ 0339 RIGHT HALF RING BELOW

o̺ 033A INVERTED BRIDGE BELOW

o̻ 033B SQUARE BELOW

o̼ 033C SEAGULL BELOW

o̽ 033D X ABOVE

o̾ 033E VERTICAL TILDE

o̿ 033F DOUBLE OVERLINE

ò 0340 GRAVE TONE MARK

ó 0341 ACUTE TONE MARK

o͂ 0342 GREEK PERISPOMENI (combined with theta)

o̓ 0343 GREEK KORONIS (combined with theta)

ö́ 0344 GREEK DIALYTIKA TONOS (combined with theta)

oͅ 0345 GREEK YPOGEGRAMMENI (combined with theta)

o͆ 0346 BRIDGE ABOVE

o͇ 0347 EQUALS SIGN BELOW

o͈ 0348 DOUBLE VERTICAL LINE BELOW

o͉ 0349 LEFT ANGLE BELOW

o͊ 034A NOT TILDE ABOVE

o͋ 034B HOMOTHETIC ABOVE

o͌ 034C ALMOST EQUAL TO ABOVE

o͍ 034D LEFT RIGHT ARROW BELOW

o͎ 034E UPWARDS ARROW BELOW

o͏ 034F GRAPHEME JOINER

o͐ 0350 RIGHT ARROWHEAD ABOVE

o͑ 0351 LEFT HALF RING ABOVE

o͒ 0352 FERMATA

o͓ 0353 X BELOW

o͔ 0354 LEFT ARROWHEAD BELOW

o͕ 0355 RIGHT ARROWHEAD BELOW

o͖ 0356 RIGHT ARROWHEAD AND UP ARROWHEAD BELOW

o͗ 0357 RIGHT HALF RING ABOVE

o͘ 0358 DOT ABOVE RIGHT

o͙ 0359 ASTERISK BELOW

o͚ 035A DOUBLE RING BELOW

o͛ 035B ZIGZAG ABOVE

͜o 035C DOUBLE BREVE BELOW

͝o 035D DOUBLE BREVE

͞o 035E DOUBLE MACRON

͟o 035F DOUBLE MACRON BELOW

͠o 0360 DOUBLE TILDE

͡o 0361 DOUBLE INVERTED BREVE

͢o 0362 DOUBLE RIGHTWARDS ARROW BELOW

oͣ 0363 LATIN SMALL LETTER A

oͤ 0364 LATIN SMALL LETTER E

oͥ 0365 LATIN SMALL LETTER I

oͦ 0366 LATIN SMALL LETTER O

oͧ 0367 LATIN SMALL LETTER U

oͨ 0368 LATIN SMALL LETTER C

oͩ 0369 LATIN SMALL LETTER D

oͪ 036A LATIN SMALL LETTER H

oͫ 036B LATIN SMALL LETTER M

oͬ 036C LATIN SMALL LETTER R

oͭ 036D LATIN SMALL LETTER T

oͮ 036E LATIN SMALL LETTER V

oͯ 036F LATIN SMALL LETTER X

Ͱ 0370 GREEK CAPITAL LETTER HETA

ͱ 0371 GREEK SMALL LETTER HETA

Ͳ 0372 GREEK CAPITAL LETTER ARCHAIC SAMPI

ͳ 0373 GREEK SMALL LETTER ARCHAIC SAMPI

ʹ 0374 GREEK NUMERAL SIGN

͵ 0375 GREEK LOWER NUMERAL SIGN

Ͷ 0376 GREEK CAPITAL LETTER PAMPHYLIAN DIGAMMA

ͷ 0377 GREEK SMALL LETTER PAMPHYLIAN DIGAMMA

ͺ 037A GREEK YPOGEGRAMMENI

ͻ 037B GREEK SMALL REVERSED LUNATE SIGMA SYMBOL

ͼ 037C GREEK SMALL DOTTED LUNATE SIGMA SYMBOL

ͽ 037D GREEK SMALL REVERSED DOTTED LUNATE SIGMA SYMBOL

; 037E GREEK QUESTION MARK

΄ 0384 GREEK TONOS

΅ 0385 GREEK DIALYTIKA TONOS

Ά 0386 GREEK CAPITAL LETTER ALPHA WITH TONOS

· 0387 GREEK ANO TELEIA

Έ 0388 GREEK CAPITAL LETTER EPSILON WITH TONOS

Ή 0389 GREEK CAPITAL LETTER ETA WITH TONOS

Ί 038A GREEK CAPITAL LETTER IOTA WITH TONOS

Ό 038C GREEK CAPITAL LETTER OMICRON WITH TONOS

Ύ 038E GREEK CAPITAL LETTER UPSILON WITH TONOS

Ώ 038F GREEK CAPITAL LETTER OMEGA WITH TONOS

ΐ 0390 GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS

Α 0391 GREEK CAPITAL LETTER ALPHA

Β 0392 GREEK CAPITAL LETTER BETA

Γ 0393 GREEK CAPITAL LETTER GAMMA

Δ 0394 GREEK CAPITAL LETTER DELTA

Ε 0395 GREEK CAPITAL LETTER EPSILON

Ζ 0396 GREEK CAPITAL LETTER ZETA

Η 0397 GREEK CAPITAL LETTER ETA

Θ 0398 GREEK CAPITAL LETTER THETA

Ι 0399 GREEK CAPITAL LETTER IOTA

Κ 039A GREEK CAPITAL LETTER KAPPA

Λ 039B GREEK CAPITAL LETTER LAMBDA

Μ 039C GREEK CAPITAL LETTER MU

Ν 039D GREEK CAPITAL LETTER NU

Ξ 039E GREEK CAPITAL LETTER XI

Ο 039F GREEK CAPITAL LETTER OMICRON

Π 03A0 GREEK CAPITAL LETTER PI

Ρ 03A1 GREEK CAPITAL LETTER RHO

Σ 03A3 GREEK CAPITAL LETTER SIGMA

Τ 03A4 GREEK CAPITAL LETTER TAU

Υ 03A5 GREEK CAPITAL LETTER UPSILON

Φ 03A6 GREEK CAPITAL LETTER PHI

Χ 03A7 GREEK CAPITAL LETTER CHI

Ψ 03A8 GREEK CAPITAL LETTER PSI

Ω 03A9 GREEK CAPITAL LETTER OMEGA

Ϊ 03AA GREEK CAPITAL LETTER IOTA WITH DIALYTIKA

Ϋ 03AB GREEK CAPITAL LETTER UPSILON WITH DIALYTIKA

ά 03AC GREEK SMALL LETTER ALPHA WITH TONOS

έ 03AD GREEK SMALL LETTER EPSILON WITH TONOS

ή 03AE GREEK SMALL LETTER ETA WITH TONOS

ί 03AF GREEK SMALL LETTER IOTA WITH TONOS

ΰ 03B0 GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND TONOS

α 03B1 GREEK SMALL LETTER ALPHA

β 03B2 GREEK SMALL LETTER BETA

γ 03B3 GREEK SMALL LETTER GAMMA

δ 03B4 GREEK SMALL LETTER DELTA

ε 03B5 GREEK SMALL LETTER EPSILON

ζ 03B6 GREEK SMALL LETTER ZETA

η 03B7 GREEK SMALL LETTER ETA

θ 03B8 GREEK SMALL LETTER THETA

ι 03B9 GREEK SMALL LETTER IOTA

κ 03BA GREEK SMALL LETTER KAPPA

λ 03BB GREEK SMALL LETTER LAMBDA

μ 03BC GREEK SMALL LETTER MU

ν 03BD GREEK SMALL LETTER NU

ξ 03BE GREEK SMALL LETTER XI

ο 03BF GREEK SMALL LETTER OMICRON

π 03C0 GREEK SMALL LETTER PI

ρ 03C1 GREEK SMALL LETTER RHO

ς 03C2 GREEK SMALL LETTER FINAL SIGMA

σ 03C3 GREEK SMALL LETTER SIGMA

τ 03C4 GREEK SMALL LETTER TAU

υ 03C5 GREEK SMALL LETTER UPSILON

φ 03C6 GREEK SMALL LETTER PHI

χ 03C7 GREEK SMALL LETTER CHI

ψ 03C8 GREEK SMALL LETTER PSI

ω 03C9 GREEK SMALL LETTER OMEGA

ϊ 03CA GREEK SMALL LETTER IOTA WITH DIALYTIKA

ϋ 03CB GREEK SMALL LETTER UPSILON WITH DIALYTIKA

ό 03CC GREEK SMALL LETTER OMICRON WITH TONOS

ύ 03CD GREEK SMALL LETTER UPSILON WITH TONOS

ώ 03CE GREEK SMALL LETTER OMEGA WITH TONOS

Ϗ 03CF GREEK CAPITAL KAI SYMBOL

ϐ 03D0 GREEK BETA SYMBOL

ϑ 03D1 GREEK THETA SYMBOL

ϒ 03D2 GREEK UPSILON WITH HOOK SYMBOL

ϓ 03D3 GREEK UPSILON WITH ACUTE AND HOOK SYMBOL

ϔ 03D4 GREEK UPSILON WITH DIAERESIS AND HOOK SYMBOL

ϕ 03D5 GREEK PHI SYMBOL

ϖ 03D6 GREEK PI SYMBOL

ϗ 03D7 GREEK KAI SYMBOL

Ϙ 03D8 GREEK LETTER ARCHAIC KOPPA

ϙ 03D9 GREEK SMALL LETTER ARCHAIC KOPPA

Ϛ 03DA GREEK LETTER STIGMA

ϛ 03DB GREEK SMALL LETTER STIGMA

Ϝ 03DC GREEK LETTER DIGAMMA

ϝ 03DD GREEK SMALL LETTER DIGAMMA

Ϟ 03DE GREEK LETTER KOPPA

ϟ 03DF GREEK SMALL LETTER KOPPA

Ϡ 03E0 GREEK LETTER SAMPI

ϡ 03E1 GREEK SMALL LETTER SAMPI

Ϣ 03E2 COPTIC CAPITAL LETTER SHEI

ϣ 03E3 COPTIC SMALL LETTER SHEI

Ϥ 03E4 COPTIC CAPITAL LETTER FEI

ϥ 03E5 COPTIC SMALL LETTER FEI

Ϧ 03E6 COPTIC CAPITAL LETTER KHEI

ϧ 03E7 COPTIC SMALL LETTER KHEI

Ϩ 03E8 COPTIC CAPITAL LETTER HORI

ϩ 03E9 COPTIC SMALL LETTER HORI

Ϫ 03EA COPTIC CAPITAL LETTER GANGIA

ϫ 03EB COPTIC SMALL LETTER GANGIA

Ϭ 03EC COPTIC CAPITAL LETTER SHIMA

ϭ 03ED COPTIC SMALL LETTER SHIMA

Ϯ 03EE COPTIC CAPITAL LETTER DEI

ϯ 03EF COPTIC SMALL LETTER DEI

ϰ 03F0 GREEK KAPPA SYMBOL

ϱ 03F1 GREEK RHO SYMBOL

ϲ 03F2 GREEK LUNATE SIGMA SYMBOL

ϳ 03F3 GREEK LETTER YOT

ϴ 03F4 GREEK CAPITAL THETA SYMBOL

ϵ 03F5 GREEK LUNATE EPSILON SYMBOL

϶ 03F6 GREEK REVERSED LUNATE EPSILON SYMBOL

Ϸ 03F7 GREEK CAPITAL LETTER SHO

ϸ 03F8 GREEK SMALL LETTER SHO

Ϲ 03F9 GREEK CAPITAL LUNATE SIGMA SYMBOL

Ϻ 03FA GREEK CAPITAL LETTER SAN

ϻ 03FB GREEK SMALL LETTER SAN

ϼ 03FC GREEK RHO WITH STROKE SYMBOL

Ͻ 03FD GREEK CAPITAL REVERSED LUNATE SIGMA SYMBOL

Ͼ 03FE GREEK CAPITAL DOTTED LUNATE SIGMA SYMBOL

Ͽ 03FF GREEK CAPITAL REVERSED DOTTED LUNATE SIGMA SYMBOL

Ѐ 0400 CYRILLIC CAPITAL LETTER IE WITH GRAVE

Ё 0401 CYRILLIC CAPITAL LETTER IO

Ђ 0402 CYRILLIC CAPITAL LETTER DJE

Ѓ 0403 CYRILLIC CAPITAL LETTER GJE

Є 0404 CYRILLIC CAPITAL LETTER UKRAINIAN IE

Ѕ 0405 CYRILLIC CAPITAL LETTER DZE

І 0406 CYRILLIC CAPITAL LETTER BYELORUSSIAN-UKRAINIAN I

Ї 0407 CYRILLIC CAPITAL LETTER YI

Ј 0408 CYRILLIC CAPITAL LETTER JE

Љ 0409 CYRILLIC CAPITAL LETTER LJE

Њ 040A CYRILLIC CAPITAL LETTER NJE

Ћ 040B CYRILLIC CAPITAL LETTER TSHE

Ќ 040C CYRILLIC CAPITAL LETTER KJE

Ѝ 040D CYRILLIC CAPITAL LETTER I WITH GRAVE

Ў 040E CYRILLIC CAPITAL LETTER SHORT U

Џ 040F CYRILLIC CAPITAL LETTER DZHE

А 0410 CYRILLIC CAPITAL LETTER A

Б 0411 CYRILLIC CAPITAL LETTER BE

В 0412 CYRILLIC CAPITAL LETTER VE

Г 0413 CYRILLIC CAPITAL LETTER GHE

Д 0414 CYRILLIC CAPITAL LETTER DE

Е 0415 CYRILLIC CAPITAL LETTER IE

Ж 0416 CYRILLIC CAPITAL LETTER ZHE

З 0417 CYRILLIC CAPITAL LETTER ZE

И 0418 CYRILLIC CAPITAL LETTER I

Й 0419 CYRILLIC CAPITAL LETTER SHORT I

К 041A CYRILLIC CAPITAL LETTER KA

Л 041B CYRILLIC CAPITAL LETTER EL

М 041C CYRILLIC CAPITAL LETTER EM

Н 041D CYRILLIC CAPITAL LETTER EN

О 041E CYRILLIC CAPITAL LETTER O

П 041F CYRILLIC CAPITAL LETTER PE

Р 0420 CYRILLIC CAPITAL LETTER ER

С 0421 CYRILLIC CAPITAL LETTER ES

Т 0422 CYRILLIC CAPITAL LETTER TE

У 0423 CYRILLIC CAPITAL LETTER U

Ф 0424 CYRILLIC CAPITAL LETTER EF

Х 0425 CYRILLIC CAPITAL LETTER HA

Ц 0426 CYRILLIC CAPITAL LETTER TSE

Ч 0427 CYRILLIC CAPITAL LETTER CHE

Ш 0428 CYRILLIC CAPITAL LETTER SHA

Щ 0429 CYRILLIC CAPITAL LETTER SHCHA

Ъ 042A CYRILLIC CAPITAL LETTER HARD SIGN

Ы 042B CYRILLIC CAPITAL LETTER YERU

Ь 042C CYRILLIC CAPITAL LETTER SOFT SIGN

Э 042D CYRILLIC CAPITAL LETTER E

Ю 042E CYRILLIC CAPITAL LETTER YU

Я 042F CYRILLIC CAPITAL LETTER YA

а 0430 CYRILLIC SMALL LETTER A

б 0431 CYRILLIC SMALL LETTER BE

в 0432 CYRILLIC SMALL LETTER VE

г 0433 CYRILLIC SMALL LETTER GHE

д 0434 CYRILLIC SMALL LETTER DE

е 0435 CYRILLIC SMALL LETTER IE

ж 0436 CYRILLIC SMALL LETTER ZHE

з 0437 CYRILLIC SMALL LETTER ZE

и 0438 CYRILLIC SMALL LETTER I

й 0439 CYRILLIC SMALL LETTER SHORT I

к 043A CYRILLIC SMALL LETTER KA

л 043B CYRILLIC SMALL LETTER EL

м 043C CYRILLIC SMALL LETTER EM

н 043D CYRILLIC SMALL LETTER EN

о 043E CYRILLIC SMALL LETTER O

п 043F CYRILLIC SMALL LETTER PE

р 0440 CYRILLIC SMALL LETTER ER

с 0441 CYRILLIC SMALL LETTER ES

т 0442 CYRILLIC SMALL LETTER TE

у 0443 CYRILLIC SMALL LETTER U

ф 0444 CYRILLIC SMALL LETTER EF

х 0445 CYRILLIC SMALL LETTER HA

ц 0446 CYRILLIC SMALL LETTER TSE

ч 0447 CYRILLIC SMALL LETTER CHE

ш 0448 CYRILLIC SMALL LETTER SHA

щ 0449 CYRILLIC SMALL LETTER SHCHA

ъ 044A CYRILLIC SMALL LETTER HARD SIGN

ы 044B CYRILLIC SMALL LETTER YERU

ь 044C CYRILLIC SMALL LETTER SOFT SIGN

э 044D CYRILLIC SMALL LETTER E

ю 044E CYRILLIC SMALL LETTER YU

я 044F CYRILLIC SMALL LETTER YA

ѐ 0450 CYRILLIC SMALL LETTER IE WITH GRAVE

ё 0451 CYRILLIC SMALL LETTER IO

ђ 0452 CYRILLIC SMALL LETTER DJE

ѓ 0453 CYRILLIC SMALL LETTER GJE

є 0454 CYRILLIC SMALL LETTER UKRAINIAN IE

ѕ 0455 CYRILLIC SMALL LETTER DZE

і 0456 CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I

ї 0457 CYRILLIC SMALL LETTER YI

ј 0458 CYRILLIC SMALL LETTER JE

љ 0459 CYRILLIC SMALL LETTER LJE

њ 045A CYRILLIC SMALL LETTER NJE

ћ 045B CYRILLIC SMALL LETTER TSHE

ќ 045C CYRILLIC SMALL LETTER KJE

ѝ 045D CYRILLIC SMALL LETTER I WITH GRAVE

ў 045E CYRILLIC SMALL LETTER SHORT U

џ 045F CYRILLIC SMALL LETTER DZHE

Ѡ 0460 CYRILLIC CAPITAL LETTER OMEGA

ѡ 0461 CYRILLIC SMALL LETTER OMEGA

Ѣ 0462 CYRILLIC CAPITAL LETTER YAT

ѣ 0463 CYRILLIC SMALL LETTER YAT

Ѥ 0464 CYRILLIC CAPITAL LETTER IOTIFIED E

ѥ 0465 CYRILLIC SMALL LETTER IOTIFIED E

Ѧ 0466 CYRILLIC CAPITAL LETTER LITTLE YUS

ѧ 0467 CYRILLIC SMALL LETTER LITTLE YUS

Ѩ 0468 CYRILLIC CAPITAL LETTER IOTIFIED LITTLE YUS

ѩ 0469 CYRILLIC SMALL LETTER IOTIFIED LITTLE YUS

Ѫ 046A CYRILLIC CAPITAL LETTER BIG YUS

ѫ 046B CYRILLIC SMALL LETTER BIG YUS

Ѭ 046C CYRILLIC CAPITAL LETTER IOTIFIED BIG YUS

ѭ 046D CYRILLIC SMALL LETTER IOTIFIED BIG YUS

Ѯ 046E CYRILLIC CAPITAL LETTER KSI

ѯ 046F CYRILLIC SMALL LETTER KSI

Ѱ 0470 CYRILLIC CAPITAL LETTER PSI

ѱ 0471 CYRILLIC SMALL LETTER PSI

Ѳ 0472 CYRILLIC CAPITAL LETTER FITA

ѳ 0473 CYRILLIC SMALL LETTER FITA

Ѵ 0474 CYRILLIC CAPITAL LETTER IZHITSA

ѵ 0475 CYRILLIC SMALL LETTER IZHITSA

Ѷ 0476 CYRILLIC CAPITAL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT

ѷ 0477 CYRILLIC SMALL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT

Ѹ 0478 CYRILLIC CAPITAL LETTER UK

ѹ 0479 CYRILLIC SMALL LETTER UK

Ѻ 047A CYRILLIC CAPITAL LETTER ROUND OMEGA

ѻ 047B CYRILLIC SMALL LETTER ROUND OMEGA

Ѽ 047C CYRILLIC CAPITAL LETTER OMEGA WITH TITLO

ѽ 047D CYRILLIC SMALL LETTER OMEGA WITH TITLO

Ѿ 047E CYRILLIC CAPITAL LETTER OT

ѿ 047F CYRILLIC SMALL LETTER OT

Ҁ 0480 CYRILLIC CAPITAL LETTER KOPPA

ҁ 0481 CYRILLIC SMALL LETTER KOPPA

҂ 0482 CYRILLIC THOUSANDS SIGN

о҃ 0483 COMBINING CYRILLIC TITLO (combined with о)

о҄ 0484 COMBINING CYRILLIC PALATALIZATION (combined with о)

о҅ 0485 COMBINING CYRILLIC DASIA PNEUMATA (combined with о)

о҆ 0486 COMBINING CYRILLIC PSILI PNEUMATA (combined with о)

о҇ 0487 COMBINING CYRILLIC POKRYTIE (combined with о)

о҈ 0488 COMBINING CYRILLIC HUNDRED THOUSANDS SIGN (combined with о)

о҉ 0489 COMBINING CYRILLIC MILLIONS SIGN (combined with о)

Ҋ 048A CYRILLIC CAPITAL LETTER SHORT I WITH TAIL

ҋ 048B CYRILLIC SMALL LETTER SHORT I WITH TAIL

Ҍ 048C CYRILLIC CAPITAL LETTER SEMISOFT SIGN

ҍ 048D CYRILLIC SMALL LETTER SEMISOFT SIGN

Ҏ 048E CYRILLIC CAPITAL LETTER ER WITH TICK

ҏ 048F CYRILLIC SMALL LETTER ER WITH TICK

Ґ 0490 CYRILLIC CAPITAL LETTER GHE WITH UPTURN

ґ 0491 CYRILLIC SMALL LETTER GHE WITH UPTURN

Ғ 0492 CYRILLIC CAPITAL LETTER GHE WITH STROKE

ғ 0493 CYRILLIC SMALL LETTER GHE WITH STROKE

Ҕ 0494 CYRILLIC CAPITAL LETTER GHE WITH MIDDLE HOOK

ҕ 0495 CYRILLIC SMALL LETTER GHE WITH MIDDLE HOOK

Җ 0496 CYRILLIC CAPITAL LETTER ZHE WITH DESCENDER

җ 0497 CYRILLIC SMALL LETTER ZHE WITH DESCENDER

Ҙ 0498 CYRILLIC CAPITAL LETTER ZE WITH DESCENDER

ҙ 0499 CYRILLIC SMALL LETTER ZE WITH DESCENDER

Қ 049A CYRILLIC CAPITAL LETTER KA WITH DESCENDER

қ 049B CYRILLIC SMALL LETTER KA WITH DESCENDER

Ҝ 049C CYRILLIC CAPITAL LETTER KA WITH VERTICAL STROKE

ҝ 049D CYRILLIC SMALL LETTER KA WITH VERTICAL STROKE

Ҟ 049E CYRILLIC CAPITAL LETTER KA WITH STROKE

ҟ 049F CYRILLIC SMALL LETTER KA WITH STROKE

Ҡ 04A0 CYRILLIC CAPITAL LETTER BASHKIR KA

ҡ 04A1 CYRILLIC SMALL LETTER BASHKIR KA

Ң 04A2 CYRILLIC CAPITAL LETTER EN WITH DESCENDER

ң 04A3 CYRILLIC SMALL LETTER EN WITH DESCENDER

Ҥ 04A4 CYRILLIC CAPITAL LIGATURE EN GHE

ҥ 04A5 CYRILLIC SMALL LIGATURE EN GHE

Ҧ 04A6 CYRILLIC CAPITAL LETTER PE WITH MIDDLE HOOK

ҧ 04A7 CYRILLIC SMALL LETTER PE WITH MIDDLE HOOK

Ҩ 04A8 CYRILLIC CAPITAL LETTER ABKHASIAN HA

ҩ 04A9 CYRILLIC SMALL LETTER ABKHASIAN HA

Ҫ 04AA CYRILLIC CAPITAL LETTER ES WITH DESCENDER

ҫ 04AB CYRILLIC SMALL LETTER ES WITH DESCENDER

Ҭ 04AC CYRILLIC CAPITAL LETTER TE WITH DESCENDER

ҭ 04AD CYRILLIC SMALL LETTER TE WITH DESCENDER

Ү 04AE CYRILLIC CAPITAL LETTER STRAIGHT U

ү 04AF CYRILLIC SMALL LETTER STRAIGHT U

Ұ 04B0 CYRILLIC CAPITAL LETTER STRAIGHT U WITH STROKE

ұ 04B1 CYRILLIC SMALL LETTER STRAIGHT U WITH STROKE

Ҳ 04B2 CYRILLIC CAPITAL LETTER HA WITH DESCENDER

ҳ 04B3 CYRILLIC SMALL LETTER HA WITH DESCENDER

Ҵ 04B4 CYRILLIC CAPITAL LIGATURE TE TSE

ҵ 04B5 CYRILLIC SMALL LIGATURE TE TSE

Ҷ 04B6 CYRILLIC CAPITAL LETTER CHE WITH DESCENDER

ҷ 04B7 CYRILLIC SMALL LETTER CHE WITH DESCENDER

Ҹ 04B8 CYRILLIC CAPITAL LETTER CHE WITH VERTICAL STROKE

ҹ 04B9 CYRILLIC SMALL LETTER CHE WITH VERTICAL STROKE

Һ 04BA CYRILLIC CAPITAL LETTER SHHA

һ 04BB CYRILLIC SMALL LETTER SHHA

Ҽ 04BC CYRILLIC CAPITAL LETTER ABKHASIAN CHE

ҽ 04BD CYRILLIC SMALL LETTER ABKHASIAN CHE

Ҿ 04BE CYRILLIC CAPITAL LETTER ABKHASIAN CHE WITH DESCENDER

ҿ 04BF CYRILLIC SMALL LETTER ABKHASIAN CHE WITH DESCENDER

Ӏ 04C0 CYRILLIC LETTER PALOCHKA

Ӂ 04C1 CYRILLIC CAPITAL LETTER ZHE WITH BREVE

ӂ 04C2 CYRILLIC SMALL LETTER ZHE WITH BREVE

Ӄ 04C3 CYRILLIC CAPITAL LETTER KA WITH HOOK

ӄ 04C4 CYRILLIC SMALL LETTER KA WITH HOOK

Ӆ 04C5 CYRILLIC CAPITAL LETTER EL WITH TAIL

ӆ 04C6 CYRILLIC SMALL LETTER EL WITH TAIL

Ӈ 04C7 CYRILLIC CAPITAL LETTER EN WITH HOOK

ӈ 04C8 CYRILLIC SMALL LETTER EN WITH HOOK

Ӊ 04C9 CYRILLIC CAPITAL LETTER EN WITH TAIL

ӊ 04CA CYRILLIC SMALL LETTER EN WITH TAIL

Ӌ 04CB CYRILLIC CAPITAL LETTER KHAKASSIAN CHE

ӌ 04CC CYRILLIC SMALL LETTER KHAKASSIAN CHE

Ӎ 04CD CYRILLIC CAPITAL LETTER EM WITH TAIL

ӎ 04CE CYRILLIC SMALL LETTER EM WITH TAIL

ӏ 04CF CYRILLIC SMALL LETTER PALOCHKA

Ӑ 04D0 CYRILLIC CAPITAL LETTER A WITH BREVE

ӑ 04D1 CYRILLIC SMALL LETTER A WITH BREVE

Ӓ 04D2 CYRILLIC CAPITAL LETTER A WITH DIAERESIS

ӓ 04D3 CYRILLIC SMALL LETTER A WITH DIAERESIS

Ӕ 04D4 CYRILLIC CAPITAL LIGATURE A IE

ӕ 04D5 CYRILLIC SMALL LIGATURE A IE

Ӗ 04D6 CYRILLIC CAPITAL LETTER IE WITH BREVE

ӗ 04D7 CYRILLIC SMALL LETTER IE WITH BREVE

Ә 04D8 CYRILLIC CAPITAL LETTER SCHWA

ә 04D9 CYRILLIC SMALL LETTER SCHWA

Ӛ 04DA CYRILLIC CAPITAL LETTER SCHWA WITH DIAERESIS

ӛ 04DB CYRILLIC SMALL LETTER SCHWA WITH DIAERESIS

Ӝ 04DC CYRILLIC CAPITAL LETTER ZHE WITH DIAERESIS

ӝ 04DD CYRILLIC SMALL LETTER ZHE WITH DIAERESIS

Ӟ 04DE CYRILLIC CAPITAL LETTER ZE WITH DIAERESIS

ӟ 04DF CYRILLIC SMALL LETTER ZE WITH DIAERESIS

Ӡ 04E0 CYRILLIC CAPITAL LETTER ABKHASIAN DZE

ӡ 04E1 CYRILLIC SMALL LETTER ABKHASIAN DZE

Ӣ 04E2 CYRILLIC CAPITAL LETTER I WITH MACRON

ӣ 04E3 CYRILLIC SMALL LETTER I WITH MACRON

Ӥ 04E4 CYRILLIC CAPITAL LETTER I WITH DIAERESIS

ӥ 04E5 CYRILLIC SMALL LETTER I WITH DIAERESIS

Ӧ 04E6 CYRILLIC CAPITAL LETTER O WITH DIAERESIS

ӧ 04E7 CYRILLIC SMALL LETTER O WITH DIAERESIS

Ө 04E8 CYRILLIC CAPITAL LETTER BARRED O

ө 04E9 CYRILLIC SMALL LETTER BARRED O

Ӫ 04EA CYRILLIC CAPITAL LETTER BARRED O WITH DIAERESIS

ӫ 04EB CYRILLIC SMALL LETTER BARRED O WITH DIAERESIS

Ӭ 04EC CYRILLIC CAPITAL LETTER E WITH DIAERESIS

ӭ 04ED CYRILLIC SMALL LETTER E WITH DIAERESIS

Ӯ 04EE CYRILLIC CAPITAL LETTER U WITH MACRON

ӯ 04EF CYRILLIC SMALL LETTER U WITH MACRON

Ӱ 04F0 CYRILLIC CAPITAL LETTER U WITH DIAERESIS

ӱ 04F1 CYRILLIC SMALL LETTER U WITH DIAERESIS

Ӳ 04F2 CYRILLIC CAPITAL LETTER U WITH DOUBLE ACUTE

ӳ 04F3 CYRILLIC SMALL LETTER U WITH DOUBLE ACUTE

Ӵ 04F4 CYRILLIC CAPITAL LETTER CHE WITH DIAERESIS

ӵ 04F5 CYRILLIC SMALL LETTER CHE WITH DIAERESIS

Ӷ 04F6 CYRILLIC CAPITAL LETTER GHE WITH DESCENDER

ӷ 04F7 CYRILLIC SMALL LETTER GHE WITH DESCENDER

Ӹ 04F8 CYRILLIC CAPITAL LETTER YERU WITH DIAERESIS

ӹ 04F9 CYRILLIC SMALL LETTER YERU WITH DIAERESIS

Ӻ 04FA CYRILLIC CAPITAL LETTER GHE WITH STROKE AND HOOK

ӻ 04FB CYRILLIC SMALL LETTER GHE WITH STROKE AND HOOK

Ӽ 04FC CYRILLIC CAPITAL LETTER HA WITH HOOK

ӽ 04FD CYRILLIC SMALL LETTER HA WITH HOOK

Ӿ 04FE CYRILLIC CAPITAL LETTER HA WITH STROKE

ӿ 04FF CYRILLIC SMALL LETTER HA WITH STROKE

Ԁ 0500 CYRILLIC CAPITAL LETTER KOMI DE

ԁ 0501 CYRILLIC SMALL LETTER KOMI DE

Ԃ 0502 CYRILLIC CAPITAL LETTER KOMI DJE

ԃ 0503 CYRILLIC SMALL LETTER KOMI DJE

Ԅ 0504 CYRILLIC CAPITAL LETTER KOMI ZJE

ԅ 0505 CYRILLIC SMALL LETTER KOMI ZJE

Ԇ 0506 CYRILLIC CAPITAL LETTER KOMI DZJE

ԇ 0507 CYRILLIC SMALL LETTER KOMI DZJE

Ԉ 0508 CYRILLIC CAPITAL LETTER KOMI LJE

ԉ 0509 CYRILLIC SMALL LETTER KOMI LJE

Ԋ 050A CYRILLIC CAPITAL LETTER KOMI NJE

ԋ 050B CYRILLIC SMALL LETTER KOMI NJE

Ԍ 050C CYRILLIC CAPITAL LETTER KOMI SJE

ԍ 050D CYRILLIC SMALL LETTER KOMI SJE

Ԏ 050E CYRILLIC CAPITAL LETTER KOMI TJE

ԏ 050F CYRILLIC SMALL LETTER KOMI TJE

Ԕ 0514 CYRILLIC CAPITAL LETTER LHA

ԕ 0515 CYRILLIC SMALL LETTER LHA

Ԗ 0516 CYRILLIC CAPITAL LETTER RHA

ԗ 0517 CYRILLIC SMALL LETTER RHA

Ԙ 0518 CYRILLIC CAPITAL LETTER YAE

ԙ 0519 CYRILLIC SMALL LETTER YAE

Ԛ 051A CYRILLIC CAPITAL LETTER QA

ԛ 051B CYRILLIC SMALL LETTER QA

Ԝ 051C CYRILLIC CAPITAL LETTER WE

ԝ 051D CYRILLIC SMALL LETTER WE

Ԟ 051E CYRILLIC CAPITAL LETTER ALEUT KA

ԟ 051F CYRILLIC SMALL LETTER ALEUT KA

Ԡ 0520 CYRILLIC CAPITAL LETTER EL WITH MIDDLE HOOK

ԡ 0521 CYRILLIC SMALL LETTER EL WITH MIDDLE HOOK

Ԣ 0522 CYRILLIC CAPITAL LETTER EN WITH MIDDLE HOOK

ԣ 0523 CYRILLIC SMALL LETTER EN WITH MIDDLE HOOK

Ԥ 0524 CYRILLIC CAPITAL LETTER PE WITH DESCENDER

ԥ 0525 CYRILLIC SMALL LETTER PE WITH DESCENDER

Ԧ 0526 CYRILLIC CAPITAL LETTER SHHA WITH DESCENDER

ԧ 0527 CYRILLIC SMALL LETTER SHHA WITH DESCENDER

Ԩ 0528

ԩ 0529

Ԫ 052A

ԫ 052B

Ԭ 052C

ԭ 052D

Ԯ 052E

ԯ 052F

  2000 EN QUAD

  2001 EM QUAD

  2002 EN SPACE

  2003 EM SPACE

  2004 THREE-PER-EM SPACE

  2005 FOUR-PER-EM SPACE

  2006 SIX-PER-EM SPACE

  2007 FIGURE SPACE

  2008 PUNCTUATION SPACE

  2009 THIN SPACE

  200A HAIR SPACE

​ 200B ZERO WIDTH SPACE

‌ 200C ZERO WIDTH NON-JOINER

‍ 200D ZERO WIDTH JOINER

‎ 200E LEFT-TO-RIGHT MARK

‏ 200F RIGHT-TO-LEFT MARK

‐ 2010 HYPHEN

‑ 2011 NON-BREAKING HYPHEN

‒ 2012 FIGURE DASH

– 2013 EN DASH

— 2014 EM DASH

― 2015 HORIZONTAL BAR

‖ 2016 DOUBLE VERTICAL LINE

‗ 2017 DOUBLE LOW LINE

‘ 2018 LEFT SINGLE QUOTATION MARK

’ 2019 RIGHT SINGLE QUOTATION MARK

‚ 201A SINGLE LOW-9 QUOTATION MARK

‛ 201B SINGLE HIGH-REVERSED-9 QUOTATION MARK

“ 201C LEFT DOUBLE QUOTATION MARK

” 201D RIGHT DOUBLE QUOTATION MARK

„ 201E DOUBLE LOW-9 QUOTATION MARK

‟ 201F DOUBLE HIGH-REVERSED-9 QUOTATION MARK

† 2020 DAGGER

‡ 2021 DOUBLE DAGGER

• 2022 BULLET

‣ 2023 TRIANGULAR BULLET

․ 2024 ONE DOT LEADER

‥ 2025 TWO DOT LEADER

… 2026 HORIZONTAL ELLIPSIS

‧ 2027 HYPHENATION POINT

  2028 LINE SEPARATOR

  2029 PARAGRAPH SEPARATOR

‪ 202A LEFT-TO-RIGHT EMBEDDING

‫ 202B RIGHT-TO-LEFT EMBEDDING

‬ 202C POP DIRECTIONAL FORMATTING

‭ 202D LEFT-TO-RIGHT OVERRIDE

‮ 202E RIGHT-TO-LEFT OVERRIDE

  202F NARROW NON-BREAK SPACE

‰ 2030 PER MILLE SIGN

‱ 2031 PER TEN THOUSAND SIGN

′ 2032 PRIME

″ 2033 DOUBLE PRIME

‴ 2034 TRIPLE PRIME

‵ 2035 REVERSED PRIME

‶ 2036 REVERSED DOUBLE PRIME

‷ 2037 REVERSED TRIPLE PRIME

‸ 2038 CARET

‹ 2039 SINGLE LEFT-POINTING ANGLE QUOTATION MARK

› 203A SINGLE RIGHT-POINTING ANGLE QUOTATION MARK

※ 203B REFERENCE MARK

‼ 203C DOUBLE EXCLAMATION MARK

‽ 203D INTERROBANG

‾ 203E OVERLINE

‿ 203F UNDERTIE

⁀ 2040 CHARACTER TIE

⁁ 2041 CARET INSERTION POINT

⁂ 2042 ASTERISM

⁃ 2043 HYPHEN BULLET

⁄ 2044 FRACTION SLASH

⁅ 2045 LEFT SQUARE BRACKET WITH QUILL

⁆ 2046 RIGHT SQUARE BRACKET WITH QUILL

⁇ 2047 DOUBLE QUESTION MARK

⁈ 2048 QUESTION EXCLAMATION MARK

⁉ 2049 EXCLAMATION QUESTION MARK

⁊ 204A TIRONIAN SIGN ET

⁋ 204B REVERSED PILCROW SIGN

⁌ 204C BLACK LEFTWARDS BULLET

⁍ 204D BLACK RIGHTWARDS BULLET

⁎ 204E LOW ASTERISK

⁏ 204F REVERSED SEMICOLON

⁐ 2050 CLOSE UP

⁑ 2051 TWO ASTERISKS ALIGNED VERTICALLY

⁒ 2052 COMMERCIAL MINUS SIGN

⁓ 2053 SWUNG DASH

⁔ 2054 INVERTED UNDERTIE

⁕ 2055 FLOWER PUNCTUATION MARK

⁖ 2056 THREE DOT PUNCTUATION

⁗ 2057 QUADRUPLE PRIME

⁘ 2058 FOUR DOT PUNCTUATION

⁙ 2059 FIVE DOT PUNCTUATION

⁚ 205A TWO DOT PUNCTUATION

⁛ 205B FOUR DOT MARK

⁜ 205C DOTTED CROSS

⁝ 205D TRICOLON

⁞ 205E VERTICAL FOUR DOTS

  205F MEDIUM MATHEMATICAL SPACE

⁠ 2060 WORD JOINER

⁡ 2061 FUNCTION APPLICATION

⁢ 2062 INVISIBLE TIMES

⁣ 2063 INVISIBLE SEPARATOR

⁤ 2064 INVISIBLE PLUS

⁦ 2066 LEFT-TO-RIGHT ISOLATE

⁧ 2067 RIGHT-TO-LEFT ISOLATE

⁨ 2068 FIRST STRONG ISOLATE

⁩ 2069 POP DIRECTIONAL ISOLATE

⁪ 206A INHIBIT SYMMETRIC SWAPPING

⁫ 206B ACTIVATE SYMMETRIC SWAPPING

⁬ 206C INHIBIT ARABIC FORM SHAPING

⁭ 206D ACTIVATE ARABIC FORM SHAPING

⁮ 206E NATIONAL DIGIT SHAPES

⁯ 206F NOMINAL DIGIT SHAPES

₠ 20A0 EURO-CURRENCY SIGN

₡ 20A1 COLON SIGN

₢ 20A2 CRUZEIRO SIGN

₣ 20A3 FRENCH FRANC SIGN

₤ 20A4 LIRA SIGN

₥ 20A5 MILL SIGN

₦ 20A6 NAIRA SIGN

₧ 20A7 PESETA SIGN

₨ 20A8 RUPEE SIGN

₩ 20A9 WON SIGN

₪ 20AA NEW SHEQEL SIGN

₫ 20AB DONG SIGN

€ 20AC EURO SIGN

₭ 20AD KIP SIGN

₮ 20AE TUGRIK SIGN

₯ 20AF DRACHMA SIGN

₰ 20B0 GERMAN PENNY SYMBOL

₱ 20B1 PESO SIGN

₲ 20B2 GUARANI SIGN

₳ 20B3 AUSTRAL SIGN

₴ 20B4 HRYVNIA SIGN

₵ 20B5 CEDI SIGN

₶ 20B6 LIVRE TOURNOIS SIGN

₷ 20B7 SPESMILO SIGN

₸ 20B8 TENGE SIGN

₹ 20B9 INDIAN RUPEE SIGN

℀ 2100 ACCOUNT OF

℁ 2101 ADDRESSED TO THE SUBJECT

ℂ 2102 DOUBLE-STRUCK CAPITAL C

℃ 2103 DEGREE CELSIUS

℄ 2104 CENTRE LINE SYMBOL

℅ 2105 CARE OF

℆ 2106 CADA UNA

ℇ 2107 EULER CONSTANT

℈ 2108 SCRUPLE

℉ 2109 DEGREE FAHRENHEIT

ℊ 210A SCRIPT SMALL G

ℋ 210B SCRIPT CAPITAL H

ℌ 210C BLACK-LETTER CAPITAL H

ℍ 210D DOUBLE-STRUCK CAPITAL H

ℎ 210E PLANCK CONSTANT

ℏ 210F PLANCK CONSTANT OVER TWO PI

ℐ 2110 SCRIPT CAPITAL I

ℑ 2111 BLACK-LETTER CAPITAL I

ℒ 2112 SCRIPT CAPITAL L

ℓ 2113 SCRIPT SMALL L

℔ 2114 L B BAR SYMBOL

ℕ 2115 DOUBLE-STRUCK CAPITAL N

№ 2116 NUMERO SIGN

℗ 2117 SOUND RECORDING COPYRIGHT

℘ 2118 SCRIPT CAPITAL P

ℙ 2119 DOUBLE-STRUCK CAPITAL P

ℚ 211A DOUBLE-STRUCK CAPITAL Q

ℛ 211B SCRIPT CAPITAL R

ℜ 211C BLACK-LETTER CAPITAL R

ℝ 211D DOUBLE-STRUCK CAPITAL R

℞ 211E PRESCRIPTION TAKE

℟ 211F RESPONSE

℠ 2120 SERVICE MARK

℡ 2121 TELEPHONE SIGN

™ 2122 TRADE MARK SIGN

℣ 2123 VERSICLE

ℤ 2124 DOUBLE-STRUCK CAPITAL Z

℥ 2125 OUNCE SIGN

Ω 2126 OHM SIGN

℧ 2127 INVERTED OHM SIGN

ℨ 2128 BLACK-LETTER CAPITAL Z

℩ 2129 TURNED GREEK SMALL LETTER IOTA

K 212A KELVIN SIGN

Å 212B ANGSTROM SIGN

ℬ 212C SCRIPT CAPITAL B

ℭ 212D BLACK-LETTER CAPITAL C

℮ 212E ESTIMATED SYMBOL

ℯ 212F SCRIPT SMALL E

ℰ 2130 SCRIPT CAPITAL E

ℱ 2131 SCRIPT CAPITAL F

Ⅎ 2132 TURNED CAPITAL F

ℳ 2133 SCRIPT CAPITAL M

ℴ 2134 SCRIPT SMALL O

ℵ 2135 ALEF SYMBOL

ℶ 2136 BET SYMBOL

ℷ 2137 GIMEL SYMBOL

ℸ 2138 DALET SYMBOL

ℹ 2139 INFORMATION SOURCE

℺ 213A ROTATED CAPITAL Q

℻ 213B FACSIMILE SIGN

ℼ 213C DOUBLE-STRUCK SMALL PI

ℽ 213D DOUBLE-STRUCK SMALL GAMMA

ℾ 213E DOUBLE-STRUCK CAPITAL GAMMA

ℿ 213F DOUBLE-STRUCK CAPITAL PI

⅀ 2140 DOUBLE-STRUCK N-ARY SUMMATION

⅁ 2141 TURNED SANS-SERIF CAPITAL G

⅂ 2142 TURNED SANS-SERIF CAPITAL L

⅃ 2143 REVERSED SANS-SERIF CAPITAL L

⅄ 2144 TURNED SANS-SERIF CAPITAL Y

ⅅ 2145 DOUBLE-STRUCK ITALIC CAPITAL D

ⅆ 2146 DOUBLE-STRUCK ITALIC SMALL D

ⅇ 2147 DOUBLE-STRUCK ITALIC SMALL E

ⅈ 2148 DOUBLE-STRUCK ITALIC SMALL I

ⅉ 2149 DOUBLE-STRUCK ITALIC SMALL J

⅊ 214A PROPERTY LINE

⅋ 214B TURNED AMPERSAND

⅌ 214C PER SIGN

⅍ 214D AKTIESELSKAB

ⅎ 214E TURNED SMALL F

⅏ 214F SYMBOL FOR SAMARITAN SOURCE

← 2190 LEFTWARDS ARROW

↑ 2191 UPWARDS ARROW

→ 2192 RIGHTWARDS ARROW

↓ 2193 DOWNWARDS ARROW

↔ 2194 LEFT RIGHT ARROW

↕ 2195 UP DOWN ARROW

↖ 2196 NORTH WEST ARROW

↗ 2197 NORTH EAST ARROW

↘ 2198 SOUTH EAST ARROW

↙ 2199 SOUTH WEST ARROW

↚ 219A LEFTWARDS ARROW WITH STROKE

↛ 219B RIGHTWARDS ARROW WITH STROKE

↜ 219C LEFTWARDS WAVE ARROW

↝ 219D RIGHTWARDS WAVE ARROW

↞ 219E LEFTWARDS TWO HEADED ARROW

↟ 219F UPWARDS TWO HEADED ARROW

↠ 21A0 RIGHTWARDS TWO HEADED ARROW

↡ 21A1 DOWNWARDS TWO HEADED ARROW

↢ 21A2 LEFTWARDS ARROW WITH TAIL

↣ 21A3 RIGHTWARDS ARROW WITH TAIL

↤ 21A4 LEFTWARDS ARROW FROM BAR

↥ 21A5 UPWARDS ARROW FROM BAR

↦ 21A6 RIGHTWARDS ARROW FROM BAR

↧ 21A7 DOWNWARDS ARROW FROM BAR

↨ 21A8 UP DOWN ARROW WITH BASE

↩ 21A9 LEFTWARDS ARROW WITH HOOK

↪ 21AA RIGHTWARDS ARROW WITH HOOK

↫ 21AB LEFTWARDS ARROW WITH LOOP

↬ 21AC RIGHTWARDS ARROW WITH LOOP

↭ 21AD LEFT RIGHT WAVE ARROW

↮ 21AE LEFT RIGHT ARROW WITH STROKE

↯ 21AF DOWNWARDS ZIGZAG ARROW

↰ 21B0 UPWARDS ARROW WITH TIP LEFTWARDS

↱ 21B1 UPWARDS ARROW WITH TIP RIGHTWARDS

↲ 21B2 DOWNWARDS ARROW WITH TIP LEFTWARDS

↳ 21B3 DOWNWARDS ARROW WITH TIP RIGHTWARDS

↴ 21B4 RIGHTWARDS ARROW WITH CORNER DOWNWARDS

↵ 21B5 DOWNWARDS ARROW WITH CORNER LEFTWARDS

↶ 21B6 ANTICLOCKWISE TOP SEMICIRCLE ARROW

↷ 21B7 CLOCKWISE TOP SEMICIRCLE ARROW

↸ 21B8 NORTH WEST ARROW TO LONG BAR

↹ 21B9 LEFTWARDS ARROW TO BAR OVER RIGHTWARDS ARROW TO BAR

↺ 21BA ANTICLOCKWISE OPEN CIRCLE ARROW

↻ 21BB CLOCKWISE OPEN CIRCLE ARROW

↼ 21BC LEFTWARDS HARPOON WITH BARB UPWARDS

↽ 21BD LEFTWARDS HARPOON WITH BARB DOWNWARDS

↾ 21BE UPWARDS HARPOON WITH BARB RIGHTWARDS

↿ 21BF UPWARDS HARPOON WITH BARB LEFTWARDS

⇀ 21C0 RIGHTWARDS HARPOON WITH BARB UPWARDS

⇁ 21C1 RIGHTWARDS HARPOON WITH BARB DOWNWARDS

⇂ 21C2 DOWNWARDS HARPOON WITH BARB RIGHTWARDS

⇃ 21C3 DOWNWARDS HARPOON WITH BARB LEFTWARDS

⇄ 21C4 RIGHTWARDS ARROW OVER LEFTWARDS ARROW

⇅ 21C5 UPWARDS ARROW LEFTWARDS OF DOWNWARDS ARROW

⇆ 21C6 LEFTWARDS ARROW OVER RIGHTWARDS ARROW

⇇ 21C7 LEFTWARDS PAIRED ARROWS

⇈ 21C8 UPWARDS PAIRED ARROWS

⇉ 21C9 RIGHTWARDS PAIRED ARROWS

⇊ 21CA DOWNWARDS PAIRED ARROWS

⇋ 21CB LEFTWARDS HARPOON OVER RIGHTWARDS HARPOON

⇌ 21CC RIGHTWARDS HARPOON OVER LEFTWARDS HARPOON

⇍ 21CD LEFTWARDS DOUBLE ARROW WITH STROKE

⇎ 21CE LEFT RIGHT DOUBLE ARROW WITH STROKE

⇏ 21CF RIGHTWARDS DOUBLE ARROW WITH STROKE

⇐ 21D0 LEFTWARDS DOUBLE ARROW

⇑ 21D1 UPWARDS DOUBLE ARROW

⇒ 21D2 RIGHTWARDS DOUBLE ARROW

⇓ 21D3 DOWNWARDS DOUBLE ARROW

⇔ 21D4 LEFT RIGHT DOUBLE ARROW

⇕ 21D5 UP DOWN DOUBLE ARROW

⇖ 21D6 NORTH WEST DOUBLE ARROW

⇗ 21D7 NORTH EAST DOUBLE ARROW

⇘ 21D8 SOUTH EAST DOUBLE ARROW

⇙ 21D9 SOUTH WEST DOUBLE ARROW

⇚ 21DA LEFTWARDS TRIPLE ARROW

⇛ 21DB RIGHTWARDS TRIPLE ARROW

⇜ 21DC LEFTWARDS SQUIGGLE ARROW

⇝ 21DD RIGHTWARDS SQUIGGLE ARROW

⇞ 21DE UPWARDS ARROW WITH DOUBLE STROKE

⇟ 21DF DOWNWARDS ARROW WITH DOUBLE STROKE

⇠ 21E0 LEFTWARDS DASHED ARROW

⇡ 21E1 UPWARDS DASHED ARROW

⇢ 21E2 RIGHTWARDS DASHED ARROW

⇣ 21E3 DOWNWARDS DASHED ARROW

⇤ 21E4 LEFTWARDS ARROW TO BAR

⇥ 21E5 RIGHTWARDS ARROW TO BAR

⇦ 21E6 LEFTWARDS WHITE ARROW

⇧ 21E7 UPWARDS WHITE ARROW

⇨ 21E8 RIGHTWARDS WHITE ARROW

⇩ 21E9 DOWNWARDS WHITE ARROW

⇪ 21EA UPWARDS WHITE ARROW FROM BAR

⇫ 21EB UPWARDS WHITE ARROW ON PEDESTAL

⇬ 21EC UPWARDS WHITE ARROW ON PEDESTAL WITH HORIZONTAL BAR

⇭ 21ED UPWARDS WHITE ARROW ON PEDESTAL WITH VERTICAL BAR

⇮ 21EE UPWARDS WHITE DOUBLE ARROW

⇯ 21EF UPWARDS WHITE DOUBLE ARROW ON PEDESTAL

⇰ 21F0 RIGHTWARDS WHITE ARROW FROM WALL

⇱ 21F1 NORTH WEST ARROW TO CORNER

⇲ 21F2 SOUTH EAST ARROW TO CORNER

⇳ 21F3 UP DOWN WHITE ARROW

⇴ 21F4 RIGHT ARROW WITH SMALL CIRCLE

⇵ 21F5 DOWNWARDS ARROW LEFTWARDS OF UPWARDS ARROW

⇶ 21F6 THREE RIGHTWARDS ARROWS

⇷ 21F7 LEFTWARDS ARROW WITH VERTICAL STROKE

⇸ 21F8 RIGHTWARDS ARROW WITH VERTICAL STROKE

⇹ 21F9 LEFT RIGHT ARROW WITH VERTICAL STROKE

⇺ 21FA LEFTWARDS ARROW WITH DOUBLE VERTICAL STROKE

⇻ 21FB RIGHTWARDS ARROW WITH DOUBLE VERTICAL STROKE

⇼ 21FC LEFT RIGHT ARROW WITH DOUBLE VERTICAL STROKE

⇽ 21FD LEFTWARDS OPEN-HEADED ARROW

⇾ 21FE RIGHTWARDS OPEN-HEADED ARROW

⇿ 21FF LEFT RIGHT OPEN-HEADED ARROW

∀ 2200 FOR ALL

∁ 2201 COMPLEMENT

∂ 2202 PARTIAL DIFFERENTIAL

∃ 2203 THERE EXISTS

∄ 2204 THERE DOES NOT EXIST

∅ 2205 EMPTY SET

∆ 2206 INCREMENT

∇ 2207 NABLA

∈ 2208 ELEMENT OF

∉ 2209 NOT AN ELEMENT OF

∊ 220A SMALL ELEMENT OF

∋ 220B CONTAINS AS MEMBER

∌ 220C DOES NOT CONTAIN AS MEMBER

∍ 220D SMALL CONTAINS AS MEMBER

∎ 220E END OF PROOF

∏ 220F N-ARY PRODUCT

∐ 2210 N-ARY COPRODUCT

∑ 2211 N-ARY SUMMATION

− 2212 MINUS SIGN

∓ 2213 MINUS-OR-PLUS SIGN

∔ 2214 DOT PLUS

∕ 2215 DIVISION SLASH

∖ 2216 SET MINUS

∗ 2217 ASTERISK OPERATOR

∘ 2218 RING OPERATOR

∙ 2219 BULLET OPERATOR

√ 221A SQUARE ROOT

∛ 221B CUBE ROOT

∜ 221C FOURTH ROOT

∝ 221D PROPORTIONAL TO

∞ 221E INFINITY

∟ 221F RIGHT ANGLE

∠ 2220 ANGLE

∡ 2221 MEASURED ANGLE

∢ 2222 SPHERICAL ANGLE

∣ 2223 DIVIDES

∤ 2224 DOES NOT DIVIDE

∥ 2225 PARALLEL TO

∦ 2226 NOT PARALLEL TO

∧ 2227 LOGICAL AND

∨ 2228 LOGICAL OR

∩ 2229 INTERSECTION

∪ 222A UNION

∫ 222B INTEGRAL

∬ 222C DOUBLE INTEGRAL

∭ 222D TRIPLE INTEGRAL

∮ 222E CONTOUR INTEGRAL

∯ 222F SURFACE INTEGRAL

∰ 2230 VOLUME INTEGRAL

∱ 2231 CLOCKWISE INTEGRAL

∲ 2232 CLOCKWISE CONTOUR INTEGRAL

∳ 2233 ANTICLOCKWISE CONTOUR INTEGRAL

∴ 2234 THEREFORE

∵ 2235 BECAUSE

∶ 2236 RATIO

∷ 2237 PROPORTION

∸ 2238 DOT MINUS

∹ 2239 EXCESS

∺ 223A GEOMETRIC PROPORTION

∻ 223B HOMOTHETIC

∼ 223C TILDE OPERATOR

∽ 223D REVERSED TILDE

∾ 223E INVERTED LAZY S

∿ 223F SINE WAVE

≀ 2240 WREATH PRODUCT

≁ 2241 NOT TILDE

≂ 2242 MINUS TILDE

≃ 2243 ASYMPTOTICALLY EQUAL TO

≄ 2244 NOT ASYMPTOTICALLY EQUAL TO

≅ 2245 APPROXIMATELY EQUAL TO

≆ 2246 APPROXIMATELY BUT NOT ACTUALLY EQUAL TO

≇ 2247 NEITHER APPROXIMATELY NOR ACTUALLY EQUAL TO

≈ 2248 ALMOST EQUAL TO

≉ 2249 NOT ALMOST EQUAL TO

≊ 224A ALMOST EQUAL OR EQUAL TO

≋ 224B TRIPLE TILDE

≌ 224C ALL EQUAL TO

≍ 224D EQUIVALENT TO

≎ 224E GEOMETRICALLY EQUIVALENT TO

≏ 224F DIFFERENCE BETWEEN

≐ 2250 APPROACHES THE LIMIT

≑ 2251 GEOMETRICALLY EQUAL TO

≒ 2252 APPROXIMATELY EQUAL TO OR THE IMAGE OF

≓ 2253 IMAGE OF OR APPROXIMATELY EQUAL TO

≔ 2254 COLON EQUALS

≕ 2255 EQUALS COLON

≖ 2256 RING IN EQUAL TO

≗ 2257 RING EQUAL TO

≘ 2258 CORRESPONDS TO

≙ 2259 ESTIMATES

≚ 225A EQUIANGULAR TO

≛ 225B STAR EQUALS

≜ 225C DELTA EQUAL TO

≝ 225D EQUAL TO BY DEFINITION

≞ 225E MEASURED BY

≟ 225F QUESTIONED EQUAL TO

≠ 2260 NOT EQUAL TO

≡ 2261 IDENTICAL TO

≢ 2262 NOT IDENTICAL TO

≣ 2263 STRICTLY EQUIVALENT TO

≤ 2264 LESS-THAN OR EQUAL TO

≥ 2265 GREATER-THAN OR EQUAL TO

≦ 2266 LESS-THAN OVER EQUAL TO

≧ 2267 GREATER-THAN OVER EQUAL TO

≨ 2268 LESS-THAN BUT NOT EQUAL TO

≩ 2269 GREATER-THAN BUT NOT EQUAL TO

≪ 226A MUCH LESS-THAN

≫ 226B MUCH GREATER-THAN

≬ 226C BETWEEN

≭ 226D NOT EQUIVALENT TO

≮ 226E NOT LESS-THAN

≯ 226F NOT GREATER-THAN

≰ 2270 NEITHER LESS-THAN NOR EQUAL TO

≱ 2271 NEITHER GREATER-THAN NOR EQUAL TO

≲ 2272 LESS-THAN OR EQUIVALENT TO

≳ 2273 GREATER-THAN OR EQUIVALENT TO

≴ 2274 NEITHER LESS-THAN NOR EQUIVALENT TO

≵ 2275 NEITHER GREATER-THAN NOR EQUIVALENT TO

≶ 2276 LESS-THAN OR GREATER-THAN

≷ 2277 GREATER-THAN OR LESS-THAN

≸ 2278 NEITHER LESS-THAN NOR GREATER-THAN

≹ 2279 NEITHER GREATER-THAN NOR LESS-THAN

≺ 227A PRECEDES

≻ 227B SUCCEEDS

≼ 227C PRECEDES OR EQUAL TO

≽ 227D SUCCEEDS OR EQUAL TO

≾ 227E PRECEDES OR EQUIVALENT TO

≿ 227F SUCCEEDS OR EQUIVALENT TO

⊀ 2280 DOES NOT PRECEDE

⊁ 2281 DOES NOT SUCCEED

⊂ 2282 SUBSET OF

⊃ 2283 SUPERSET OF

⊄ 2284 NOT A SUBSET OF

⊅ 2285 NOT A SUPERSET OF

⊆ 2286 SUBSET OF OR EQUAL TO

⊇ 2287 SUPERSET OF OR EQUAL TO

⊈ 2288 NEITHER A SUBSET OF NOR EQUAL TO

⊉ 2289 NEITHER A SUPERSET OF NOR EQUAL TO

⊊ 228A SUBSET OF WITH NOT EQUAL TO

⊋ 228B SUPERSET OF WITH NOT EQUAL TO

⊌ 228C MULTISET

⊍ 228D MULTISET MULTIPLICATION

⊎ 228E MULTISET UNION

⊏ 228F SQUARE IMAGE OF

⊐ 2290 SQUARE ORIGINAL OF

⊑ 2291 SQUARE IMAGE OF OR EQUAL TO

⊒ 2292 SQUARE ORIGINAL OF OR EQUAL TO

⊓ 2293 SQUARE CAP

⊔ 2294 SQUARE CUP

⊕ 2295 CIRCLED PLUS

⊖ 2296 CIRCLED MINUS

⊗ 2297 CIRCLED TIMES

⊘ 2298 CIRCLED DIVISION SLASH

⊙ 2299 CIRCLED DOT OPERATOR

⊚ 229A CIRCLED RING OPERATOR

⊛ 229B CIRCLED ASTERISK OPERATOR

⊜ 229C CIRCLED EQUALS

⊝ 229D CIRCLED DASH

⊞ 229E SQUARED PLUS

⊟ 229F SQUARED MINUS

⊠ 22A0 SQUARED TIMES

⊡ 22A1 SQUARED DOT OPERATOR

⊢ 22A2 RIGHT TACK

⊣ 22A3 LEFT TACK

⊤ 22A4 DOWN TACK

⊥ 22A5 UP TACK

⊦ 22A6 ASSERTION

⊧ 22A7 MODELS

⊨ 22A8 TRUE

⊩ 22A9 FORCES

⊪ 22AA TRIPLE VERTICAL BAR RIGHT TURNSTILE

⊫ 22AB DOUBLE VERTICAL BAR DOUBLE RIGHT TURNSTILE

⊬ 22AC DOES NOT PROVE

⊭ 22AD NOT TRUE

⊮ 22AE DOES NOT FORCE

⊯ 22AF NEGATED DOUBLE VERTICAL BAR DOUBLE RIGHT TURNSTILE

⊰ 22B0 PRECEDES UNDER RELATION

⊱ 22B1 SUCCEEDS UNDER RELATION

⊲ 22B2 NORMAL SUBGROUP OF

⊳ 22B3 CONTAINS AS NORMAL SUBGROUP

⊴ 22B4 NORMAL SUBGROUP OF OR EQUAL TO

⊵ 22B5 CONTAINS AS NORMAL SUBGROUP OR EQUAL TO

⊶ 22B6 ORIGINAL OF

⊷ 22B7 IMAGE OF

⊸ 22B8 MULTIMAP

⊹ 22B9 HERMITIAN CONJUGATE MATRIX

⊺ 22BA INTERCALATE

⊻ 22BB XOR

⊼ 22BC NAND

⊽ 22BD NOR

⊾ 22BE RIGHT ANGLE WITH ARC

⊿ 22BF RIGHT TRIANGLE

⋀ 22C0 N-ARY LOGICAL AND

⋁ 22C1 N-ARY LOGICAL OR

⋂ 22C2 N-ARY INTERSECTION

⋃ 22C3 N-ARY UNION

⋄ 22C4 DIAMOND OPERATOR

⋅ 22C5 DOT OPERATOR

⋆ 22C6 STAR OPERATOR

⋇ 22C7 DIVISION TIMES

⋈ 22C8 BOWTIE

⋉ 22C9 LEFT NORMAL FACTOR SEMIDIRECT PRODUCT

⋊ 22CA RIGHT NORMAL FACTOR SEMIDIRECT PRODUCT

⋋ 22CB LEFT SEMIDIRECT PRODUCT

⋌ 22CC RIGHT SEMIDIRECT PRODUCT

⋍ 22CD REVERSED TILDE EQUALS

⋎ 22CE CURLY LOGICAL OR

⋏ 22CF CURLY LOGICAL AND

⋐ 22D0 DOUBLE SUBSET

⋑ 22D1 DOUBLE SUPERSET

⋒ 22D2 DOUBLE INTERSECTION

⋓ 22D3 DOUBLE UNION

⋔ 22D4 PITCHFORK

⋕ 22D5 EQUAL AND PARALLEL TO

⋖ 22D6 LESS-THAN WITH DOT

⋗ 22D7 GREATER-THAN WITH DOT

⋘ 22D8 VERY MUCH LESS-THAN

⋙ 22D9 VERY MUCH GREATER-THAN

⋚ 22DA LESS-THAN EQUAL TO OR GREATER-THAN

⋛ 22DB GREATER-THAN EQUAL TO OR LESS-THAN

⋜ 22DC EQUAL TO OR LESS-THAN

⋝ 22DD EQUAL TO OR GREATER-THAN

⋞ 22DE EQUAL TO OR PRECEDES

⋟ 22DF EQUAL TO OR SUCCEEDS

⋠ 22E0 DOES NOT PRECEDE OR EQUAL

⋡ 22E1 DOES NOT SUCCEED OR EQUAL

⋢ 22E2 NOT SQUARE IMAGE OF OR EQUAL TO

⋣ 22E3 NOT SQUARE ORIGINAL OF OR EQUAL TO

⋤ 22E4 SQUARE IMAGE OF OR NOT EQUAL TO

⋥ 22E5 SQUARE ORIGINAL OF OR NOT EQUAL TO

⋦ 22E6 LESS-THAN BUT NOT EQUIVALENT TO

⋧ 22E7 GREATER-THAN BUT NOT EQUIVALENT TO

⋨ 22E8 PRECEDES BUT NOT EQUIVALENT TO

⋩ 22E9 SUCCEEDS BUT NOT EQUIVALENT TO

⋪ 22EA NOT NORMAL SUBGROUP OF

⋫ 22EB DOES NOT CONTAIN AS NORMAL SUBGROUP

⋬ 22EC NOT NORMAL SUBGROUP OF OR EQUAL TO

⋭ 22ED DOES NOT CONTAIN AS NORMAL SUBGROUP OR EQUAL

⋮ 22EE VERTICAL ELLIPSIS

⋯ 22EF MIDLINE HORIZONTAL ELLIPSIS

⋰ 22F0 UP RIGHT DIAGONAL ELLIPSIS

⋱ 22F1 DOWN RIGHT DIAGONAL ELLIPSIS

⋲ 22F2 ELEMENT OF WITH LONG HORIZONTAL STROKE

⋳ 22F3 ELEMENT OF WITH VERTICAL BAR AT END OF HORIZONTAL STROKE

⋴ 22F4 SMALL ELEMENT OF WITH VERTICAL BAR AT END OF HORIZONTAL STROKE

⋵ 22F5 ELEMENT OF WITH DOT ABOVE

⋶ 22F6 ELEMENT OF WITH OVERBAR

⋷ 22F7 SMALL ELEMENT OF WITH OVERBAR

⋸ 22F8 ELEMENT OF WITH UNDERBAR

⋹ 22F9 ELEMENT OF WITH TWO HORIZONTAL STROKES

⋺ 22FA CONTAINS WITH LONG HORIZONTAL STROKE

⋻ 22FB CONTAINS WITH VERTICAL BAR AT END OF HORIZONTAL STROKE

⋼ 22FC SMALL CONTAINS WITH VERTICAL BAR AT END OF HORIZONTAL STROKE

⋽ 22FD CONTAINS WITH OVERBAR

⋾ 22FE SMALL CONTAINS WITH OVERBAR

⋿ 22FF Z NOTATION BAG MEMBERSHIP

─ 2500 BOX DRAWINGS LIGHT HORIZONTAL

━ 2501 BOX DRAWINGS HEAVY HORIZONTAL

│ 2502 BOX DRAWINGS LIGHT VERTICAL

┃ 2503 BOX DRAWINGS HEAVY VERTICAL

┄ 2504 BOX DRAWINGS LIGHT TRIPLE DASH HORIZONTAL

┅ 2505 BOX DRAWINGS HEAVY TRIPLE DASH HORIZONTAL

┆ 2506 BOX DRAWINGS LIGHT TRIPLE DASH VERTICAL

┇ 2507 BOX DRAWINGS HEAVY TRIPLE DASH VERTICAL

┈ 2508 BOX DRAWINGS LIGHT QUADRUPLE DASH HORIZONTAL

┉ 2509 BOX DRAWINGS HEAVY QUADRUPLE DASH HORIZONTAL

┊ 250A BOX DRAWINGS LIGHT QUADRUPLE DASH VERTICAL

┋ 250B BOX DRAWINGS HEAVY QUADRUPLE DASH VERTICAL

┌ 250C BOX DRAWINGS LIGHT DOWN AND RIGHT

┍ 250D BOX DRAWINGS DOWN LIGHT AND RIGHT HEAVY

┎ 250E BOX DRAWINGS DOWN HEAVY AND RIGHT LIGHT

┏ 250F BOX DRAWINGS HEAVY DOWN AND RIGHT

┐ 2510 BOX DRAWINGS LIGHT DOWN AND LEFT

┑ 2511 BOX DRAWINGS DOWN LIGHT AND LEFT HEAVY

┒ 2512 BOX DRAWINGS DOWN HEAVY AND LEFT LIGHT

┓ 2513 BOX DRAWINGS HEAVY DOWN AND LEFT

└ 2514 BOX DRAWINGS LIGHT UP AND RIGHT

┕ 2515 BOX DRAWINGS UP LIGHT AND RIGHT HEAVY

┖ 2516 BOX DRAWINGS UP HEAVY AND RIGHT LIGHT

┗ 2517 BOX DRAWINGS HEAVY UP AND RIGHT

┘ 2518 BOX DRAWINGS LIGHT UP AND LEFT

┙ 2519 BOX DRAWINGS UP LIGHT AND LEFT HEAVY

┚ 251A BOX DRAWINGS UP HEAVY AND LEFT LIGHT

┛ 251B BOX DRAWINGS HEAVY UP AND LEFT

├ 251C BOX DRAWINGS LIGHT VERTICAL AND RIGHT

┝ 251D BOX DRAWINGS VERTICAL LIGHT AND RIGHT HEAVY

┞ 251E BOX DRAWINGS UP HEAVY AND RIGHT DOWN LIGHT

┟ 251F BOX DRAWINGS DOWN HEAVY AND RIGHT UP LIGHT

┠ 2520 BOX DRAWINGS VERTICAL HEAVY AND RIGHT LIGHT

┡ 2521 BOX DRAWINGS DOWN LIGHT AND RIGHT UP HEAVY

┢ 2522 BOX DRAWINGS UP LIGHT AND RIGHT DOWN HEAVY

┣ 2523 BOX DRAWINGS HEAVY VERTICAL AND RIGHT

┤ 2524 BOX DRAWINGS LIGHT VERTICAL AND LEFT

┥ 2525 BOX DRAWINGS VERTICAL LIGHT AND LEFT HEAVY

┦ 2526 BOX DRAWINGS UP HEAVY AND LEFT DOWN LIGHT

┧ 2527 BOX DRAWINGS DOWN HEAVY AND LEFT UP LIGHT

┨ 2528 BOX DRAWINGS VERTICAL HEAVY AND LEFT LIGHT

┩ 2529 BOX DRAWINGS DOWN LIGHT AND LEFT UP HEAVY

┪ 252A BOX DRAWINGS UP LIGHT AND LEFT DOWN HEAVY

┫ 252B BOX DRAWINGS HEAVY VERTICAL AND LEFT

┬ 252C BOX DRAWINGS LIGHT DOWN AND HORIZONTAL

┭ 252D BOX DRAWINGS LEFT HEAVY AND RIGHT DOWN LIGHT

┮ 252E BOX DRAWINGS RIGHT HEAVY AND LEFT DOWN LIGHT

┯ 252F BOX DRAWINGS DOWN LIGHT AND HORIZONTAL HEAVY

┰ 2530 BOX DRAWINGS DOWN HEAVY AND HORIZONTAL LIGHT

┱ 2531 BOX DRAWINGS RIGHT LIGHT AND LEFT DOWN HEAVY

┲ 2532 BOX DRAWINGS LEFT LIGHT AND RIGHT DOWN HEAVY

┳ 2533 BOX DRAWINGS HEAVY DOWN AND HORIZONTAL

┴ 2534 BOX DRAWINGS LIGHT UP AND HORIZONTAL

┵ 2535 BOX DRAWINGS LEFT HEAVY AND RIGHT UP LIGHT

┶ 2536 BOX DRAWINGS RIGHT HEAVY AND LEFT UP LIGHT

┷ 2537 BOX DRAWINGS UP LIGHT AND HORIZONTAL HEAVY

┸ 2538 BOX DRAWINGS UP HEAVY AND HORIZONTAL LIGHT

┹ 2539 BOX DRAWINGS RIGHT LIGHT AND LEFT UP HEAVY

┺ 253A BOX DRAWINGS LEFT LIGHT AND RIGHT UP HEAVY

┻ 253B BOX DRAWINGS HEAVY UP AND HORIZONTAL

┼ 253C BOX DRAWINGS LIGHT VERTICAL AND HORIZONTAL

┽ 253D BOX DRAWINGS LEFT HEAVY AND RIGHT VERTICAL LIGHT

┾ 253E BOX DRAWINGS RIGHT HEAVY AND LEFT VERTICAL LIGHT

┿ 253F BOX DRAWINGS VERTICAL LIGHT AND HORIZONTAL HEAVY

╀ 2540 BOX DRAWINGS UP HEAVY AND DOWN HORIZONTAL LIGHT

╁ 2541 BOX DRAWINGS DOWN HEAVY AND UP HORIZONTAL LIGHT

╂ 2542 BOX DRAWINGS VERTICAL HEAVY AND HORIZONTAL LIGHT

╃ 2543 BOX DRAWINGS LEFT UP HEAVY AND RIGHT DOWN LIGHT

╄ 2544 BOX DRAWINGS RIGHT UP HEAVY AND LEFT DOWN LIGHT

╅ 2545 BOX DRAWINGS LEFT DOWN HEAVY AND RIGHT UP LIGHT

╆ 2546 BOX DRAWINGS RIGHT DOWN HEAVY AND LEFT UP LIGHT

╇ 2547 BOX DRAWINGS DOWN LIGHT AND UP HORIZONTAL HEAVY

╈ 2548 BOX DRAWINGS UP LIGHT AND DOWN HORIZONTAL HEAVY

╉ 2549 BOX DRAWINGS RIGHT LIGHT AND LEFT VERTICAL HEAVY

╊ 254A BOX DRAWINGS LEFT LIGHT AND RIGHT VERTICAL HEAVY

╋ 254B BOX DRAWINGS HEAVY VERTICAL AND HORIZONTAL

╌ 254C BOX DRAWINGS LIGHT DOUBLE DASH HORIZONTAL

╍ 254D BOX DRAWINGS HEAVY DOUBLE DASH HORIZONTAL

╎ 254E BOX DRAWINGS LIGHT DOUBLE DASH VERTICAL

╏ 254F BOX DRAWINGS HEAVY DOUBLE DASH VERTICAL

═ 2550 BOX DRAWINGS DOUBLE HORIZONTAL

║ 2551 BOX DRAWINGS DOUBLE VERTICAL

╒ 2552 BOX DRAWINGS DOWN SINGLE AND RIGHT DOUBLE

╓ 2553 BOX DRAWINGS DOWN DOUBLE AND RIGHT SINGLE

╔ 2554 BOX DRAWINGS DOUBLE DOWN AND RIGHT

╕ 2555 BOX DRAWINGS DOWN SINGLE AND LEFT DOUBLE

╖ 2556 BOX DRAWINGS DOWN DOUBLE AND LEFT SINGLE

╗ 2557 BOX DRAWINGS DOUBLE DOWN AND LEFT

╘ 2558 BOX DRAWINGS UP SINGLE AND RIGHT DOUBLE

╙ 2559 BOX DRAWINGS UP DOUBLE AND RIGHT SINGLE

╚ 255A BOX DRAWINGS DOUBLE UP AND RIGHT

╛ 255B BOX DRAWINGS UP SINGLE AND LEFT DOUBLE

╜ 255C BOX DRAWINGS UP DOUBLE AND LEFT SINGLE

╝ 255D BOX DRAWINGS DOUBLE UP AND LEFT

╞ 255E BOX DRAWINGS VERTICAL SINGLE AND RIGHT DOUBLE

╟ 255F BOX DRAWINGS VERTICAL DOUBLE AND RIGHT SINGLE

╠ 2560 BOX DRAWINGS DOUBLE VERTICAL AND RIGHT

╡ 2561 BOX DRAWINGS VERTICAL SINGLE AND LEFT DOUBLE

╢ 2562 BOX DRAWINGS VERTICAL DOUBLE AND LEFT SINGLE

╣ 2563 BOX DRAWINGS DOUBLE VERTICAL AND LEFT

╤ 2564 BOX DRAWINGS DOWN SINGLE AND HORIZONTAL DOUBLE

╥ 2565 BOX DRAWINGS DOWN DOUBLE AND HORIZONTAL SINGLE

╦ 2566 BOX DRAWINGS DOUBLE DOWN AND HORIZONTAL

╧ 2567 BOX DRAWINGS UP SINGLE AND HORIZONTAL DOUBLE

╨ 2568 BOX DRAWINGS UP DOUBLE AND HORIZONTAL SINGLE

╩ 2569 BOX DRAWINGS DOUBLE UP AND HORIZONTAL

╪ 256A BOX DRAWINGS VERTICAL SINGLE AND HORIZONTAL DOUBLE

╫ 256B BOX DRAWINGS VERTICAL DOUBLE AND HORIZONTAL SINGLE

╬ 256C BOX DRAWINGS DOUBLE VERTICAL AND HORIZONTAL

╭ 256D BOX DRAWINGS LIGHT ARC DOWN AND RIGHT

╮ 256E BOX DRAWINGS LIGHT ARC DOWN AND LEFT

╯ 256F BOX DRAWINGS LIGHT ARC UP AND LEFT

╰ 2570 BOX DRAWINGS LIGHT ARC UP AND RIGHT

╱ 2571 BOX DRAWINGS LIGHT DIAGONAL UPPER RIGHT TO LOWER LEFT

╲ 2572 BOX DRAWINGS LIGHT DIAGONAL UPPER LEFT TO LOWER RIGHT

╳ 2573 BOX DRAWINGS LIGHT DIAGONAL CROSS

╴ 2574 BOX DRAWINGS LIGHT LEFT

╵ 2575 BOX DRAWINGS LIGHT UP

╶ 2576 BOX DRAWINGS LIGHT RIGHT

╷ 2577 BOX DRAWINGS LIGHT DOWN

╸ 2578 BOX DRAWINGS HEAVY LEFT

╹ 2579 BOX DRAWINGS HEAVY UP

╺ 257A BOX DRAWINGS HEAVY RIGHT

╻ 257B BOX DRAWINGS HEAVY DOWN

╼ 257C BOX DRAWINGS LIGHT LEFT AND HEAVY RIGHT

╽ 257D BOX DRAWINGS LIGHT UP AND HEAVY DOWN

╾ 257E BOX DRAWINGS HEAVY LEFT AND LIGHT RIGHT

╿ 257F BOX DRAWINGS HEAVY UP AND LIGHT DOWN

▀ 2580 UPPER HALF BLOCK

▁ 2581 LOWER ONE EIGHTH BLOCK

▂ 2582 LOWER ONE QUARTER BLOCK

▃ 2583 LOWER THREE EIGHTHS BLOCK

▄ 2584 LOWER HALF BLOCK

▅ 2585 LOWER FIVE EIGHTHS BLOCK

▆ 2586 LOWER THREE QUARTERS BLOCK

▇ 2587 LOWER SEVEN EIGHTHS BLOCK

█ 2588 FULL BLOCK

▉ 2589 LEFT SEVEN EIGHTHS BLOCK

▊ 258A LEFT THREE QUARTERS BLOCK

▋ 258B LEFT FIVE EIGHTHS BLOCK

▌ 258C LEFT HALF BLOCK

▍ 258D LEFT THREE EIGHTHS BLOCK

▎ 258E LEFT ONE QUARTER BLOCK

▏ 258F LEFT ONE EIGHTH BLOCK

▐ 2590 RIGHT HALF BLOCK

░ 2591 LIGHT SHADE

▒ 2592 MEDIUM SHADE

▓ 2593 DARK SHADE

▔ 2594 UPPER ONE EIGHTH BLOCK

▕ 2595 RIGHT ONE EIGHTH BLOCK

▖ 2596 QUADRANT LOWER LEFT

▗ 2597 QUADRANT LOWER RIGHT

▘ 2598 QUADRANT UPPER LEFT

▙ 2599 QUADRANT UPPER LEFT AND LOWER LEFT AND LOWER RIGHT

▚ 259A QUADRANT UPPER LEFT AND LOWER RIGHT

▛ 259B QUADRANT UPPER LEFT AND UPPER RIGHT AND LOWER LEFT

▜ 259C QUADRANT UPPER LEFT AND UPPER RIGHT AND LOWER RIGHT

▝ 259D QUADRANT UPPER RIGHT

▞ 259E QUADRANT UPPER RIGHT AND LOWER LEFT

▟ 259F QUADRANT UPPER RIGHT AND LOWER LEFT AND LOWER RIGHT

■ 25A0 BLACK SQUARE

□ 25A1 WHITE SQUARE

▢ 25A2 WHITE SQUARE WITH ROUNDED CORNERS

▣ 25A3 WHITE SQUARE CONTAINING BLACK SMALL SQUARE

▤ 25A4 SQUARE WITH HORIZONTAL FILL

▥ 25A5 SQUARE WITH VERTICAL FILL

▦ 25A6 SQUARE WITH ORTHOGONAL CROSSHATCH FILL

▧ 25A7 SQUARE WITH UPPER LEFT TO LOWER RIGHT FILL

▨ 25A8 SQUARE WITH UPPER RIGHT TO LOWER LEFT FILL

▩ 25A9 SQUARE WITH DIAGONAL CROSSHATCH FILL

▪ 25AA BLACK SMALL SQUARE

▫ 25AB WHITE SMALL SQUARE

▬ 25AC BLACK RECTANGLE

▭ 25AD WHITE RECTANGLE

▮ 25AE BLACK VERTICAL RECTANGLE

▯ 25AF WHITE VERTICAL RECTANGLE

▰ 25B0 BLACK PARALLELOGRAM

▱ 25B1 WHITE PARALLELOGRAM

▲ 25B2 BLACK UP-POINTING TRIANGLE

△ 25B3 WHITE UP-POINTING TRIANGLE

▴ 25B4 BLACK UP-POINTING SMALL TRIANGLE

▵ 25B5 WHITE UP-POINTING SMALL TRIANGLE

▶ 25B6 BLACK RIGHT-POINTING TRIANGLE

▷ 25B7 WHITE RIGHT-POINTING TRIANGLE

▸ 25B8 BLACK RIGHT-POINTING SMALL TRIANGLE

▹ 25B9 WHITE RIGHT-POINTING SMALL TRIANGLE

► 25BA BLACK RIGHT-POINTING POINTER

▻ 25BB WHITE RIGHT-POINTING POINTER

▼ 25BC BLACK DOWN-POINTING TRIANGLE

▽ 25BD WHITE DOWN-POINTING TRIANGLE

▾ 25BE BLACK DOWN-POINTING SMALL TRIANGLE

▿ 25BF WHITE DOWN-POINTING SMALL TRIANGLE

◀ 25C0 BLACK LEFT-POINTING TRIANGLE

◁ 25C1 WHITE LEFT-POINTING TRIANGLE

◂ 25C2 BLACK LEFT-POINTING SMALL TRIANGLE

◃ 25C3 WHITE LEFT-POINTING SMALL TRIANGLE

◄ 25C4 BLACK LEFT-POINTING POINTER

◅ 25C5 WHITE LEFT-POINTING POINTER

◆ 25C6 BLACK DIAMOND

◇ 25C7 WHITE DIAMOND

◈ 25C8 WHITE DIAMOND CONTAINING BLACK SMALL DIAMOND

◉ 25C9 FISHEYE

◊ 25CA LOZENGE

○ 25CB WHITE CIRCLE

◌ 25CC DOTTED CIRCLE

◍ 25CD CIRCLE WITH VERTICAL FILL

◎ 25CE BULLSEYE

● 25CF BLACK CIRCLE

◐ 25D0 CIRCLE WITH LEFT HALF BLACK

◑ 25D1 CIRCLE WITH RIGHT HALF BLACK

◒ 25D2 CIRCLE WITH LOWER HALF BLACK

◓ 25D3 CIRCLE WITH UPPER HALF BLACK

◔ 25D4 CIRCLE WITH UPPER RIGHT QUADRANT BLACK

◕ 25D5 CIRCLE WITH ALL BUT UPPER LEFT QUADRANT BLACK

◖ 25D6 LEFT HALF BLACK CIRCLE

◗ 25D7 RIGHT HALF BLACK CIRCLE

◘ 25D8 INVERSE BULLET

◙ 25D9 INVERSE WHITE CIRCLE

◚ 25DA UPPER HALF INVERSE WHITE CIRCLE

◛ 25DB LOWER HALF INVERSE WHITE CIRCLE

◜ 25DC UPPER LEFT QUADRANT CIRCULAR ARC

◝ 25DD UPPER RIGHT QUADRANT CIRCULAR ARC

◞ 25DE LOWER RIGHT QUADRANT CIRCULAR ARC

◟ 25DF LOWER LEFT QUADRANT CIRCULAR ARC

◠ 25E0 UPPER HALF CIRCLE

◡ 25E1 LOWER HALF CIRCLE

◢ 25E2 BLACK LOWER RIGHT TRIANGLE

◣ 25E3 BLACK LOWER LEFT TRIANGLE

◤ 25E4 BLACK UPPER LEFT TRIANGLE

◥ 25E5 BLACK UPPER RIGHT TRIANGLE

◦ 25E6 WHITE BULLET

◧ 25E7 SQUARE WITH LEFT HALF BLACK

◨ 25E8 SQUARE WITH RIGHT HALF BLACK

◩ 25E9 SQUARE WITH UPPER LEFT DIAGONAL HALF BLACK

◪ 25EA SQUARE WITH LOWER RIGHT DIAGONAL HALF BLACK

◫ 25EB WHITE SQUARE WITH VERTICAL BISECTING LINE

◬ 25EC WHITE UP-POINTING TRIANGLE WITH DOT

◭ 25ED UP-POINTING TRIANGLE WITH LEFT HALF BLACK

◮ 25EE UP-POINTING TRIANGLE WITH RIGHT HALF BLACK

◯ 25EF LARGE CIRCLE

◰ 25F0 WHITE SQUARE WITH UPPER LEFT QUADRANT

◱ 25F1 WHITE SQUARE WITH LOWER LEFT QUADRANT

◲ 25F2 WHITE SQUARE WITH LOWER RIGHT QUADRANT

◳ 25F3 WHITE SQUARE WITH UPPER RIGHT QUADRANT

◴ 25F4 WHITE CIRCLE WITH UPPER LEFT QUADRANT

◵ 25F5 WHITE CIRCLE WITH LOWER LEFT QUADRANT

◶ 25F6 WHITE CIRCLE WITH LOWER RIGHT QUADRANT

◷ 25F7 WHITE CIRCLE WITH UPPER RIGHT QUADRANT

◸ 25F8 UPPER LEFT TRIANGLE

◹ 25F9 UPPER RIGHT TRIANGLE

◺ 25FA LOWER LEFT TRIANGLE

◻ 25FB WHITE MEDIUM SQUARE

◼ 25FC BLACK MEDIUM SQUARE

◽ 25FD WHITE MEDIUM SMALL SQUARE

◾ 25FE BLACK MEDIUM SMALL SQUARE

◿ 25FF LOWER RIGHT TRIANGLE

☀ 2600 BLACK SUN WITH RAYS

☁ 2601 CLOUD

☂ 2602 UMBRELLA

☃ 2603 SNOWMAN

☄ 2604 COMET

★ 2605 BLACK STAR

☆ 2606 WHITE STAR

☇ 2607 LIGHTNING

☈ 2608 THUNDERSTORM

☉ 2609 SUN

☊ 260A ASCENDING NODE

☋ 260B DESCENDING NODE

☌ 260C CONJUNCTION

☍ 260D OPPOSITION

☎ 260E BLACK TELEPHONE

☏ 260F WHITE TELEPHONE

☐ 2610 BALLOT BOX

☑ 2611 BALLOT BOX WITH CHECK

☒ 2612 BALLOT BOX WITH X

☓ 2613 SALTIRE

☔ 2614 UMBRELLA WITH RAIN DROPS

☕ 2615 HOT BEVERAGE

☖ 2616 WHITE SHOGI PIECE

☗ 2617 BLACK SHOGI PIECE

☘ 2618 SHAMROCK

☙ 2619 REVERSED ROTATED FLORAL HEART BULLET

☚ 261A BLACK LEFT POINTING INDEX

☛ 261B BLACK RIGHT POINTING INDEX

☜ 261C WHITE LEFT POINTING INDEX

☝ 261D WHITE UP POINTING INDEX

☞ 261E WHITE RIGHT POINTING INDEX

☟ 261F WHITE DOWN POINTING INDEX

☠ 2620 SKULL AND CROSSBONES

☡ 2621 CAUTION SIGN

☢ 2622 RADIOACTIVE SIGN

☣ 2623 BIOHAZARD SIGN

☤ 2624 CADUCEUS

☥ 2625 ANKH

☦ 2626 ORTHODOX CROSS

☧ 2627 CHI RHO

☨ 2628 CROSS OF LORRAINE

☩ 2629 CROSS OF JERUSALEM

☪ 262A STAR AND CRESCENT

☫ 262B FARSI SYMBOL

☬ 262C KHANDA

☭ 262D HAMMER AND SICKLE

☮ 262E PEACE SYMBOL

☯ 262F YIN YANG

☰ 2630 TRIGRAM FOR HEAVEN

☱ 2631 TRIGRAM FOR LAKE

☲ 2632 TRIGRAM FOR FIRE

☳ 2633 TRIGRAM FOR THUNDER

☴ 2634 TRIGRAM FOR WIND

☵ 2635 TRIGRAM FOR WATER

☶ 2636 TRIGRAM FOR MOUNTAIN

☷ 2637 TRIGRAM FOR EARTH

☸ 2638 WHEEL OF DHARMA

☹ 2639 WHITE FROWNING FACE

☺ 263A WHITE SMILING FACE

☻ 263B BLACK SMILING FACE

☼ 263C WHITE SUN WITH RAYS

☽ 263D FIRST QUARTER MOON

☾ 263E LAST QUARTER MOON

☿ 263F MERCURY

♀ 2640 FEMALE SIGN

♁ 2641 EARTH

♂ 2642 MALE SIGN

♃ 2643 JUPITER

♄ 2644 SATURN

♅ 2645 URANUS

♆ 2646 NEPTUNE

♇ 2647 PLUTO

♈ 2648 ARIES

♉ 2649 TAURUS

♊ 264A GEMINI

♋ 264B CANCER

♌ 264C LEO

♍ 264D VIRGO

♎ 264E LIBRA

♏ 264F SCORPIUS

♐ 2650 SAGITTARIUS

♑ 2651 CAPRICORN

♒ 2652 AQUARIUS

♓ 2653 PISCES

♔ 2654 WHITE CHESS KING

♕ 2655 WHITE CHESS QUEEN

♖ 2656 WHITE CHESS ROOK

♗ 2657 WHITE CHESS BISHOP

♘ 2658 WHITE CHESS KNIGHT

♙ 2659 WHITE CHESS PAWN

♚ 265A BLACK CHESS KING

♛ 265B BLACK CHESS QUEEN

♜ 265C BLACK CHESS ROOK

♝ 265D BLACK CHESS BISHOP

♞ 265E BLACK CHESS KNIGHT

♟ 265F BLACK CHESS PAWN

♠ 2660 BLACK SPADE SUIT

♡ 2661 WHITE HEART SUIT

♢ 2662 WHITE DIAMOND SUIT

♣ 2663 BLACK CLUB SUIT

♤ 2664 WHITE SPADE SUIT

♥ 2665 BLACK HEART SUIT

♦ 2666 BLACK DIAMOND SUIT

♧ 2667 WHITE CLUB SUIT

♨ 2668 HOT SPRINGS

♩ 2669 QUARTER NOTE

♪ 266A EIGHTH NOTE

♫ 266B BEAMED EIGHTH NOTES

♬ 266C BEAMED SIXTEENTH NOTES

♭ 266D MUSIC FLAT SIGN

♮ 266E MUSIC NATURAL SIGN

♯ 266F MUSIC SHARP SIGN

♰ 2670 WEST SYRIAC CROSS

♱ 2671 EAST SYRIAC CROSS

♲ 2672 UNIVERSAL RECYCLING SYMBOL

♳ 2673 RECYCLING SYMBOL FOR TYPE-1 PLASTICS

♴ 2674 RECYCLING SYMBOL FOR TYPE-2 PLASTICS

♵ 2675 RECYCLING SYMBOL FOR TYPE-3 PLASTICS

♶ 2676 RECYCLING SYMBOL FOR TYPE-4 PLASTICS

♷ 2677 RECYCLING SYMBOL FOR TYPE-5 PLASTICS

♸ 2678 RECYCLING SYMBOL FOR TYPE-6 PLASTICS

♹ 2679 RECYCLING SYMBOL FOR TYPE-7 PLASTICS

♺ 267A RECYCLING SYMBOL FOR GENERIC MATERIALS

♻ 267B BLACK UNIVERSAL RECYCLING SYMBOL

♼ 267C RECYCLED PAPER SYMBOL

♽ 267D PARTIALLY-RECYCLED PAPER SYMBOL

♾ 267E PERMANENT PAPER SIGN

♿ 267F WHEELCHAIR SYMBOL

⚀ 2680 DIE FACE-1

⚁ 2681 DIE FACE-2

⚂ 2682 DIE FACE-3

⚃ 2683 DIE FACE-4

⚄ 2684 DIE FACE-5

⚅ 2685 DIE FACE-6

⚆ 2686 WHITE CIRCLE WITH DOT RIGHT

⚇ 2687 WHITE CIRCLE WITH TWO DOTS

⚈ 2688 BLACK CIRCLE WITH WHITE DOT RIGHT

⚉ 2689 BLACK CIRCLE WITH TWO WHITE DOTS

⚊ 268A MONOGRAM FOR YANG

⚋ 268B MONOGRAM FOR YIN

⚌ 268C DIGRAM FOR GREATER YANG

⚍ 268D DIGRAM FOR LESSER YIN

⚎ 268E DIGRAM FOR LESSER YANG

⚏ 268F DIGRAM FOR GREATER YIN

⚐ 2690 WHITE FLAG

⚑ 2691 BLACK FLAG

⚒ 2692 HAMMER AND PICK

⚓ 2693 ANCHOR

⚔ 2694 CROSSED SWORDS

⚕ 2695 STAFF OF AESCULAPIUS

⚖ 2696 SCALES

⚗ 2697 ALEMBIC

⚘ 2698 FLOWER

⚙ 2699 GEAR

⚚ 269A STAFF OF HERMES

⚛ 269B ATOM SYMBOL

⚜ 269C FLEUR-DE-LIS

⚝ 269D OUTLINED WHITE STAR

⚞ 269E THREE LINES CONVERGING RIGHT

⚟ 269F THREE LINES CONVERGING LEFT

⚠ 26A0 WARNING SIGN

⚡ 26A1 HIGH VOLTAGE SIGN

⚢ 26A2 DOUBLED FEMALE SIGN

⚣ 26A3 DOUBLED MALE SIGN

⚤ 26A4 INTERLOCKED FEMALE AND MALE SIGN

⚥ 26A5 MALE AND FEMALE SIGN

⚦ 26A6 MALE WITH STROKE SIGN

⚧ 26A7 MALE WITH STROKE AND MALE AND FEMALE SIGN

⚨ 26A8 VERTICAL MALE WITH STROKE SIGN

⚩ 26A9 HORIZONTAL MALE WITH STROKE SIGN

⚪ 26AA MEDIUM WHITE CIRCLE

⚫ 26AB MEDIUM BLACK CIRCLE

⚬ 26AC MEDIUM SMALL WHITE CIRCLE

⚭ 26AD MARRIAGE SYMBOL

⚮ 26AE DIVORCE SYMBOL

⚯ 26AF UNMARRIED PARTNERSHIP SYMBOL

⚰ 26B0 COFFIN

⚱ 26B1 FUNERAL URN

⚲ 26B2 NEUTER

⚳ 26B3 CERES

⚴ 26B4 PALLAS

⚵ 26B5 JUNO

⚶ 26B6 VESTA

⚷ 26B7 CHIRON

⚸ 26B8 BLACK MOON LILITH

⚹ 26B9 SEXTILE

⚺ 26BA SEMISEXTILE

⚻ 26BB QUINCUNX

⚼ 26BC SESQUIQUADRATE

⚽ 26BD SOCCER BALL

⚾ 26BE BASEBALL

⚿ 26BF SQUARED KEY

⛀ 26C0 WHITE DRAUGHTS MAN

⛁ 26C1 WHITE DRAUGHTS KING

⛂ 26C2 BLACK DRAUGHTS MAN

⛃ 26C3 BLACK DRAUGHTS KING

⛄ 26C4 SNOWMAN WITHOUT SNOW

⛅ 26C5 SUN BEHIND CLOUD

⛆ 26C6 RAIN

⛇ 26C7 BLACK SNOWMAN

⛈ 26C8 THUNDER CLOUD AND RAIN

⛉ 26C9 TURNED WHITE SHOGI PIECE

⛊ 26CA TURNED BLACK SHOGI PIECE

⛋ 26CB WHITE DIAMOND IN SQUARE

⛌ 26CC CROSSING LANES

⛍ 26CD DISABLED CAR

⛎ 26CE OPHIUCHUS

⛏ 26CF PICK

⛐ 26D0 CAR SLIDING

⛑ 26D1 HELMET WITH WHITE CROSS

⛒ 26D2 CIRCLED CROSSING LANES

⛓ 26D3 CHAINS

⛔ 26D4 NO ENTRY

⛕ 26D5 ALTERNATE ONE-WAY LEFT WAY TRAFFIC

⛖ 26D6 BLACK TWO-WAY LEFT WAY TRAFFIC

⛗ 26D7 WHITE TWO-WAY LEFT WAY TRAFFIC

⛘ 26D8 BLACK LEFT LANE MERGE

⛙ 26D9 WHITE LEFT LANE MERGE

⛚ 26DA DRIVE SLOW SIGN

⛛ 26DB HEAVY WHITE DOWN-POINTING TRIANGLE

⛜ 26DC LEFT CLOSED ENTRY

⛝ 26DD SQUARED SALTIRE

⛞ 26DE FALLING DIAGONAL IN WHITE CIRCLE IN BLACK SQUARE

⛟ 26DF BLACK TRUCK

⛠ 26E0 RESTRICTED LEFT ENTRY-1

⛡ 26E1 RESTRICTED LEFT ENTRY-2

⛢ 26E2 ASTRONOMICAL SYMBOL FOR URANUS

⛣ 26E3 HEAVY CIRCLE WITH STROKE AND TWO DOTS ABOVE

⛤ 26E4 PENTAGRAM

⛥ 26E5 RIGHT-HANDED INTERLACED PENTAGRAM

⛦ 26E6 LEFT-HANDED INTERLACED PENTAGRAM

⛧ 26E7 INVERTED PENTAGRAM

⛨ 26E8 BLACK CROSS ON SHIELD

⛩ 26E9 SHINTO SHRINE

⛪ 26EA CHURCH

⛫ 26EB CASTLE

⛬ 26EC HISTORIC SITE

⛭ 26ED GEAR WITHOUT HUB

⛮ 26EE GEAR WITH HANDLES

⛯ 26EF MAP SYMBOL FOR LIGHTHOUSE

⛰ 26F0 MOUNTAIN

⛱ 26F1 UMBRELLA ON GROUND

⛲ 26F2 FOUNTAIN

⛳ 26F3 FLAG IN HOLE

⛴ 26F4 FERRY

⛵ 26F5 SAILBOAT

⛶ 26F6 SQUARE FOUR CORNERS

⛷ 26F7 SKIER

⛸ 26F8 ICE SKATE

⛹ 26F9 PERSON WITH BALL

⛺ 26FA TENT

⛻ 26FB JAPANESE BANK SYMBOL

⛼ 26FC HEADSTONE GRAVEYARD SYMBOL

⛽ 26FD FUEL PUMP

⛾ 26FE CUP ON BLACK SQUARE

⛿ 26FF WHITE FLAG WITH HORIZONTAL MIDDLE BLACK STRIPE

✁ 2701 UPPER BLADE SCISSORS

✂ 2702 BLACK SCISSORS

✃ 2703 LOWER BLADE SCISSORS

✄ 2704 WHITE SCISSORS

✅ 2705 WHITE HEAVY CHECK MARK

✆ 2706 TELEPHONE LOCATION SIGN

✇ 2707 TAPE DRIVE

✈ 2708 AIRPLANE

✉ 2709 ENVELOPE

✊ 270A RAISED FIST

✋ 270B RAISED HAND

✌ 270C VICTORY HAND

✍ 270D WRITING HAND

✎ 270E LOWER RIGHT PENCIL

✏ 270F PENCIL

✐ 2710 UPPER RIGHT PENCIL

✑ 2711 WHITE NIB

✒ 2712 BLACK NIB

✓ 2713 CHECK MARK

✔ 2714 HEAVY CHECK MARK

✕ 2715 MULTIPLICATION X

✖ 2716 HEAVY MULTIPLICATION X

✗ 2717 BALLOT X

✘ 2718 HEAVY BALLOT X

✙ 2719 OUTLINED GREEK CROSS

✚ 271A HEAVY GREEK CROSS

✛ 271B OPEN CENTRE CROSS

✜ 271C HEAVY OPEN CENTRE CROSS

✝ 271D LATIN CROSS

✞ 271E SHADOWED WHITE LATIN CROSS

✟ 271F OUTLINED LATIN CROSS

✠ 2720 MALTESE CROSS

✡ 2721 STAR OF DAVID

✢ 2722 FOUR TEARDROP-SPOKED ASTERISK

✣ 2723 FOUR BALLOON-SPOKED ASTERISK

✤ 2724 HEAVY FOUR BALLOON-SPOKED ASTERISK

✥ 2725 FOUR CLUB-SPOKED ASTERISK

✦ 2726 BLACK FOUR POINTED STAR

✧ 2727 WHITE FOUR POINTED STAR

✨ 2728 SPARKLES

✩ 2729 STRESS OUTLINED WHITE STAR

✪ 272A CIRCLED WHITE STAR

✫ 272B OPEN CENTRE BLACK STAR

✬ 272C BLACK CENTRE WHITE STAR

✭ 272D OUTLINED BLACK STAR

✮ 272E HEAVY OUTLINED BLACK STAR

✯ 272F PINWHEEL STAR

✰ 2730 SHADOWED WHITE STAR

✱ 2731 HEAVY ASTERISK

✲ 2732 OPEN CENTRE ASTERISK

✳ 2733 EIGHT SPOKED ASTERISK

✴ 2734 EIGHT POINTED BLACK STAR

✵ 2735 EIGHT POINTED PINWHEEL STAR

✶ 2736 SIX POINTED BLACK STAR

✷ 2737 EIGHT POINTED RECTILINEAR BLACK STAR

✸ 2738 HEAVY EIGHT POINTED RECTILINEAR BLACK STAR

✹ 2739 TWELVE POINTED BLACK STAR

✺ 273A SIXTEEN POINTED ASTERISK

✻ 273B TEARDROP-SPOKED ASTERISK

✼ 273C OPEN CENTRE TEARDROP-SPOKED ASTERISK

✽ 273D HEAVY TEARDROP-SPOKED ASTERISK

✾ 273E SIX PETALLED BLACK AND WHITE FLORETTE

✿ 273F BLACK FLORETTE

❀ 2740 WHITE FLORETTE

❁ 2741 EIGHT PETALLED OUTLINED BLACK FLORETTE

❂ 2742 CIRCLED OPEN CENTRE EIGHT POINTED STAR

❃ 2743 HEAVY TEARDROP-SPOKED PINWHEEL ASTERISK

❄ 2744 SNOWFLAKE

❅ 2745 TIGHT TRIFOLIATE SNOWFLAKE

❆ 2746 HEAVY CHEVRON SNOWFLAKE

❇ 2747 SPARKLE

❈ 2748 HEAVY SPARKLE

❉ 2749 BALLOON-SPOKED ASTERISK

❊ 274A EIGHT TEARDROP-SPOKED PROPELLER ASTERISK

❋ 274B HEAVY EIGHT TEARDROP-SPOKED PROPELLER ASTERISK

❌ 274C CROSS MARK

❍ 274D SHADOWED WHITE CIRCLE

❎ 274E NEGATIVE SQUARED CROSS MARK

❏ 274F LOWER RIGHT DROP-SHADOWED WHITE SQUARE

❐ 2750 UPPER RIGHT DROP-SHADOWED WHITE SQUARE

❑ 2751 LOWER RIGHT SHADOWED WHITE SQUARE

❒ 2752 UPPER RIGHT SHADOWED WHITE SQUARE

❓ 2753 BLACK QUESTION MARK ORNAMENT

❔ 2754 WHITE QUESTION MARK ORNAMENT

❕ 2755 WHITE EXCLAMATION MARK ORNAMENT

❖ 2756 BLACK DIAMOND MINUS WHITE X

❗ 2757 HEAVY EXCLAMATION MARK SYMBOL

❘ 2758 LIGHT VERTICAL BAR

❙ 2759 MEDIUM VERTICAL BAR

❚ 275A HEAVY VERTICAL BAR

❛ 275B HEAVY SINGLE TURNED COMMA QUOTATION MARK ORNAMENT

❜ 275C HEAVY SINGLE COMMA QUOTATION MARK ORNAMENT

❝ 275D HEAVY DOUBLE TURNED COMMA QUOTATION MARK ORNAMENT

❞ 275E HEAVY DOUBLE COMMA QUOTATION MARK ORNAMENT

❟ 275F HEAVY LOW SINGLE COMMA QUOTATION MARK ORNAMENT

❠ 2760 HEAVY LOW DOUBLE COMMA QUOTATION MARK ORNAMENT

❡ 2761 CURVED STEM PARAGRAPH SIGN ORNAMENT

❢ 2762 HEAVY EXCLAMATION MARK ORNAMENT

❣ 2763 HEAVY HEART EXCLAMATION MARK ORNAMENT

❤ 2764 HEAVY BLACK HEART

❥ 2765 ROTATED HEAVY BLACK HEART BULLET

❦ 2766 FLORAL HEART

❧ 2767 ROTATED FLORAL HEART BULLET

❨ 2768 MEDIUM LEFT PARENTHESIS ORNAMENT

❩ 2769 MEDIUM RIGHT PARENTHESIS ORNAMENT

❪ 276A MEDIUM FLATTENED LEFT PARENTHESIS ORNAMENT

❫ 276B MEDIUM FLATTENED RIGHT PARENTHESIS ORNAMENT

❬ 276C MEDIUM LEFT-POINTING ANGLE BRACKET ORNAMENT

❭ 276D MEDIUM RIGHT-POINTING ANGLE BRACKET ORNAMENT

❮ 276E HEAVY LEFT-POINTING ANGLE QUOTATION MARK ORNAMENT

❯ 276F HEAVY RIGHT-POINTING ANGLE QUOTATION MARK ORNAMENT

❰ 2770 HEAVY LEFT-POINTING ANGLE BRACKET ORNAMENT

❱ 2771 HEAVY RIGHT-POINTING ANGLE BRACKET ORNAMENT

❲ 2772 LIGHT LEFT TORTOISE SHELL BRACKET ORNAMENT

❳ 2773 LIGHT RIGHT TORTOISE SHELL BRACKET ORNAMENT

❴ 2774 MEDIUM LEFT CURLY BRACKET ORNAMENT

❵ 2775 MEDIUM RIGHT CURLY BRACKET ORNAMENT

❶ 2776 DINGBAT NEGATIVE CIRCLED DIGIT ONE

❷ 2777 DINGBAT NEGATIVE CIRCLED DIGIT TWO

❸ 2778 DINGBAT NEGATIVE CIRCLED DIGIT THREE

❹ 2779 DINGBAT NEGATIVE CIRCLED DIGIT FOUR

❺ 277A DINGBAT NEGATIVE CIRCLED DIGIT FIVE

❻ 277B DINGBAT NEGATIVE CIRCLED DIGIT SIX

❼ 277C DINGBAT NEGATIVE CIRCLED DIGIT SEVEN

❽ 277D DINGBAT NEGATIVE CIRCLED DIGIT EIGHT

❾ 277E DINGBAT NEGATIVE CIRCLED DIGIT NINE

❿ 277F DINGBAT NEGATIVE CIRCLED NUMBER TEN

➀ 2780 DINGBAT CIRCLED SANS-SERIF DIGIT ONE

➁ 2781 DINGBAT CIRCLED SANS-SERIF DIGIT TWO

➂ 2782 DINGBAT CIRCLED SANS-SERIF DIGIT THREE

➃ 2783 DINGBAT CIRCLED SANS-SERIF DIGIT FOUR

➄ 2784 DINGBAT CIRCLED SANS-SERIF DIGIT FIVE

➅ 2785 DINGBAT CIRCLED SANS-SERIF DIGIT SIX

➆ 2786 DINGBAT CIRCLED SANS-SERIF DIGIT SEVEN

➇ 2787 DINGBAT CIRCLED SANS-SERIF DIGIT EIGHT

➈ 2788 DINGBAT CIRCLED SANS-SERIF DIGIT NINE

➉ 2789 DINGBAT CIRCLED SANS-SERIF NUMBER TEN

➊ 278A DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT ONE

➋ 278B DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT TWO

➌ 278C DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT THREE

➍ 278D DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT FOUR

➎ 278E DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT FIVE

➏ 278F DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT SIX

➐ 2790 DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT SEVEN

➑ 2791 DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT EIGHT

➒ 2792 DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT NINE

➓ 2793 DINGBAT NEGATIVE CIRCLED SANS-SERIF NUMBER TEN

➔ 2794 HEAVY WIDE-HEADED RIGHTWARDS ARROW

➕ 2795 HEAVY PLUS SIGN

➖ 2796 HEAVY MINUS SIGN

➗ 2797 HEAVY DIVISION SIGN

➘ 2798 HEAVY SOUTH EAST ARROW

➙ 2799 HEAVY RIGHTWARDS ARROW

➚ 279A HEAVY NORTH EAST ARROW

➛ 279B DRAFTING POINT RIGHTWARDS ARROW

➜ 279C HEAVY ROUND-TIPPED RIGHTWARDS ARROW

➝ 279D TRIANGLE-HEADED RIGHTWARDS ARROW

➞ 279E HEAVY TRIANGLE-HEADED RIGHTWARDS ARROW

➟ 279F DASHED TRIANGLE-HEADED RIGHTWARDS ARROW

➠ 27A0 HEAVY DASHED TRIANGLE-HEADED RIGHTWARDS ARROW

➡ 27A1 BLACK RIGHTWARDS ARROW

➢ 27A2 THREE-D TOP-LIGHTED RIGHTWARDS ARROWHEAD

➣ 27A3 THREE-D BOTTOM-LIGHTED RIGHTWARDS ARROWHEAD

➤ 27A4 BLACK RIGHTWARDS ARROWHEAD

➥ 27A5 HEAVY BLACK CURVED DOWNWARDS AND RIGHTWARDS ARROW

➦ 27A6 HEAVY BLACK CURVED UPWARDS AND RIGHTWARDS ARROW

➧ 27A7 SQUAT BLACK RIGHTWARDS ARROW

➨ 27A8 HEAVY CONCAVE-POINTED BLACK RIGHTWARDS ARROW

➩ 27A9 RIGHT-SHADED WHITE RIGHTWARDS ARROW

➪ 27AA LEFT-SHADED WHITE RIGHTWARDS ARROW

➫ 27AB BACK-TILTED SHADOWED WHITE RIGHTWARDS ARROW

➬ 27AC FRONT-TILTED SHADOWED WHITE RIGHTWARDS ARROW

➭ 27AD HEAVY LOWER RIGHT-SHADOWED WHITE RIGHTWARDS ARROW

➮ 27AE HEAVY UPPER RIGHT-SHADOWED WHITE RIGHTWARDS ARROW

➯ 27AF NOTCHED LOWER RIGHT-SHADOWED WHITE RIGHTWARDS ARROW

➰ 27B0 CURLY LOOP

➱ 27B1 NOTCHED UPPER RIGHT-SHADOWED WHITE RIGHTWARDS ARROW

➲ 27B2 CIRCLED HEAVY WHITE RIGHTWARDS ARROW

➳ 27B3 WHITE-FEATHERED RIGHTWARDS ARROW

➴ 27B4 BLACK-FEATHERED SOUTH EAST ARROW

➵ 27B5 BLACK-FEATHERED RIGHTWARDS ARROW

➶ 27B6 BLACK-FEATHERED NORTH EAST ARROW

➷ 27B7 HEAVY BLACK-FEATHERED SOUTH EAST ARROW

➸ 27B8 HEAVY BLACK-FEATHERED RIGHTWARDS ARROW

➹ 27B9 HEAVY BLACK-FEATHERED NORTH EAST ARROW

➺ 27BA TEARDROP-BARBED RIGHTWARDS ARROW

➻ 27BB HEAVY TEARDROP-SHANKED RIGHTWARDS ARROW

➼ 27BC WEDGE-TAILED RIGHTWARDS ARROW

➽ 27BD HEAVY WEDGE-TAILED RIGHTWARDS ARROW

➾ 27BE OPEN-OUTLINED RIGHTWARDS ARROW

➿ 27BF DOUBLE CURLY LOOP

# 第120章 CSS 听觉参考手册

CSS 听觉参考手册

"CSS" 列指示该属性在哪个 CSS 版本中定义（CSS1 或 CSS2）。

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 描述 | 值 | CSS |
| azimuth | 设置声音的来源。 | angle  left-side  far-left  left  center-left  center  center-right  right  far-right  right-side  behind  leftwards  rightwards | 2 |
| cue | 在一条声明中设置 cue 属性。 | cue-before  cue-after | 2 |
| cue-after | 规定念出元素内容后要播放的声音。 | none  url | 2 |
| cue-before | 规定念出元素内容前要播放的声音。 | url | 2 |
| elevation | 设置声音的来源。 | angle  below  level  above  higher  lower | 2 |
| pause | 在一条声明中设置暂停属性。 | pause-before  pause-after | 2 |
| pause-after | 规定在念出元素内容后暂停。 | time  % | 2 |
| pause-before | 规定在念出元素内容前暂停。 | time  % | 2 |
| pitch | 规定说话的声音。 | frequency  x-low  low  medium  high  x-high | 2 |
| pitch-range | 规定语音的变化（单调还是动听的声音？） | number | 2 |
| play-during | 规定在念出元素内容时要播放的声音。 | auto  none  url  mix  repeat | 2 |
| richness | 指定语音的丰富程度。（声音丰富还是稀薄？） | number | 2 |
| speak | 规定内容是否将以声音形式呈现。 | normal  none  spell-out | 2 |
| speak-header | 指定如何处理表格标题。应该在每个单元格之前朗读标题，还是仅在标题与前一个单元格不同的单元格之前念出标题。 | always  once | 2 |
| speak-numeral | 规定如何念出数字。 | digits  continuous | 2 |
| speak-punctuation | 规定如何念出标点符号。 | none  code | 2 |
| speech-rate | 规定说话的速度。 | number  x-slow  slow  medium  fast  x-fast  faster  slower | 2 |
| stress | 规定语音中的“压力” | number | 2 |
| voice-family | 规定语音的语音家族。 | specific-voice  generic-voice | 2 |
| volume | 规定说话的音量。 | number  %  silent  x-soft  soft  medium  loud  x-loud  2 | 2 |

# 第121章 CSS align-content 属性

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 70px;

height: 300px;

border: 1px solid #c3c3c3;

display: flex;

flex-wrap: wrap;

align-content: center;

}

#main div {

width: 70px;

height: 70px;

}

</style>

</head>

<body>

<h1>align-content 属性</h1>

<div id="main">

<div style="background-color:coral;"></div>

<div style="background-color:lightblue;"></div>

<div style="background-color:pink;"></div>

</div>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 align-content 属性。</p>

</body>

</html>

## 121.1 定义和用法

align-content 属性修改 flex-wrap 属性的行为。它与 align-items 相似，但是它不对齐弹性项目，而是对齐弹性线。

注意：必须有多行项目，此属性才能生效！

提示：使用 justify-content 属性可将主轴上的项目（main-axis）水平对齐。

默认值： stretch

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.alignContent="center"

## 121.2 CSS 语法

align-content: stretch|center|flex-start|flex-end|space-between|space-around|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| stretch | 默认值。行拉伸以占据剩余空间。 |
| center | 朝着弹性容器的中央对行打包。 |
| flex-start | 朝着弹性容器的开头对行打包。 |
| flex-end | 朝着弹性容器的结尾对行打包。 |
| space-between | 行均匀分布在弹性容器中。 |
| space-around | 行均匀分布在弹性容器中，两端各占一半。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第122章 CSS align-items 属性

实例

将弹性 <div> 元素的所有项目的居中对齐：

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 220px;

height: 300px;

border: 1px solid black;

display: flex;

align-items: center;

}

#main div {

flex: 1;

}

</style>

</head>

<body>

<div id="main">

<div style="background-color:coral;">红色</div>

<div style="background-color:lightblue;">蓝色</div>

<div style="background-color:lightgreen;">有更多内容的绿色 div</div>

</div>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 align-items 属性。</p>

</body>

</html>

## 122.1 定义和用法

align-items 属性为弹性容器内的项目指定默认对齐方式。

提示：请使用每个项目的 align-self 属性来覆盖这个 align-items 属性。

默认值： stretch

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.alignItems="center"

## 122.2 CSS 语法

align-items: stretch|center|flex-start|flex-end|baseline|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| stretch | 默认。项目被拉伸以适合容器。 |
| center | 项目位于容器的中央。 |
| flex-start | 项目位于容器的开头。 |
| flex-end | 项目位于容器的末端。 |
| baseline | 项目被定位到容器的基线。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |

# 第123章 CSS align-self 属性

实例

把弹性元素内的项目之一居中对齐：

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 220px;

height: 300px;

border: 1px solid black;

display: flex;

align-items: flex-start;

}

#main div {

flex: 1;

}

#myBlueDiv {

align-self: center;

}

</style>

</head>

<body>

<div id="main">

<div style="background-color:coral;">红色</div>

<div style="background-color:lightblue;" id="myBlueDiv">蓝色</div>

<div style="background-color:lightgreen;">有更多内容的绿色 div</div>

</div>

<p><b>注释：</b>align-self 属性会覆盖容器的 align-items 属性。</p>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 align-self 属性。</p>

</body>

</html>

## 123.1 定义和用法

align-self 属性指定弹性容器内所选项目的对齐方式。

注意：align-self 属性将覆盖弹性容器的 align-items 属性。

默认值： auto

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.alignSelf="center"

## 123.2 CSS 语法

align-self: auto|stretch|center|flex-start|flex-end|baseline|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。元素继承其父容器的 align-items 属性，如果没有父容器，则为 "stretch"。 |
| stretch | 定位元素以适合容器。 |
| center | 元素位于容器的中央。 |
| flex-start | 元素位于容器的开头。 |
| flex-end | 元素位于容器的末端。 |
| baseline | 元素被定位到容器的基线。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第124章 CSS all 属性

实例

将应用于元素或元素父元素的所有属性更改为其初始值：

<!DOCTYPE html>

<html>

<head>

<style>

html {

font-size: small;

color: blue;

}

#ex1 {

background-color: yellow;

color: red;

}

#ex2 {

background-color: yellow;

color: red;

all: inherit;

}

#ex3 {

background-color: yellow;

color: red;

all: initial;

}

#ex4 {

background-color: yellow;

color: red;

all: unset;

}

</style>

</head>

<body>

<p>未设置 all 属性：</p>

<div id="ex1">Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</div>

<p>all: inherit:</p>

<div id="ex2">Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</div>

<p>all: initial:</p>

<div id="ex3">Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</div>

<p>all: unset:</p>

<div id="ex4">Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</div>

</body>

</html>

## 124.1 定义和用法

all 属性将除 unicode-bidi 和 direction 之外的所有属性重置为其初始值或继承的值。

默认值： none

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.all="initial"

## 124.2 CSS 语法

all: initial|inherit|unset;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |
| unset | 如果可继承，则将应用于元素或元素父元素的所有属性更改为其父值，否则将其更改为初始值。 |

# 第125章 CSS animation 属性

实例

使用简写属性，将动画与 div 元素绑定：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s infinite;

-webkit-animation:mymove 5s infinite; /\*Safari and Chrome\*/

}

@keyframes mymove

{

from {left:0px;}

to {left:200px;}

}

@-webkit-keyframes mymove /\*Safari and Chrome\*/

{

from {left:0px;}

to {left:200px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation 属性。</p>

<div></div>

</body>

</html>

## 125.1 定义和用法

animation 属性是一个简写属性，用于设置六个动画属性：

animation-name

animation-duration

animation-timing-function

animation-delay

animation-iteration-count

animation-direction

注释：请始终规定 animation-duration 属性，否则时长为 0，就不会播放动画了。

默认值： none 0 ease 0 1 normal

继承性： no

版本： CSS3

JavaScript 语法： object.style.animation="mymove 5s infinite"

## 125.2 语法

animation: name duration timing-function delay iteration-count direction;

|  |  |
| --- | --- |
| 值 | 描述 |
| animation-name | 规定需要绑定到选择器的 keyframe 名称。。 |
| animation-duration | 规定完成动画所花费的时间，以秒或毫秒计。 |
| animation-timing-function | 规定动画的速度曲线。 |
| animation-delay | 规定在动画开始之前的延迟。 |
| animation-iteration-count | 规定动画应该播放的次数。 |
| animation-direction | 规定是否应该轮流反向播放动画。 |

# 第126章 CSS animation-delay 属性

实例

等待两秒，然后开始动画：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s infinite;

animation-delay:2s;

/\*Safari and Chrome\*/

-webkit-animation:mymove 5s infinite;

-webkit-animation-delay:2s;

}

@keyframes mymove

{

from {left:0px;}

to {left:200px;}

}

@-webkit-keyframes mymove /\*Safari and Chrome\*/

{

from {left:0px;}

to {left:200px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-delay 属性。</p>

<div></div>

</body>

</html>

## 126.1 定义和用法

animation-delay 属性定义动画何时开始。

animation-delay 值以秒或毫秒计。

提示：允许负值，-2s 使动画马上开始，但跳过 2 秒进入动画。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationDelay="2s"

## 126.2 语法

animation-delay: time;

|  |  |
| --- | --- |
| 值 | 描述 |
| time | 可选。定义动画开始前等待的时间，以秒或毫秒计。默认值是 0。 |

# 第127章 CSS animation-direction 属性

实例

暂停动画：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:myfirst 5s infinite;

animation-direction:alternate;

/\* Safari and Chrome \*/

-webkit-animation:myfirst 5s infinite;

-webkit-animation-direction:alternate;

}

@keyframes myfirst

{

0% {background:red; left:0px; top:0px;}

25% {background:yellow; left:200px; top:0px;}

50% {background:blue; left:200px; top:200px;}

75% {background:green; left:0px; top:200px;}

100% {background:red; left:0px; top:0px;}

}

@-webkit-keyframes myfirst /\* Safari and Chrome \*/

{

0% {background:red; left:0px; top:0px;}

25% {background:yellow; left:200px; top:0px;}

50% {background:blue; left:200px; top:200px;}

75% {background:green; left:0px; top:200px;}

100% {background:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-direction 属性。</p>

<div></div>

</body>

</html>

## 127.1 定义和用法

animation-direction 属性定义是否应该轮流反向播放动画。

如果 animation-direction 值是 "alternate"，则动画会在奇数次数（1、3、5 等等）正常播放，而在偶数次数（2、4、6 等等）向后播放。

注释：如果把动画设置为只播放一次，则该属性没有效果。

默认值： normal

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationDirection="alternate"

## 127.2 语法

animation-direction: normal|alternate;

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认值。动画应该正常播放。 |
| alternate | 动画应该轮流反向播放。 |

# 第128章 CSS animation-duration 属性

实例

为 @keyframes 动画规定一个名称：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove infinite;

animation-duration:2s;

/\* Safari and Chrome \*/

-webkit-animation:mymove infinite;

-webkit-animation-duration:2s;

}

@keyframes mymove

{

from {top:0px;}

to {top:200px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {top:0px;}

to {top:200px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-name 属性。</p>

<div></div>

<p><b>注释：</b>始终规定 animation-duration 属性，否则时长为 0，就不会播放动画了。</p>

</body>

</html>

## 128.1 定义和用法

animation-duration 属性定义动画完成一个周期所需要的时间，以秒或毫秒计。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationDuration="3s"

## 128.2语法

animation-duration: time;

|  |  |
| --- | --- |
| 值 | 描述 |
| time | 规定完成动画所花费的时间。默认值是 0，意味着没有动画效果。 |

# 第129章 CSS animation-fill-mode 属性

实例

为 h1 元素规定填充模式：

h1

{

animation-fill-mode: forwards;

}

## 129.1 定义和用法

animation-fill-mode 属性规定动画在播放之前或之后，其动画效果是否可见。

注释：其属性值是由逗号分隔的一个或多个填充模式关键词。

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationFillMode=none

## 129.2 语法

animation-fill-mode : none | forwards | backwards | both;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 不改变默认行为。 |
| forwards | 当动画完成后，保持最后一个属性值（在最后一个关键帧中定义）。 |
| backwards | 在 animation-delay 所指定的一段时间内，在动画显示之前，应用开始属性值（在第一个关键帧中定义）。 |
| both | 向前和向后填充模式都被应用。 |

# 第130章 CSS animation-iteration-count 属性

实例

播放动画三次：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 3s;

animation-iteration-count:3;

/\* Safari and Chrome \*/

-webkit-animation:mymove 3s;

-webkit-animation-iteration-count:3;

}

@keyframes mymove

{

from {top:0px;}

to {top:200px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {top:0px;}

to {top:200px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-iteration-count 属性。</p>

<div></div>

</body>

</html>

## 130.1 定义和用法

animation-iteration-count 属性定义动画的播放次数。

默认值： 1

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationIterationCount=3

## 130.2 语法

animation-iteration-count: n|infinite;

|  |  |
| --- | --- |
| 值 | 描述 |
| n | 定义动画播放次数的数值。 |
| infinite | 规定动画应该无限次播放。 |

# CSS animation-name 属性

CSS animation-name 属性

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation-name:mymove;

animation-duration:5s;

/\* Safari and Chrome \*/

-webkit-animation-name:mymove;

-webkit-animation-duration:5s;

}

@keyframes mymove

{

from {left:0px;}

to {left:200px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {left:0px;}

to {left:200px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-name 属性。</p>

<div></div>

<p><b>注释：</b>始终规定 animation-duration 属性，否则时长为 0，就不会播放动画了。</p>

</body>

</html>

## 131.1 定义和用法

animation-name 属性为 @keyframes 动画规定名称。

注释：请始终规定 animation-duration 属性，否则时长为 0，就不会播放动画了。

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationName="mymove"

## 131.2 语法

animation-name: keyframename|none;

|  |  |
| --- | --- |
| 值 | 描述 |
| keyframename | 规定需要绑定到选择器的 keyframe 的名称。 |
| none | 规定无动画效果（可用于覆盖来自级联的动画）。 |

# CSS animation-play-state 属性

实例

暂停动画：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s;

animation-play-state:paused;

/\* Safari and Chrome \*/

-webkit-animation:mymove 5s;

-webkit-animation-play-state:paused;

}

@keyframes mymove

{

from {left:0px;}

to {left:200px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {left:0px;}

to {left:200px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-play-state 属性。</p>

<div></div>

</body>

</html>

## 132.1 定义和用法

animation-play-state 属性规定动画正在运行还是暂停。

注释：您可以在 JavaScript 中使用该属性，这样就能在播放过程中暂停动画。

默认值： running

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationPlayState="paused"

## 132.2 语法

animation-play-state: paused|running;

|  |  |
| --- | --- |
| 值 | 描述 |
| paused | 规定动画已暂停。 |
| running | 规定动画正在播放。 |

# CSS animation-timing-function 属性

实例

从开头到结尾以相同的速度来播放动画：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s infinite;

animation-timing-function:linear;

/\* Safari and Chrome \*/

-webkit-animation:mymove 5s infinite;

-webkit-animation-timing-function:linear;

}

@keyframes mymove

{

from {left:0px;}

to {left:200px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {left:0px;}

to {left:200px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-timing-function 属性。</p>

<div></div>

</body>

</html>

## 133.1 定义和用法

animation-timing-function 规定动画的速度曲线。

速度曲线定义动画从一套 CSS 样式变为另一套所用的时间。

速度曲线用于使变化更为平滑。

默认值： ease

继承性： no

版本： CSS3

JavaScript 语法： object.style.animationTimingFunction="linear"

## 133.2 语法

animation-timing-function: value;

animation-timing-function 使用名为三次贝塞尔（Cubic Bezier）函数的数学函数，来生成速度曲线。您能够在该函数中使用自己的值，也可以预定义的值：

|  |  |
| --- | --- |
| 值 | 描述 |
| linear | 动画从头到尾的速度是相同的。 |
| ease | 默认。动画以低速开始，然后加快，在结束前变慢。 |
| ease-in | 动画以低速开始。 |
| ease-out | 动画以低速结束。 |
| ease-in-out | 动画以低速开始和结束。 |
| cubic-bezier(n,n,n,n) | 在 cubic-bezier 函数中自己的值。可能的值是从 0 到 1 的数值。 |

实例 1

为了更好地理解不同的定时函数值，这里提供了设置五个不同值的五个不同的 div 元素：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:50px;

background:red;

color:white;

font-weight:bold;

position:relative;

animation:mymove 5s infinite;

-webkit-animation:mymove 5s infinite; /\* Safari and Chrome \*/

}

#div1 {animation-timing-function:linear;}

#div2 {animation-timing-function:ease;}

#div3 {animation-timing-function:ease-in;}

#div4 {animation-timing-function:ease-out;}

#div5 {animation-timing-function:ease-in-out;}

/\* Safari and Chrome: \*/

#div1 {-webkit-animation-timing-function:linear;}

#div2 {-webkit-animation-timing-function:ease;}

#div3 {-webkit-animation-timing-function:ease-in;}

#div4 {-webkit-animation-timing-function:ease-out;}

#div5 {-webkit-animation-timing-function:ease-in-out;}

@keyframes mymove

{

from {left:0px;}

to {left:300px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {left:0px;}

to {left:300px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-timing-function 属性。</p>

<div id="div1">linear</div>

<div id="div2">ease</div>

<div id="div3">ease-in</div>

<div id="div4">ease-out</div>

<div id="div5">ease-in-out</div>

</body>

</html>

实例 2

与上例相同，但是通过 cubic-bezier 函数来定义速度曲线：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:50px;

background:red;

color:white;

font-weight:bold;

position:relative;

animation:mymove 5s infinite;

-webkit-animation:mymove 5s infinite; /\* Safari and Chrome \*/

}

#div1 {animation-timing-function:cubic-bezier(0,0,0.25,1);}

#div2 {animation-timing-function:cubic-bezier(0.25,0.1,0.25,1);}

#div3 {animation-timing-function:cubic-bezier(0.42,0,1,1);}

#div4 {animation-timing-function:cubic-bezier(0,0,0.58,1);}

#div5 {animation-timing-function:cubic-bezier(0.42,0,0.58,1);}

/\* Safari and Chrome: \*/

#div1 {-webkit-animation-timing-function:cubic-bezier(0,0,1,1);}

#div2 {-webkit-animation-timing-function:cubic-bezier(0.25,0.1,0.25,1);}

#div3 {-webkit-animation-timing-function:cubic-bezier(0.42,0,1,1);}

#div4 {-webkit-animation-timing-function:cubic-bezier(0,0,0.58,1);}

#div5 {-webkit-animation-timing-function:cubic-bezier(0.42,0,0.58,1);}

@keyframes mymove

{

from {left:0px;}

to {left:300px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {left:0px;}

to {left:300px;}

}

</style>

</head>

<body>

<p><strong>注释：</strong>Internet Explorer 9 以及更早的版本不支持 animation-timing-function 属性。</p>

<div id="div1">linear</div>

<div id="div2">ease</div>

<div id="div3">ease-in</div>

<div id="div4">ease-out</div>

<div id="div5">ease-in-out</div>

</body>

</html>

# 第134章 CSS backface-visibility 属性

实例

隐藏被旋转的 div 元素的背面：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

position:relative;

height:60px;

width:60px;

border:1px solid #000;

background-color:yellow;

transform:rotateY(180deg);

-webkit-transform:rotateY(180deg); /\* Chrome and Safari \*/

-moz-transform:rotateY(180deg); /\* Firefox \*/

}

#div1

{

-webkit-backface-visibility:hidden;

-moz-backface-visibility:hidden;

-ms-backface-visibility:hidden;

}

#div2

{

-webkit-backface-visibility:visible;

-moz-backface-visibility:visible;

-ms-backface-visibility:visible;

}

</style>

</head>

<body>

<p>本例有两个 div 元素，均旋转 180 度，背向用户。</p>

<p>第一个 div 元素的 backface-visibility 属性设置为 "hidden"，所以应该是不可见的。</p>

<div id="div1">DIV 1</div>

<div id="div2">DIV 2</div>

<p><b>注释：</b>本例只在 Internet Explorer 10、Firefox、Chrome 以及 Safari 中有效。</p>

</body>

</html>

## 134.1 定义和用法

backface-visibility 属性定义当元素不面向屏幕时是否可见。

如果在旋转元素不希望看到其背面时，该属性很有用。

默认值： visible

继承性： no

版本： CSS3

JavaScript 语法： object.style.backfaceVisibility="hidden"

## 134.2 语法

backface-visibility: visible|hidden;

|  |  |
| --- | --- |
| 值 | 描述 |
| visible | 背面是可见的。 |
| hidden | 背面是不可见的。 |

# 第135章 CSS background 属性

实例

如何在一个声明中设置所有背景属性：

<html>

<head>

<style type="text/css">

body

{

background: #ff0000 url(/i/eg\_bg\_03.gif) no-repeat fixed center;

}

</style>

</head>

<body>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

<p>这是一些文本。</p>

</body>

</html>

## 135.1 定义和用法

background 简写属性在一个声明中设置所有的背景属性。

可以设置如下属性：

background-color

background-position

background-size

background-repeat

background-origin

background-clip

background-attachment

background-image

如果不设置其中的某个值，也不会出问题，比如 background:#ff0000 url('smiley.gif'); 也是允许的。

通常建议使用这个属性，而不是分别使用单个属性，因为这个属性在较老的浏览器中能够得到更好的支持，而且需要键入的字母也更少。

默认值： not specified

继承性： no

版本： CSS1 + CSS3

JavaScript 语法： object.style.background="white url(paper.gif) repeat-y"

## 135.2可能的值

|  |  |  |
| --- | --- | --- |
| 值 | 描述 | CSS |
| background-color | 规定要使用的背景颜色。 | 1 |
| background-position | 规定背景图像的位置。 | 1 |
| background-size | 规定背景图片的尺寸。 | 3 |
| background-repeat | 规定如何重复背景图像。 | 1 |
| background-origin | 规定背景图片的定位区域。 | 3 |
| background-clip | 规定背景的绘制区域。 | 3 |
| background-attachment | 规定背景图像是否固定或者随着页面的其余部分滚动。 | 1 |
| background-image | 规定要使用的背景图像。 | 1 |
| inherit | 规定应该从父元素继承 background 属性的设置。 | 1 |

# 第136章 CSS background-attachment 属性

实例

如何设置固定的背景图像：

<html>

<head>

<style type="text/css">

body

{

background-image:url(/i/eg\_bg\_02.gif);

background-repeat:no-repeat;

background-attachment:fixed

}

</style>

</head>

<body>

<p>图像不会随页面的其余部分滚动。</p>

<p>A</p>

<p>B</p>

<p>C</p>

<p>D</p>

<p>E</p>

<p>F</p>

<p>G</p>

<p>H</p>

<p>I</p>

<p>J</p>

<p>K</p>

<p>L</p>

<p>M</p>

<p>N</p>

<p>O</p>

<p>P</p>

<p>Q</p>

<p>R</p>

<p>S</p>

<p>T</p>

<p>W</p>

<p>X</p>

<p>Y</p>

<p>Z</p>

<p>1</p>

<p>2</p>

<p>3</p>

<p>4</p>

<p>5</p>

<p>6</p>

<p>7</p>

<p>8</p>

<p>9</p>

</body>

</html>

## 136.1 定义和用法

background-attachment 属性设置背景图像是否固定或者随着页面的其余部分滚动。

默认值： scroll

继承性： no

版本： CSS1

JavaScript 语法： object.style.backgroundAttachment="fixed"

## 136.2 可能的值

|  |  |
| --- | --- |
| 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。 |

# 第137章 CSS background-blend-mode 属性

实例

把背景图像的混合模式指定为 "lighten"（变亮）：

<!DOCTYPE html>

<html>

<head>

<style>

#myDIV {

width: 400px;

height: 400px;

background-repeat: no-repeat, repeat;

background-image: url("/i/photo/tree.png"), url("/i/paper.jpg");

background-blend-mode: lighten;

}

</style>

</head>

<body>

<h1>background-blend-mode 属性</h1>

<div id="myDIV"></div>

<p><b>注释：</b>79 之前的 Edge 不支持 background-blend-mode 属性。</p>

</body>

</html>

## 137.1 定义和用法

background-blend-mode 属性定义每个背景层（颜色和/或图像）的混合模式。

默认值： normal

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.backgroundBlendMode="screen"

## 137.2 CSS 语法

background-blend-mode: normal|multiply|screen|overlay|darken|lighten|color-dodge|saturation|color|luminosity;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认。把混合模式设置为普通。 |
| multiply | 把混合模式设置为 multiply（乘）。 |
| screen | 把混合模式设置为 screen（屏幕）。 |
| overlay | 把混合模式设置为 overlay（覆盖）。 |
| darken | 把混合模式设置为 overlay（覆盖）。 |
| lighten | 把混合模式设置为 lighten（变亮）。 |
| color-dodge | 把混合模式设置为 color-dodge（颜色减淡）。 |
| saturation | 把混合模式设置为 saturation（饱和度）。 |
| color | 把混合模式设置为 color（颜色）。 |
| luminosity | 把混合模式设置为 luminosity（亮度）。 |

# 第138章 CSS background-clip 属性

实例

规定背景的绘制区域：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:300px;

height:300px;

padding:50px;

background-color:yellow;

background-clip:content-box;

border:2px solid #92b901;

}

</style>

</head>

<body>

<div>

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

</div>

</body>

</html>

## 137.1 定义和用法

background-clip 属性规定背景的绘制区域。

默认值： border-box

继承性： no

版本： CSS3

JavaScript 语法： object.style.backgroundClip="content-box"

## 137.2 语法

background-clip: border-box|padding-box|content-box;

|  |  |
| --- | --- |
| 值 | 描述 |
| border-box | 背景被裁剪到边框盒。 |
| padding-box | 背景被裁剪到内边距框。 |
| content-box | 背景被裁剪到内容框。 |

# 第138章 CSS background-color 属性

<html>

<head>

<style type="text/css">

body {background-color: yellow}

h1 {background-color: #00ff00}

h2 {background-color: transparent}

p {background-color: rgb(250,0,255)}

p.no2 {background-color: gray; padding: 20px;}

</style>

</head>

<body>

<h1>这是标题 1</h1>

<h2>这是标题 2</h2>

<p>这是段落</p>

<p class="no2">这个段落设置了内边距。</p>

</body>

</html>

## 138.1 定义和用法

background-color 属性设置元素的背景颜色。

元素背景的范围

background-color 属性为元素设置一种纯色。这种颜色会填充元素的内容、内边距和边框区域，扩展到元素边框的外边界（但不包括外边距）。如果边框有透明部分（如虚线边框），会透过这些透明部分显示出背景色。

transparent 值

尽管在大多数情况下，没有必要使用 transparent。不过如果您不希望某元素拥有背景色，同时又不希望用户对浏览器的颜色设置影响到您的设计，那么设置 transparent 值还是有必要的。

默认值： transparent

继承性： no

版本： CSS1

JavaScript 语法： object.style.backgroundColor="#00FF00"

## 138.2 可能的值

|  |  |
| --- | --- |
| 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。 |

# 第139章 CSS background-image 属性

<html>

<head>

<style type="text/css">

body {background-image:url(/i/eg\_bg\_04.gif);}

</style>

</head>

<body></body>

</html>

## 139.1 定义和用法

background-image 属性为元素设置背景图像。

元素的背景占据了元素的全部尺寸，包括内边距和边框，但不包括外边距。

默认地，背景图像位于元素的左上角，并在水平和垂直方向上重复。

提示：请设置一种可用的背景颜色，这样的话，假如背景图像不可用，页面也可获得良好的视觉效果。

详细说明

background-image 属性会在元素的背景中设置一个图像。

根据 background-repeat 属性的值，图像可以无限平铺、沿着某个轴（x 轴或 y 轴）平铺，或者根本不平铺。

初始背景图像（原图像）根据 background-position 属性的值放置。

默认值： none

继承性： no

版本： CSS1

JavaScript 语法： object.style.backgroundImage="url(stars.gif)"

## 139.2 可能的值

|  |  |
| --- | --- |
| 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。 |

# 第140章 CSS background-origin 属性

实例

相对于内容框来定位背景图像：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

border:1px solid black;

padding:35px;

background-image:url('/i/bg\_flower.gif');

background-repeat:no-repeat;

background-position:left;

}

#div1

{

background-origin:border-box;

}

#div2

{

background-origin:content-box;

}

</style>

</head>

<body>

<p>background-origin:border-box:</p>

<div id="div1">

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

</div>

<p>background-origin:content-box:</p>

<div id="div2">

这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。这是文本。

</div>

</body>

</html>

## 140.1 定义和用法

background-origin 属性规定 background-position 属性相对于什么位置来定位。

注释：如果背景图像的 background-attachment 属性为 "fixed"，则该属性没有效果。

默认值： padding-box

继承性： no

版本： CSS3

JavaScript 语法： object.style.backgroundOrigin="content-box"

## 140.2 语法

background-origin: padding-box|border-box|content-box;

|  |  |
| --- | --- |
| 值 | 描述 |
| padding-box | 背景图像相对于内边距框来定位。 |
| border-box | 背景图像相对于边框盒来定位。 |
| content-box | 背景图像相对于内容框来定位。 |

# 第141章 CSS background-position 属性

实例

如何定位背景图像：

<html>

<head>

<style type="text/css">

body

{

background-image:url('/i/eg\_bg\_03.gif');

background-repeat:no-repeat;

background-attachment:fixed;

background-position:center;

}

</style>

</head>

<body>

<body>

<p><b>提示：</b>您需要把 background-attachment 属性设置为 "fixed"，才能保证该属性在 Firefox 和 Opera 中正常工作。</p>

</body>

</body>

</html>

## 141.1 定义和用法

background-position 属性设置背景图像的起始位置。

这个属性设置背景原图像（由 background-image 定义）的位置，背景图像如果要重复，将从这一点开始。

提示：您需要把 background-attachment 属性设置为 "fixed"，才能保证该属性在 Firefox 和 Opera 中正常工作。

默认值： 0% 0%

继承性： no

版本： CSS1

JavaScript 语法： object.style.backgroundPosition="center"

## 141.2 可能的值

|  |  |
| --- | --- |
| 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。 |

# 第142章 CSS background-repeat 属性

<html>

<head>

<style type="text/css">

body

{

background-image:

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

background-repeat: repeat-y

}

</style>

</head>

<body>

</body>

</html>

## 142.1 定义和用法

background-repeat 属性设置是否及如何重复背景图像。

默认地，背景图像在水平和垂直方向上重复。

详细说明

background-repeat 属性定义了图像的平铺模式。

从原图像开始重复，原图像由 background-image 定义，并根据 background-position 的值放置。

默认值： repeat

继承性： no

版本： CSS1

JavaScript 语法： object.style.backgroundRepeat="repeat-y"

142.2 提示和注释

提示：背景图像的位置是根据 background-position 属性设置的。如果未规定 background-position 属性，图像会被放置在元素的左上角。

## 142.3 可能的值

|  |  |
| --- | --- |
| 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。 |

## 第143章 CSS background-size 属性

规定背景图像的尺寸：

<!DOCTYPE html>

<html>

<head>

<style>

body

{

background:url(/i/bg\_flower.gif);

background-size:63px 100px;

-moz-background-size:63px 100px; /\* 老版本的 Firefox \*/

background-repeat:no-repeat;

padding-top:80px;

}

</style>

</head>

<body>

<p>上面是缩小的背景图片。</p>

<p>原始图片：<img src="/i/bg\_flower.gif" alt="Flowers"></p>

</body>

</html>

## 143.1 定义和用法

background-size 属性规定背景图像的尺寸。

默认值： auto

继承性： no

版本： CSS3

JavaScript 语法： object.style.backgroundSize="60px 80px"

## 143.2 语法

background-size: length|percentage|cover|contain;

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 设置背景图像的高度和宽度。  第一个值设置宽度，第二个值设置高度。  如果只设置一个值，则第二个值会被设置为 "auto"。 |
| percentage | 以父元素的百分比来设置背景图像的宽度和高度。  第一个值设置宽度，第二个值设置高度。  如果只设置一个值，则第二个值会被设置为 "auto"。 |
| cover | 把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。  背景图像的某些部分也许无法显示在背景定位区域中。 |
| contain | 把图像图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域。 |

# 第144章 CSS border 属性

设置 4 个边框的样式：

<html>

<head>

<style type="text/css">

p

{

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

}

</style>

</head>

<body>

<p>Some text</p>

</body>

</html>

## 144.1 定义和用法

border 简写属性在一个声明设置所有的边框属性。

可以按顺序设置如下属性：

border-width

border-style

border-color

如果不设置其中的某个值，也不会出问题，比如 border:solid #ff0000; 也是允许的。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.border="3px solid blue"

144.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| border-width | 规定边框的宽度。参阅：border-width 中可能的值。 |
| border-style | 规定边框的样式。参阅：border-style 中可能的值。 |
| border-color | 规定边框的颜色。参阅：border-color 中可能的值。 |
| inherit | 规定应该从父元素继承 border 属性的设置。 |

# 第145章 CSS border-bottom 属性

设置下边框的样式：

<html>

<head>

<style type="text/css">

p

{

border-style:solid;

border-bottom:thick dotted #ff0000;

}

</style>

</head>

<body>

<p>This is some text in a paragraph.</p>

</body>

</html>

## 145.1 定义和用法

border-bottom 简写属性把下边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

border-bottom-width

border-bottom-style

border-bottom-color

如果不设置其中的某个值，也不会出问题，比如 border-bottom:solid #ff0000; 也是允许的。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderBottom="3px solid blue"

## 145.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| border-bottom-width | 规定下边框的宽度。参阅：border-bottom-width 中可能的值。 |
| border-bottom-style | 规定下边框的样式。参阅：border-bottom-style 中可能的值。 |
| border-bottom-color | 规定下边框的颜色。参阅：border-bottom-color 中可能的值。 |
| inherit | 规定应该从父元素继承 border-bottom 属性的设置。 |

# 第146章 CSS border-bottom-color 属性

设置下边框的颜色：

<html>

<head>

<style type="text/css">

p

{

border-style:solid;

border-bottom-color:#ff0000;

}

</style>

</head>

<body>

<p>This is some text in a paragraph.</p>

</body>

</html>

## 146.1 定义和用法

border-bottom-color 设置元素的下边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderBottomColor="blue"

146.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| color\_name | 规定颜色值为颜色名称的边框颜色（比如 red）。 |
| hex\_number | 规定颜色值为十六进制值的边框颜色（比如 #ff0000）。 |
| rgb\_number | 规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。 |
| transparent | 默认值。边框颜色为透明。 |
| inherit | 规定应该从父元素继承边框颜色。 |

# CSS border-bottom-left-radius 属性

向 div 元素的左下角添加圆角边框：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

border:2px solid;

padding:10px;

background:#dddddd;

border-bottom-left-radius:2em;

}

</style>

</head>

<body>

<div>border-bottom-left-radius 属性允许您向左下角添加圆角边框。</div>

</body>

</html>

## 147.1 定义和用法

border-bottom-left-radius 属性定义左下角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderBottomLeftRadius="5px"

## 147.2 语法

border-bottom-left-radius: length|% [length|%];

注释：border-bottom-left-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 定义左下角的形状。 |
| % | 以百分比值定义左下角的形状。 |

# 第148章 CSS border-bottom-right-radius 属性

向 div 元素的右下角添加圆角边框：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

border:2px solid;

padding:10px;

background:#dddddd;

border-bottom-right-radius:2em;

}

</style>

</head>

<body>

<div>border-bottom-right-radius 属性允许您向右下角添加圆角边框。</div>

</body>

</html>

## 148.1 定义和用法

border-bottom-right-radius 属性定义右下角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderBottomRightRadius="5px"

## 148.2 语法

border-bottom-right-radius: length|% [length|%];

注释：border-bottom-right-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 定义右下角的形状。 |
| % | 以百分比值定义右下角的形状。 |

# 第149章 CSS border-bottom-style 属性

设置下边框的样式：

<html>

<head>

<style type="text/css">

p {border-style:solid}

p.none {border-bottom-style:none}

p.dotted {border-bottom-style:dotted}

p.dashed {border-bottom-style:dashed}

p.solid {border-bottom-style:solid}

p.double {border-bottom-style:double}

p.groove {border-bottom-style:groove}

p.ridge {border-bottom-style:ridge}

p.inset {border-bottom-style:inset}

p.outset {border-bottom-style:outset}

</style>

</head>

<body>

<p class="none">No bottom border.</p>

<p class="dotted">A dotted bottom border.</p>

<p class="dashed">A dashed bottom border.</p>

<p class="solid">A solid bottom border.</p>

<p class="double">A double bottom border.</p>

<p class="groove">A groove bottom border.</p>

<p class="ridge">A ridge bottom border.</p>

<p class="inset">An inset bottom border.</p>

<p class="outset">An outset bottom border.</p>

</body>

</html>

## 149.1 定义和用法

border-bottom-style 设置元素下边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderBottomStyle="dotted"

## 149.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 定义无边框。 |
| hidden | 与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。 |
| dotted | 定义点状边框。在大多数浏览器中呈现为实线。 |
| dashed | 定义虚线。在大多数浏览器中呈现为实线。 |
| solid | 定义实线。 |
| double | 定义双线。双线的宽度等于 border-width 的值。 |
| groove | 定义 3D 凹槽边框。其效果取决于 border-color 的值。 |
| ridge | 定义 3D 垄状边框。其效果取决于 border-color 的值。 |
| inset | 定义 3D inset 边框。其效果取决于 border-color 的值。 |
| outset | 定义 3D outset 边框。其效果取决于 border-color 的值。 |
| inherit | 规定应该从父元素继承边框样式。 |

# 第150章 CSS border-bottom-width 属性

设置下边框的宽度：

<html>

<head>

<style type="text/css">

p.one

{

border-style: solid;

border-bottom-width: 15px

}

p.two

{

border-style: solid;

border-bottom-width: thin

}

</style>

</head>

<body>

<p class="one"><b>注释：</b>"border-bottom-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。</p>

<p class="two">Some text. Some more text.</p>

</body>

</html>

## 150.1 定义和用法

border-bottom-width 属性设置元素的下边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-bottom-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值： medium

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderBottomWidth="thick"

## 150.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| thin | 定义细的下边框。 |
| medium | 默认值。定义中等的下边框。 |
| thick | 定义粗的下边框。 |
| length | 允许您自定义下边框的宽度。 |
| inherit | 规定应该从父元素继承边框宽度。 |

# 第151章 CSS border-collapse 属性

为表格设置合并边框模型：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<style type="text/css">

table

{

border-collapse:collapse;

}

table, td, th

{

border:1px solid black;

}

</style>

</head>

<body>

<table>

<tr>

<th>Firstname</th>

<th>Lastname</th>

</tr>

<tr>

<td>Bill</td>

<td>Gates</td>

</tr>

<tr>

<td>Steven</td>

<td>Jobs</td>

</tr>

</table>

<p><b>注释：</b>如果没有规定 !DOCTYPE，border-collapse 属性可能会引起意想不到的错误。</p>

</body>

</html>

## 151.1 定义和用法

border-collapse 属性设置表格的边框是否被合并为一个单一的边框，还是象在标准的 HTML 中那样分开显示。

默认值： separate

继承性： yes

版本： CSS2

JavaScript 语法： object.style.borderCollapse="collapse"

## 151.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| separate | 默认值。边框会被分开。不会忽略 border-spacing 和 empty-cells 属性。 |
| collapse | 如果可能，边框会合并为一个单一的边框。会忽略 border-spacing 和 empty-cells 属性。 |
| inherit | 规定应该从父元素继承 border-collapse 属性的值。 |

# 第152章 CSS border-color 属性

设置 4 个边框的颜色：

<html>

<head>

<style type="text/css">

p.one

{

border-style: solid;

border-color: #0000ff

}

p.two

{

border-style: solid;

border-color: #ff0000 #0000ff

}

p.three

{

border-style: solid;

border-color: #ff0000 #00ff00 #0000ff

}

p.four

{

border-style: solid;

border-color: #ff0000 #00ff00 #0000ff rgb(250,0,255)

}

</style>

</head>

<body>

<p class="one">One-colored border!</p>

<p class="two">Two-colored border!</p>

<p class="three">Three-colored border!</p>

<p class="four">Four-colored border!</p>

<p><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。</p>

</body>

</html>

## 152.1 定义和用法

border-color 属性设置四条边框的颜色。此属性可设置 1 到 4 种颜色。

border-color 属性是一个简写属性，可设置一个元素的所有边框中可见部分的颜色，或者为 4 个边分别设置不同的颜色。请看下面的例子：

例子 1

border-color:red green blue pink;

上边框是红色

右边框是绿色

下边框是蓝色

左边框是粉色

例子 2

border-color:red green blue;

上边框是红色

右边框和左边框是绿色

下边框是蓝色

例子 3

border-color:dotted red green;

上边框和下边框是红色

右边框和左边框是绿色

例子 4

border-color:red;

所有 4 个边框都是红色

要记住，边框的样式不能为 none 或 hidden，否则边框不会出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderColor="#FF0000 blue"

## 152.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| color\_name | 规定颜色值为颜色名称的边框颜色（比如 red）。 |
| hex\_number | 规定颜色值为十六进制值的边框颜色（比如 #ff0000）。 |
| rgb\_number | 规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。 |
| transparent | 默认值。边框颜色为透明。 |
| inherit | 规定应该从父元素继承边框颜色。 |

# 第153章 CSS border-image 属性

将图片规定为包围 div 元素的边框：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

border:15px solid transparent;

width:300px;

padding:10px 20px;

}

#round

{

-moz-border-image:url(/i/border.png) 30 30 round; /\* Old Firefox \*/

-webkit-border-image:url(/i/border.png) 30 30 round; /\* Safari and Chrome \*/

-o-border-image:url(/i/border.png) 30 30 round; /\* Opera \*/

border-image:url(/i/border.png) 30 30 round;

}

#stretch

{

-moz-border-image:url(/i/border.png) 30 30 stretch; /\* Old Firefox \*/

-webkit-border-image:url(/i/border.png) 30 30 stretch; /\* Safari and Chrome \*/

-o-border-image:url(/i/border.png) 30 30 stretch; /\* Opera \*/

border-image:url(/i/border.png) 30 30 stretch;

}

</style>

</head>

<body>

<div id="round">在这里，图片铺满整个边框。</div>

<br>

<div id="stretch">在这里，图片被拉伸以填充该区域。</div>

<p>这是我们使用的图片：</p>

<img src="/i/border.png">

<p><b>注释：</b> Internet Explorer 不支持 border-image 属性。</p>

<p>border-image 属性规定了用作边框的图片。</p>

</body>

</html>

## 153.1 定义和用法

border-image 属性是一个简写属性，用于设置以下属性：

border-image-source

border-image-slice

border-image-width

border-image-outset

border-image-repeat

如果省略值，会设置其默认值。

提示：请使用 border-image-\* 属性来构造漂亮的可伸缩按钮！

默认值： none 100% 1 0 stretch

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderImage="url(border.png) 30 30 round"

## 153.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| border-image-source | 用在边框的图片的路径。 |
| border-image-slice | 图片边框向内偏移。 |
| border-image-width | 图片边框的宽度。 |
| border-image-outset | 边框图像区域超出边框的量。 |
| border-image-repeat | 图像边框是否应平铺(repeated)、铺满(rounded)或拉伸(stretched)。 |

# 第154章 CSS border-image-outset 属性

设置 border-image-outset 属性：

div

{

border-image-source: url(border.png);

border-image-outset: 30 30;

}

## 154.1 定义和用法

border-image-outset 属性规定边框图像超过边框盒的量。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderImageOutset="30 30"

## 154.2 可能的值

border-image-outset: length|number;

注释：border-image-outset 属性规定边框图像超出边框盒的量。在上、右、下、左侧。如果忽略第四个值，则与第二个值相同。如果省略第三个值，则与第一个值相同。如果省略第二个值，则与第一个值相同。不允许任何负值作为 border-image-outset 值。

|  |  |
| --- | --- |
| 值 | 描述 |
| length |  |
| number | 代表对应的 border-width 的倍数。 |

# 第155章 CSS border-image-repeat 属性

规定如何重复图像边框：

div

{

border-image-source: url(border.png);

border-image-repeat: round;

}

## 155.1 定义和用法

border-image-repeat 属性规定图像边框是否应该被重复（repeated）、拉伸（stretched）或铺满（rounded）。

默认值： stretch

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderImageRepeat="round"

## 155.2可能的值

border-image-repeat: stretch|repeat|round;

注释：该属性规定如何延展和铺排边框图像的边缘和中间部分。因此，您可以规定两个值。如果省略第二个值，则采取与第一个值相同的值。

|  |  |
| --- | --- |
| 值 | 描述 |
| stretch | 拉伸图像来填充区域 |
| repeat | 平铺（重复）图像来填充区域。 |
| round | 类似 repeat 值。如果无法完整平铺所有图像，则对图像进行缩放以适应区域。 |

# 第156章 CSS border-image-slice 属性

规定图像边框的向内偏移：

div

{

border-image-source: url(border.png);

border-image-slice: 50% 50%;

}

## 156.1 定义和用法

border-image-slice 属性规定图像边框的向内偏移。

默认值： 100%

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderImageSlice="50% 50%"

## 156.2 语法

border-image-slice: number|%|fill;

注释：该属性规定图像的上、右、下、左侧边缘的向内偏移，图像被分割为九个区域：四个角、四条边以及一个中间区域。除非使用了关键词 fill，否则中间的图像部分会被丢弃。如果省略第四个数值/百分比，则与第二个值相同。如果省略第三个值，则与第一个值相同。如果省略第二个值，则与第一个值相同。

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 数字值，代表图像中像素（如果是光栅图像）或矢量坐标（如果是矢量图像）。 |
| % | 相对于图像尺寸的百分比值：图像的宽度影响水平偏移，高度影响垂直偏移。 |
| fill | 保留边框图像的中间部分。 |

# 第157章 CSS border-image-source 属性

使用一幅图像作为围绕 div 元素的边框：

div

{

border-image-source: url(border.png);

}

## 157.1 定义和用法

border-image-source 属性规定要使用的图像，代替 border-style 属性中设置的边框样式。

提示：如果值为 "none"，或者如果图像无法显示，则使用边框样式。

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderImageSource="url(border.png)"

## 157.2 语法

border-image-source: none|image;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 不使用图像。 |
| image | 用作边框的图像的路径。 |

# 第158章 CSS border-image-width 属性

规定图像边框的宽度：

div

{

border-image-source: url(border.png);

border-image-width: 30 30;

}

## 158.1 定义和用法

border-image-width 属性规定图像边框的宽度。

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderImageWidth="30 30"

## 158.2 语法

border-image-width: number|%|auto;

注释：border-image-width 属性的四个之规定将边框图像分割为九个部分的偏移。它们代表了从区域的上、右、下、左侧向内的距离。如果忽略第四个值，则与第二个值相同。如果省略第三个值，则与第一个值相同。如果省略第二个值，则与第一个值相同。不允许任何负值作为 border-image-width 值。

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 代表对应的 border-width 的倍数。 |
| % | 参考边框图像区域的尺寸：区域的高度影响水平偏移，宽度影响垂直偏移。 |
| auto | 如果规定该属性，则宽度为对应的图像切片的固有宽度。 |

# 第159章 CSS border-left 属性

设置左边框的样式：

<html>

<head>

<style type="text/css">

p

{

border-style:solid;

border-left:thick double #ff0000;

}

</style>

</head>

<body>

<p>This is some text in a paragraph.</p>

</body>

</html>

## 159.1 定义和用法

border-left 简写属性把左边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

border-left-width

border-left-style

border-left-color

如果不设置其中的某个值，也不会出问题，比如 border-left:solid #ff0000; 也是允许的。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderLeft="3px solid blue"

## 159.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| border-left-width | 规定左边框的宽度。参阅：border-left-width 中可能的值。 |
| border-left-style | 规定左边框的样式。参阅：border-left-style 中可能的值。 |
| border-left-color | 规定左边框的颜色。参阅：border-left-color 中可能的值。 |
| inherit | 规定应该从父元素继承 border-left 属性的设置。 |

# 第160章 CSS border-left-color 属性

设置左边框的颜色：

<html>

<head>

<style type="text/css">

p

{

border-style: solid;

border-left-color: #ff0000

}

</style>

</head>

<body>

<p>Some text.</p>

</body>

</html>

## 160.1 定义和用法

border-left-color 设置元素的左边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderLeftColor="blue"

## 160.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| color\_name | 规定颜色值为颜色名称的边框颜色（比如 red）。 |
| hex\_number | 规定颜色值为十六进制值的边框颜色（比如 #ff0000）。 |
| rgb\_number | 规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。 |
| transparent | 默认值。边框颜色为透明。 |
| inherit | 规定应该从父元素继承边框颜色。 |

# 第161章 CSS border-left-style 属性

设置左边框的样式

<html>

<head>

<style type="text/css">

p

{

border-style:solid;

}

p.none {border-left-style:none}

p.dotted {border-left-style:dotted}

p.dashed {border-left-style:dashed}

p.solid {border-left-style:solid}

p.double {border-left-style:double}

p.groove {border-left-style:groove}

p.ridge {border-left-style:ridge}

p.inset {border-left-style:inset}

p.outset {border-left-style:outset}

</style>

</head>

<body>

<p class="none">No left border.</p>

<p class="dotted">A dotted left border.</p>

<p class="dashed">A dashed left border.</p>

<p class="solid">A solid left border.</p>

<p class="double">A double left border.</p>

<p class="groove">A groove left border.</p>

<p class="ridge">A ridge left border.</p>

<p class="inset">An inset left border.</p>

<p class="outset">An outset left border.</p>

</body>

</html>

## 161.1 定义和用法

border-left-style 设置元素左边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderLeftStyle="dotted"

## 161.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 定义无边框。 |
| hidden | 与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。 |
| dotted | 定义点状边框。在大多数浏览器中呈现为实线。 |
| dashed | 定义虚线。在大多数浏览器中呈现为实线。 |
| solid | 定义实线。 |
| double | 定义双线。双线的宽度等于 border-width 的值。 |
| groove | 定义 3D 凹槽边框。其效果取决于 border-color 的值。 |
| ridge | 定义 3D 垄状边框。其效果取决于 border-color 的值。 |
| inset | 定义 3D inset 边框。其效果取决于 border-color 的值。 |
| outset | 定义 3D outset 边框。其效果取决于 border-color 的值。 |
| inherit | 规定应该从父元素继承边框样式。 |

# 第162章 CSS border-left-width 属性

设置左边框的宽度：

<html>

<head>

<style type="text/css">

p.one

{

border-style: solid;

border-left-width: 15px

}

p.two

{

border-style: solid;

border-left-width: thin

}

</style>

</head>

<body>

<p class="one"><b>注释：</b>"border-left-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。</p>

<p class="two">Some text. Some more text.</p>

</body>

</html>

## 162.1 定义和用法

border-left-width 属性设置元素的左边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-left-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值： medium

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderLeftWidth="thick"

## 162.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| thin | 定义细的左边框。 |
| medium | 默认值。定义中等的左边框。 |
| thick | 定义粗的左边框。 |
| length | 允许您自定义左边框的宽度。 |
| inherit | 规定应该从父元素继承边框宽度。 |

# 第163章 CSS border-radius 属性

向 div 元素添加圆角边框：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

text-align:center;

border:2px solid #a1a1a1;

padding:10px 40px;

background:#dddddd;

width:350px;

border-radius:25px;

-moz-border-radius:25px; /\* 老的 Firefox \*/

}

</style>

</head>

<body>

<div>border-radius 属性允许您向元素添加圆角。</div>

</body>

</html>

## 163.1 定义和用法

border-radius 属性是一个简写属性，用于设置四个 border-\*-radius 属性。

提示：该属性允许您为元素添加圆角边框！

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderRadius="5px"

## 163.2 语法

border-radius: 1-4 length|% / 1-4 length|%;

注释：按此顺序设置每个 radii 的四个值。如果省略 bottom-left，则与 top-right 相同。如果省略 bottom-right，则与 top-left 相同。如果省略 top-right，则与 top-left 相同。

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 定义圆角的形状。 |
| % | 以百分比定义圆角的形状。 |

例子 1

border-radius:2em;

等价于：

border-top-left-radius:2em;

border-top-right-radius:2em;

border-bottom-right-radius:2em;

border-bottom-left-radius:2em;

例子 2

border-radius: 2em 1em 4em / 0.5em 3em;

等价于：

border-top-left-radius: 2em 0.5em;

border-top-right-radius: 1em 3em;

border-bottom-right-radius: 4em 0.5em;

border-bottom-left-radius: 1em 3em;

# 第164章 CSS border-right 属性

设置右边框的样式：

<html>

<head>

<style type="text/css">

p

{

border-style:solid;

border-right:thick double #ff0000;

}

</style>

</head>

<body>

<p>This is some text in a paragraph.</p>

</body>

</html>

## 164.1 定义和用法

border-right 简写属性把右边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

border-right-width

border-right-style

border-right-color

如果不设置其中的某个值，也不会出问题，比如 border-right:solid #ff0000; 也是允许的。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderRight="3px solid blue"

## 164.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| border-right-width | 规定右边框的宽度。参阅：border-right-width 中可能的值。 |
| border-right-style | 规定右边框的样式。参阅：border-right-style 中可能的值。 |
| border-right-color | 规定右边框的颜色。参阅：border-right-color 中可能的值。 |
| inherit | 规定应该从父元素继承 border-right 属性的设置。 |

# 第165章 CSS border-right-color 属性

设置右边框的颜色：

<html>

<head>

<style type="text/css">

p

{

border-style: solid;

border-right-color: #ff0000

}

</style>

</head>

<body>

<p>Some text.</p>

</body>

</html>

## 165.1 定义和用法

border-right-color 设置元素的右边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderRightColor="blue"

## 165.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| color\_name | 规定颜色值为颜色名称的边框颜色（比如 red）。 |
| hex\_number | 规定颜色值为十六进制值的边框颜色（比如 #ff0000）。 |
| rgb\_number | 规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。 |
| transparent | 默认值。边框颜色为透明。 |
| inherit | 规定应该从父元素继承边框颜色。 |

# 第166章 CSS border-right-style 属性

设置右边框的样式：

<!DOCTYPE html>

<html>

<head>

<style type="text/css">

p.dotted {border-right-style: dotted}

p.dashed {border-right-style: dashed}

p.solid {border-right-style: solid}

p.double {border-right-style: double}

p.groove {border-right-style: groove}

p.ridge {border-right-style: ridge}

p.inset {border-right-style: inset}

p.outset {border-right-style: outset}

</style>

</head>

<body>

<p class="dotted">A dotted border</p>

<p class="dashed">A dashed border</p>

<p class="solid">A solid border</p>

<p class="double">A double border</p>

<p class="groove">A groove border</p>

<p class="ridge">A ridge border</p>

<p class="inset">An inset border</p>

<p class="outset">An outset border</p>

</body>

</html>

## 166.1 定义和用法

border-right-style 设置元素右边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderRightStyle="dotted"

## 166.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 定义无边框。 |
| hidden | 与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。 |
| dotted | 定义点状边框。在大多数浏览器中呈现为实线。 |
| dashed | 定义虚线。在大多数浏览器中呈现为实线。 |
| solid | 定义实线。 |
| double | 定义双线。双线的宽度等于 border-width 的值。 |
| groove | 定义 3D 凹槽边框。其效果取决于 border-color 的值。 |
| ridge | 定义 3D 垄状边框。其效果取决于 border-color 的值。 |
| inset | 定义 3D inset 边框。其效果取决于 border-color 的值。 |
| outset | 定义 3D outset 边框。其效果取决于 border-color 的值。 |
| inherit | 规定应该从父元素继承边框样式。 |

# 第167章 CSS border-right-width 属性

设置右边框的宽度：

p

{

border-style:solid;

border-right-width:15px;

}

## 167.1 定义和用法

border-right-width 属性设置元素的右边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-right-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值： medium

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderRightWidth="thick"

## 167.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| thin | 定义细的右边框。 |
| medium | 默认值。定义中等的右边框。 |
| thick | 定义粗的右边框。 |
| length | 允许您自定义右边框的宽度。 |
| inherit | 规定应该从父元素继承边框宽度。 |

# 第168章 CSS border-spacing 属性

为表格设置 border-spacing：

table

{

border-collapse:separate;

border-spacing:10px 50px;

}

## 168.1 定义和用法

border-spacing 属性设置相邻单元格的边框间的距离（仅用于“边框分离”模式）。

注释：某些版本的IE浏览器不支持此属性。

说明

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"

## 168.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| length length | 规定相邻单元的边框之间的距离。使用 px、cm 等单位。不允许使用负值。  如果定义一个 length 参数，那么定义的是水平和垂直间距。  如果定义两个 length 参数，那么第一个设置水平间距，而第二个设置垂直间距。 |
| inherit | 规定应该从父元素继承 border-spacing 属性的值。 |

# 第169章 CSS border-style 属性

设置 4 个边框的样式：

p

{

border-style:solid;

}

## 169.1 定义和用法

border-style 属性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。

只有当这个值不是 none 时边框才可能出现。

例子 1

border-style:dotted solid double dashed;

上边框是点状

右边框是实线

下边框是双线

左边框是虚线

例子 2

border-style:dotted solid double;

上边框是点状

右边框和左边框是实线

下边框是双线

例子 3

border-style:dotted solid;

上边框和下边框是点状

右边框和左边框是实线

例子 4

border-style:dotted;

所有 4 个边框都是点状

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderStyle="dotted double"

## 169.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 定义无边框。 |
| hidden | 与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。 |
| dotted | 定义点状边框。在大多数浏览器中呈现为实线。 |
| dashed | 定义虚线。在大多数浏览器中呈现为实线。 |
| solid | 定义实线。 |
| double | 定义双线。双线的宽度等于 border-width 的值。 |
| groove | 定义 3D 凹槽边框。其效果取决于 border-color 的值。 |
| ridge | 定义 3D 垄状边框。其效果取决于 border-color 的值。 |
| inset | 定义 3D inset 边框。其效果取决于 border-color 的值。 |
| outset | 定义 3D outset 边框。其效果取决于 border-color 的值。 |
| inherit | 规定应该从父元素继承边框样式。 |

描述

最不可预测的边框样式是 double。它定义为两条线的宽度再加上这两条线之间的空间等于 border-width 值。不过，CSS 规范并没有说其中一条线是否比另一条粗或者两条线是否应该是一样的粗，也没有指出线之间的空间是否应当比线粗。所有这些都有用户代理决定，创作人员对这个决定没有任何影响。

# 第170章 CSS border-top 属性

设置上边框的样式：

p

{

border-style:solid;

border-top:thick double #ff0000;

}

## 170.1 定义和用法

border-top 简写属性把上边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

border-top-width

border-top-style

border-top-color

如果不设置其中的某个值，也不会出问题，比如 border-top:solid #ff0000; 也是允许的。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderTop="3px solid blue"

## 170.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| border-top-width | 规定上边框的宽度。参阅：border-top-width 中可能的值。 |
| border-top-style | 规定上边框的样式。参阅：border-top-style 中可能的值。 |
| border-top-color | 规定上边框的颜色。参阅：border-top-color 中可能的值。 |
| inherit | 规定应该从父元素继承 border-top 属性的设置。 |

# 第171章 CSS border-top-color 属性

设置上边框的颜色：

p

{

border-style:solid;

border-top-color:#ff0000;

}

## 171.1 定义和用法

border-top-color 设置元素的上边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderTopColor="blue"定义和用法

border-top-color 设置元素的上边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderTopColor="blue"

## 171.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| color\_name | 规定颜色值为颜色名称的边框颜色（比如 red）。 |
| hex\_number | 规定颜色值为十六进制值的边框颜色（比如 #ff0000）。 |
| rgb\_number | 规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。 |
| transparent | 默认值。边框颜色为透明。 |
| inherit | 规定应该从父元素继承边框颜色。 |

# 第172章 CSS border-top-left-radius 属性

向 div 元素的左上角添加圆角边框：

div

{

border:2px solid;

border-top-left-radius:2em;

}

## 172.1 定义和用法

border-top-left-radius 属性定义左上角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderTopLeftRadius="5px"

## 172.2 语法

border-bottom-right-radius: length|% [length|%];

注释：border-top-left-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 定义左下角的形状。 |
| % | 以百分比值定义左下角的形状。 |

# 第173章 CSS border-top-right-radius 属性

向 div 元素的右上角添加圆角边框：

div

{

border:2px solid;

border-top-right-radius:2em;

}

## 173.1 定义和用法

border-top-right-radius 属性定义右下角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.borderTopRightRadius="5px"

## 173.2 语法

border-top-right-radius: length|% [length|%];

注释：border-top-right-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 定义右上角的形状。 |
| % | 以百分比值定义右上角的形状。 |

# 第174章 CSS border-top-style 属性

设置上边框的样式：

p

{

border-style:solid;

border-top-style:dotted;

}

## 174.1 定义和用法

border-top-style 设置元素上边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值： not specified

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderTopStyle="dotted"

## 174.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 定义无边框。 |
| hidden | 与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。 |
| dotted | 定义点状边框。在大多数浏览器中呈现为实线。 |
| dashed | 定义虚线。在大多数浏览器中呈现为实线。 |
| solid | 定义实线。 |
| double | 定义双线。双线的宽度等于 border-width 的值。 |
| groove | 定义 3D 凹槽边框。其效果取决于 border-color 的值。 |
| ridge | 定义 3D 垄状边框。其效果取决于 border-color 的值。 |
| inset | 定义 3D inset 边框。其效果取决于 border-color 的值。 |
| outset | 定义 3D outset 边框。其效果取决于 border-color 的值。 |
| inherit | 规定应该从父元素继承边框样式。 |

# 第175章 CSS border-top-width 属性

设置上边框的宽度：

p

{

border-style:solid;

border-top-width:15px;

}

## 175.1 定义和用法

border-top-width 属性设置元素的上边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-top-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值： medium

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderTopWidth="thick"

## 175.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| thin | 定义细的上边框。 |
| medium | 默认值。定义中等的上边框。 |
| thick | 定义粗的上边框。 |
| length | 允许您自定义上边框的宽度。 |
| inherit | 规定应该从父元素继承边框宽度。 |

# 第176章 CSS border-width 属性

设置四个边框的宽度：

p

{

border-style:solid;

border-width:15px;

}

## 176.1 定义和用法

border-width 简写属性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。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

例子 1

border-width:thin medium thick 10px;

上边框是细边框

右边框是中等边框

下边框是粗边框

左边框是 10px 宽的边框

例子 2

border-width:thin medium thick;

上边框是 10px

右边框和左边框是中等边框

下边框是粗边框

例子 3

border-width:thin medium;

上边框和下边框是细边框

右边框和左边框是中等边框

例子 4

border-width:thin;

所有 4 个边框都是细边框

默认值： medium

继承性： no

版本： CSS1

JavaScript 语法： object.style.borderWidth="thin thick"

## 176.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| thin | 定义细的边框。 |
| medium | 默认。定义中等的边框。 |
| thick | 定义粗的边框。 |
| length | 允许您自定义边框的宽度。 |
| inherit | 规定应该从父元素继承边框宽度。 |

# 第177章 CSS bottom 属性

把图像的底边缘设置在其包含元素底边缘之上 5 像素高的位置：

img

{

position:absolute;

bottom:5px;

}

## 177.1 定义和用法

bottom 属性规定元素的底部边缘。该属性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。

注释：如果 "position" 属性的值为 "static"，那么设置 "bottom" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，如果 bottom 和 top 都是 auto，其计算值则都是 0；如果其中之一为 auto，则取另一个值的相反数；如果二者都不是 auto，bottom 将取 top 值的相反数。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.bottom="50px"

177.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。通过浏览器计算底部的位置。 |
| % | 设置以包含元素的百分比计的底边位置。可使用负值。 |
| length | 使用 px、cm 等单位设置元素的底边位置。可使用负值。 |
| inherit | 规定应该从父元素继承 bottom 属性的值。 |

# 第178章 CSS box-decoration-break 属性

规定 box-decoration-break 属性：

span.ex1 {

-webkit-box-decoration-break: clone;

-o-box-decoration-break: clone;

box-decoration-break: clone;

}

span.ex2 {

-webkit-box-decoration-break: slice;

-o-box-decoration-break: slice;

box-decoration-break: slice;

}

## 178.1 定义和用法

box-decoration-break 属性规定当元素框被分段时，如何应用元素的 background、padding、border、border-image、box-shadow、margin 以及 clip-path。

默认值： slice

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

## 178.2 CSS 语法

box-decoration-break: slice|clone|initial|inherit|unset;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| slice | 默认。框装饰作为整体应用于元素，并在元素片段的边缘断裂。 |
| clone | 框装饰适用于元素的每个片段，就像片段是单个元素一样。  边框包裹元素的每个片段的四个边缘，并且完整地重绘每个片段的背景 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |
|  |  |

# 第179章 CSS box-shadow 属性

向 div 元素添加 box-shadow：

div

{

box-shadow: 10px 10px 5px #888888;

}

## 179.1 定义和用法

box-shadow 属性向框添加一个或多个阴影。

提示：请使用 border-image-\* 属性来构造漂亮的可伸缩按钮！

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.boxShadow="10px 10px 5px #888888"

## 179.2 语法

box-shadow: h-shadow v-shadow blur spread color inset;

注释：box-shadow 向框添加一个或多个阴影。该属性是由逗号分隔的阴影列表，每个阴影由 2-4 个长度值、可选的颜色值以及可选的 inset 关键词来规定。省略长度的值是 0。

|  |  |
| --- | --- |
| 值 | 描述 |
| h-shadow | 必需。水平阴影的位置。允许负值。 |
| v-shadow | 必需。垂直阴影的位置。允许负值。 |
| blur | 可选。模糊距离。 |
| spread | 可选。阴影的尺寸。 |
| color | 可选。阴影的颜色。请参阅 CSS 颜色值。 |
| inset | 可选。将外部阴影 (outset) 改为内部阴影。 |

# 第180章 CSS box-sizing 属性

规定两个并排的带边框的框

div

{

box-sizing:border-box;

-moz-box-sizing:border-box; /\* Firefox \*/

-webkit-box-sizing:border-box; /\* Safari \*/

width:50%;

float:left;

}

## 180.1 定义和用法

box-sizing 属性允许您以特定的方式定义匹配某个区域的特定元素。

例如，假如您需要并排放置两个带边框的框，可通过将 box-sizing 设置为 "border-box"。这可令浏览器呈现出带有指定宽度和高度的框，并把边框和内边距放入框中。

默认值： content-box

继承性： no

版本： CSS3

JavaScript 语法： object.style.boxSizing="border-box"

## 180.2 语法

box-sizing: content-box|border-box|inherit;

|  |  |
| --- | --- |
| 值 | 描述 |
| content-box | 这是由 CSS2.1 规定的宽度高度行为。  宽度和高度分别应用到元素的内容框。  在宽度和高度之外绘制元素的内边距和边框。 |
| border-box | 为元素设定的宽度和高度决定了元素的边框盒。  就是说，为元素指定的任何内边距和边框都将在已设定的宽度和高度内进行绘制。  通过从已设定的宽度和高度分别减去边框和内边距才能得到内容的宽度和高度。 |
| inherit | 规定应从父元素继承 box-sizing 属性的值。 |

# 第181章 CSS break-after 属性

始终在 <footer> 元素之前插入分页符：

@media print {

footer {

break-after: always;

}

}

## 181.1 定义和用法

break-after 属性规定在指定元素之后是否应发生分页（page-break）、分列（column-break）或分区（region-break）。

break-after 属性扩展了 CSS2 的 page-break-after 属性。

通过使用 break-after，您可以告知浏览器在应用 break-after 属性的元素之后，对页面、列或区域进行中断，或者避免将元素拆分并跨越两个页面。

默认值： auto

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.breakAfter="always"

## 181.2 CSS 语法

break-after: auto|all|always|avoid|avoid-column|avoid-page|avoid-region|column|left|page|recto|region|right|verso|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。在元素之后自动进行分页、分列、分区。 |
| all | 始终在主体框之后立即插入分页符。 |
| always | 始终在元素之后插入分页符。 |
| avoid | 避免在元素之后出现页、列、区域中断。 |
| avoid-column | 避免在元素之后分列。 |
| avoid-page | 避免在元素之后分页。 |
| avoid-region | 避免在元素之后分区。 |
| column | 始终在元素之后插入分列符。 |
| left | 在元素之后插入一个或两个分页符，以便将下一页格式化为左页。 |
| page | 始终在元素之后插入分页。 |
| recto | 在主体框后插入一个或两个分页符，以便将下一页格式化为 recto 页。 |
| region | 始终在元素之后插入分区符。 |
| right | 在元素之后插入一个或两个分页符，以便将下一页格式化为右页。 |
| verso | 在主体框后插入一个或两个分页符，以便将下一页格式化为 verso 页。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

实例

始终在 id 为 "toc" 的元素之后插入分页符：

@media print {

#toc {

break-after: always;

}

}

实例

始终在某个区域中的 <ul> 元素之后插入分区符：

.region ul {

break-after: region;

}

}

# 第182章 CSS break-before 属性

实例

始终在 <h1> 元素之前插入分页符：

@media print {

h1 {

break-before: always;

}

}

## 182.1 定义和用法

break-before 属性规定在指定元素之前是否应发生分页（page-break）、分列（column-break）或分区（region-break）。

break-before 属性扩展了 CSS2 的 page-break-before 属性。

通过使用 break-before，您可以告知浏览器在应用 break-after 属性的元素之前，对页面、列或区域进行中断，或者避免将元素拆分并跨越两个页面。

默认值： auto

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.breakBefore="always"

## 182.2 CSS 语法

break-before: auto|all|always|avoid|avoid-column|avoid-page|avoid-region|column|left|page|recto|region|right|verso|initial|inherit;

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。在元素之前自动进行分页、分列、分区。 |
| all | 始终在主体框之前立即插入分页符。 |
| always | 始终在元素之前插入分页符（page-break）。 |
| avoid | 避免在元素之前出现页、列、区域中断。 |
| avoid-column | 避免在元素之前分列。 |
| avoid-page | 避免在元素之前分页。 |
| avoid-region | 避免在元素之前分区。 |
| column | 始终在元素之前插入分列符。 |
| left | 在元素之前插入一个或两个分页符，以便将下一页格式化为左页。 |
| page | 始终在元素之前插入分页符。 |
| recto | 在主体框前插入一个或两个分页符，以便将下一页格式化为 recto 页。 |
| region | 始终在元素之前插入分区符。 |
| right | 在元素之前插入一个或两个分页符，以便将下一页格式化为右页。 |
| verso | 在主体框前插入一个或两个分页符，以便将下一页格式化为 verso 页。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

实例

为确保在打印时所有新章节都应从正确的页面（如书本中）开始，您可以对所有 <h1> 元素使用 break-before: ：

@media print {

h1 {

break-before: right;

}

}

实例

始终在 region 中的 <ul> 之前插入一个 region-break：

.region ul {

break-before: region;

}

}

# 第183章 CSS break-inside 属性

实例

避免在 <img> 元素内分页：

@media print {

img {

display: block;

break-inside: avoid;

}

}

## 183.1 定义和用法

break-inside 属性规定在指定元素内部是否应发生分页（page-break）、分列（column-break）或分区（region-break）。

break-inside 属性扩展了 CSS2 的 page-break-inside 属性。

通过使用 break-inside，您可以告知浏览器在图像、代码片段、表格以及列表内部避免中断。

默认值： auto

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.breakInside="always"

## 183.2 CSS 语法

break-inside: auto|all|always|avoid|avoid-column|avoid-page|avoid-region|column|left|page|recto|region|right|verso|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。在元素内自动进行分页、分列、分区。 |
| avoid | 避免在元素内出现页、列、区域中断。 |
| avoid-column | 避免在元素内分列。 |
| avoid-page | 避免在元素内分页。 |
| avoid-region | 避免在元素内分区。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

避免在 <table>、<ul>、<ol> 元素内分页：

@media print {

table, ul, ol {

break-inside: avoid;

}

}

# 第184章 CSS caption-side 属性

规定表格标题的放置方式：

caption

{

caption-side:bottom;

}

## 184.1 定义和用法

caption-side 属性设置表格标题的位置。

说明

该属性指定了表标题相对于表框的放置位置。表标题显示为好像它是表之前（或之后）的一个块级元素。

默认值： top

继承性： yes

版本： CSS2

JavaScript 语法： object.style.captionSide="bottom"

## 184.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| top | 默认值。把表格标题定位在表格之上。 |
| bottom | 把表格标题定位在表格之下。 |
| inherit | 规定应该从父元素继承 caption-side 属性的值。 |

# 第185章 CSS caret-color 属性

设置 input 元素中光标的颜色：

input {

caret-color: red;

}

## 185.1 定义和用法

caret-color 属性规定 input、textareas 或任何可编辑元素中的光标（插入符号）的颜色。

默认值： auto

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.caretColor="red"

## 185.2 CSS 语法

caret-color: auto|color;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。浏览器将 currentColor 用于插入符号。 |
| color | 规定用于插入符号的颜色。可以使用所有合法的颜色值（rgb、十六进制、命名颜色等）。 |
|  |  |

# 第186章 CSS @charset 规则

实例

把样式表的编码设置为统一码 UTF-8：

@charset "UTF-8";

## 186.1 定义和用法

@charset 规则指定样式表中使用的字符编码。

@charset 规则必须是样式表中的第一个元素，并且不能以任何字符开头。如果定义了多个 @charset 规则，则仅使用第一个。 @charset 规则不能在 style 属性内（在 HTML 元素上）或在与 HTML 页面的字符集相关的 <style> 元素内使用。

## 186.2 CSS 语法

@charset "charset";

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| charset | 规定要使用的字符编码。 |

# 第187章 CSS clear 属性

实例

图像的左侧和右侧均不允许出现浮动元素：

img

{

float:left;

clear:both;

}

## 187.1 定义和用法

clear 属性规定元素的哪一侧不允许其他浮动元素。

说明

clear 属性定义了元素的哪边上不允许出现浮动元素。在 CSS1 和 CSS2 中，这是通过自动为清除元素（即设置了 clear 属性的元素）增加上外边距实现的。在 CSS2.1 中，会在元素上外边距之上增加清除空间，而外边距本身并不改变。不论哪一种改变，最终结果都一样，如果声明为左边或右边清除，会使元素的上外边框边界刚好在该边上浮动元素的下外边距边界之下。

默认值： none

继承性： no

版本： CSS1

JavaScript 语法： object.style.clear="left"

## 187.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| left | 在左侧不允许浮动元素。 |
| right | 在右侧不允许浮动元素。 |
| both | 在左右两侧均不允许浮动元素。 |
| none | 默认值。允许浮动元素出现在两侧。 |
| inherit | 规定应该从父元素继承 clear 属性的值。 |

# 第188章 CSS clip 属性

剪裁图像：

img

{

position:absolute;

clip:rect(0px,60px,200px,0px);

}

## 188.1 定义和用法

clip 属性剪裁绝对定位元素。

当一幅图像的尺寸大于包含它的元素时会发生什么呢？"clip" 属性允许您规定一个元素的可见尺寸，这样此元素就会被修剪并显示为这个形状。

说明

这个属性用于定义一个剪裁矩形。对于一个绝对定义元素，在这个矩形内的内容才可见。出了这个剪裁区域的内容会根据 overflow 的值来处理。剪裁区域可能比元素的内容区大，也可能比内容区小。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.clip="rect(0px,50px,50px,0px)"

## 188.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| shape | 设置元素的形状。唯一合法的形状值是：rect (top, right, bottom, left) |
| auto | 默认值。不应用任何剪裁。 |
| inherit | 规定应该从父元素继承 clip 属性的值。 |

# 第189章 CSS clip-path 属性

把图像裁剪为 50% 的圆：

img {

clip-path: circle(50%);

}

## 189.1 定义和用法

clip-path 属性使您可以将元素裁剪为基本形状或 SVG 源。

注释：clip-path 将替代废弃的 clip 属性。

默认值： none

继承： 否

动画制作： 支持（仅限 basic-shape 值）。请参阅：动画相关属性。

版本： CSS Masking Module Level 1

JavaScript 语法： object.style.clipPath="circle(50%)"

## 189.2 CSS 语法

clip-path: clip-source|basic-shape|margin-box|border-box|padding-box|content-box|fill-box|stroke-box|view-box|none;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| clip-source | 定义指向 SVG <clipPath> 元素的 URL。 |
| basic-shape | 把元素剪裁为基础形状：圆、椭圆、多边形或星形。 |
| margin-box | 使用 margin box 作为引用框。 |
| border-box | 使用 border box 作为引用框。 |
| padding-box | 使用 padding box 作为引用框。 |
| content-box | 使用 content box 作为引用框。 |
| fill-box | 利用对象边界框（object bounding box）作为引用框。 |
| stroke-box | 使用笔触边界框（stroke bounding box）作为引用框。 |
| view-box | 使用 SVG 视口作为引用框。 |
| none | 默认值。不创建的剪切路径。 |

# 第190章 CSS color 属性

为不同元素设置文本颜色：

body

{

color:red;

}

h1

{

color:#00ff00;

}

p

{

color:rgb(0,0,255);

}

## 191.1 定义和用法

color 属性规定文本的颜色。

这个属性设置了一个元素的前景色（在 HTML 表现中，就是元素文本的颜色）；光栅图像不受 color 影响。这个颜色还会应用到元素的所有边框，除非被 border-color 或另外某个边框颜色属性覆盖。

要设置一个元素的前景色，最容易的方法是使用 color 属性。

默认值： not specified

继承性： yes

版本： CSS1

JavaScript 语法： object.style.color="#FF0000"

提示和注释

提示：请使用合理的背景颜色和文本颜色搭配，这样可以提高文本的可读性。

## 190.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| color\_name | 规定颜色值为颜色名称的颜色（比如 red）。 |
| hex\_number | 规定颜色值为十六进制值的颜色（比如 #ff0000）。 |
| rgb\_number | 规定颜色值为 rgb 代码的颜色（比如 rgb(255,0,0)）。 |
| inherit | 规定应该从父元素继承颜色。 |

# 第191章 CSS column-count 属性

将 div 元素中的文本分为三列：

div

{

-moz-column-count:3; /\* Firefox \*/

-webkit-column-count:3; /\* Safari 和 Chrome \*/

column-count:3;

}

## 191.1 定义和用法

column-count 属性规定元素应该被划分的列数。

默认值： auto

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnCount=3

## 191.2 语法

column-count: number|auto;

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 元素内容将被划分的最佳列数。 |
| auto | 由其他属性决定列数，比如 "column-width"。 |

# 第192章 CSS column-fill 属性

规定如何对列进行填充：

div

{

column-fill:auto;

}

## 192.1 定义和用法

column-fill 属性规定如何填充列（是否进行协调）。

默认值： balance

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnFill="auto"

## 192.1 语法

column-fill: balance|auto;

|  |  |
| --- | --- |
| 值 | 描述 |
| balance | 对列进行协调。浏览器应对列长度的差异进行最小化处理。 |
| auto | 按顺序对列进行填充，列长度会各有不同。 |

# 第193章 CSS column-gap 属性

规定列间的间隔为 40 像素：

div

{

-moz-column-gap:40px; /\* Firefox \*/

-webkit-column-gap:40px; /\* Safari 和 Chrome \*/

column-gap:40px;

}

## 193.1 定义和用法

column-gap 属性规定列之间的间隔。

注释：如果列之间设置了 column-rule，它会在间隔中间显示。

默认值： normal

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnGap="40px"

## 193.2 语法

column-gap: length|normal;

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 把列间的间隔设置为指定的长度。 |
| normal | 规定列间间隔为一个常规的间隔。W3C 建议的值是 1em。 |

# 第194章 CSS column-rule 属性

规定列之间的宽度、样式和颜色规则：

div

{

-moz-column-rule:3px outset #ff00ff; /\* Firefox \*/

-webkit-column-rule:3px outset #ff00ff; /\* Safari 和 Chrome \*/

column-rule:3px outset #ff00ff;

}

## 194.1 定义和用法

column-rule 属性是一个简写属性，用于设置所有 column-rule-\* 属性。

column-rule 属性设置列只觉得宽度、样式和颜色规则。

默认值： medium none black

继承性： no

版本： CSS3

JavaScript 语法： object.columnRule="3px outset #ff00ff"

## 194.2 语法

column-rule: column-rule-width column-rule-style column-rule-color;

|  |  |
| --- | --- |
| 值 | 描述 |
| column-rule-width | 设置列之间的宽度规则。 |
| column-rule-style | 设置列之间的样式规则。 |
| column-rule-color | 设置列之间的颜色规则。 |

# 第195章 CSS column-rule-color 属性

设置列之间的颜色规则：

div

{

-moz-column-rule-color:#ff0000; /\* Firefox \*/

-webkit-column-rule-color:#ff0000; /\* Safari 和 Chrome \*/

column-rule-color:#ff0000;

}

## 195.1 定义和用法

column-rule-color 属性规定列之间的颜色规则。

默认值： black

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnRuleColor="#ff00ff"

## 195.2 语法

column-rule-color: color;

|  |  |
| --- | --- |
| 值 | 描述 |
| color | 规定颜色规则。请参阅 CSS 颜色值。 |
|  |  |

# 第196章 CSS column-rule-style 属性

设置列之间的颜色规则：

div

{

-moz-column-rule-style:dotted; /\* Firefox \*/

-webkit-column-rule-style:dotted; /\* Safari 和 Chrome \*/

column-rule-style:dotted;

}

## 196.1 定义和用法

column-rule-style 属性规定列之间的样式规则。

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnRuleStyle="dotted"

## 196.2 语法

column-rule-style: none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 定义没有规则。 |
| hidden | 定义隐藏规则。 |
| dotted | 定义点状规则。 |
| dashed | 定义虚线规则。 |
| solid | 定义实线规则。 |
| double | 定义双线规则。 |
| groove | 定义 3D grooved 规则。该效果取决于宽度和颜色值。 |
| ridge | 定义 3D ridged 规则。该效果取决于宽度和颜色值。 |
| inset | 定义 3D inset 规则。该效果取决于宽度和颜色值。 |
| outset | 定义 3D outset 规则。该效果取决于宽度和颜色值。 |

# 第197章 CSS column-rule-width 属性

设置列之间的颜色规则：

div

{

-moz-column-rule-width:1px; /\* Firefox \*/

-webkit-column-rule-width:1px; /\* Safari 和 Chrome \*/

column-rule-width:1px;

}

## 197.1 定义和用法

column-rule-width 属性规定列之间的宽度规则。

默认值： medium

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnRuleWidth="thin"

## 197.2 语法

column-rule-width: thin|medium|thick|length;

|  |  |
| --- | --- |
| 值 | 描述 |
| thin | 定义纤细规则。 |
| medium | 定义中等规则。 |
| thick | 定义宽厚规则。 |
| length | 规定规则的宽度。 |

# 第198章 CSS column-span 属性

使 h2 元素横跨所有列：

h2

{

-webkit-column-span:all; /\* Chrome \*/

column-span:all;

}

## 198.1 定义和用法

column-span 属性规定元素应横跨多少列。

默认值： 1

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnSpan="all"

## 198.2 语法

column-span: 1|all;

|  |  |
| --- | --- |
| 值 | 描述 |
| 1 | 元素应横跨一列。 |
| all | 元素应横跨所有列。 |

# 第199章 CSS column-width 属性

规定列的宽度：

div

{

column-width:100px;

-moz-column-width:100px; /\* Firefox \*/

-webkit-column-width:100px; /\* Safari 和 Chrome \*/

}

## 199.1 定义和用法

column-width 属性规定列的宽度。

默认值： auto

继承性： no

版本： CSS3

JavaScript 语法： object.style.columnWidth="100px"

## 199.2 语法

column-width: auto|length;

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 由浏览器决定列宽。 |
| length | 规定列的宽度。 |
|  |  |

# 第200章 CSS columns 属性

规定列的宽度和列数：

div

{

columns:100px 3;

-moz-columns:100px 3; /\* Firefox \*/

-webkit-columns:100px 3; /\* Safari 和 Chrome \*/

}

## 200.1 定义和用法

columns 属性是一个简写属性，用于设置列宽和列数。

默认值： auto auto

继承性： no

版本： CSS3

JavaScript 语法： object.style.columns="100px 3"

## 200.2 语法

columns: column-width column-count;

|  |  |
| --- | --- |
| 值 | 描述 |
| column-width | 列的宽度。 |
| column-count | 列数。 |

# 第201章 CSS content 属性

下面的例子在每个链接后插入括号中的 URL：

a:after

{

content: " (" attr(href) ")";

}

## 201.1 定义和用法

content 属性与 :before 及 :after 伪元素配合使用，来插入生成内容。

说明

该属性用于定义元素之前或之后放置的生成内容。默认地，这往往是行内内容，不过该内容创建的框类型可以用属性 display 控制。

默认值： normal

继承性： no

版本： CSS2

JavaScript 语法： object.style.content="url(beep.wav)"

## 201.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none |  |
| normal |  |
| content specifications |  |
| inherit | 规定应该从父元素继承 content 属性的值 |

# 第202章 CSS counter-increment 属性

对部分和子部分进行编号（比如 "Section 1"、"1.1"、"1.2"）的方法：

body

{

counter-reset:section;

}

h1

{

counter-reset:subsection;

}

h1:before

{

content:"Section " counter(section) ". ";

counter-increment:section;

}

h2:before

{

counter-increment:subsection;

content:counter(section) "." counter(subsection) " ";

}

## 202.1 定义和用法

counter-increment 属性设置某个选取器每次出现的计数器增量。默认增量是 1。

说明

利用这个属性，计数器可以递增（或递减）某个值，这可以是正值或负值。如果没有提供 number 值，则默认为 1。

注释：如果使用了 "display: none"，则无法增加计数。如使用 "visibility: hidden"，则可增加计数。

默认值： none

继承性： no

版本： CSS2

JavaScript 语法： object.style.counterIncrement="subsection"

## 202.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。选择器无计数器增量。 |
| id number | id 定义将增加计数的选择器、id 或 class。  number 定义增量。number 可以是正数、零或者负数。 |
| inherit | 规定应该从父元素继承 counter-increment 属性的值。 |

# 第203章 CSS counter-reset 属性

对部分和子部分进行编号（比如 "Section 1"、"1.1"、"1.2"）的方法：

body

{

counter-reset:section;

}

h1

{

counter-reset:subsection;

}

h1:before

{

content:"Section " counter(section) ". ";

counter-increment:section;

}

h2:before

{

counter-increment:subsection;

content:counter(section) "." counter(subsection) " ";

}

## 203.1 定义和用法

counter-reset 属性设置某个选择器出现次数的计数器的值。默认为 0。

利用这个属性，计数器可以设置或重置为任何值，可以是正值或负值。如果没有提供 number，则默认为 0。

注释：如果使用 "display: none"，则无法重置计数器。如果使用 "visibility: hidden"，则可以重置计数器。

默认值： none

继承性： no

版本： CSS2

JavaScript 语法： object.style.counterReset="subsection"

203.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。不能对选择器的计数器进行重置。 |
| id number | id 定义重置计数器的选择器、id 或 class。  number 可设置此选择器出现次数的计数器的值。可以是正数、零或负数。 |
| inherit | 规定应该从父元素继承 counter-reset 属性的值。 |

# 第204章 CSS cursor 属性

一些不同的光标：

span.crosshair {cursor:crosshair;}

span.help {cursor:help;}

span.wait {cursor:wait;}

## 204.1 定义和用法

cursor 属性规定要显示的光标的类型（形状）。

该属性定义了鼠标指针放在一个元素边界范围内时所用的光标形状（不过 CSS2.1 没有定义由哪个边界确定这个范围）。

默认值： auto

继承性： yes

版本： CSS2

JavaScript 语法： object.style.cursor="crosshair"

## 204.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| url |  |
| 需使用的自定义光标的 URL。 |  |
|  |  |
| 注释：请在此列表的末端始终定义一种普通的光标，以防没有由 URL 定义的可用光标。 |  |
|  |  |
| default | 默认光标（通常是一个箭头） |
| auto | 默认。浏览器设置的光标。 |
| crosshair | 光标呈现为十字线。 |
| pointer | 光标呈现为指示链接的指针（一只手） |
| move | 此光标指示某对象可被移动。 |
| e-resize | 此光标指示矩形框的边缘可被向右（东）移动。 |
| ne-resize | 此光标指示矩形框的边缘可被向上及向右移动（北/东）。 |
| nw-resize | 此光标指示矩形框的边缘可被向上及向左移动（北/西）。 |
| n-resize | 此光标指示矩形框的边缘可被向上（北）移动。 |
| se-resize | 此光标指示矩形框的边缘可被向下及向右移动（南/东）。 |
| sw-resize | 此光标指示矩形框的边缘可被向下及向左移动（南/西）。 |
| s-resize | 此光标指示矩形框的边缘可被向下移动（南）。 |

# 第205章 CSS direction 属性

把文本方向设置为“从右向左”：

div

{

direction: rtl;

}

## 205.1 定义和用法

direction 属性规定文本的方向 / 书写方向。

该属性指定了块的基本书写方向，以及针对 Unicode 双向算法的嵌入和覆盖方向。不支持双向文本的用户代理可以忽略这个属性。

默认值： ltr

继承性： yes

版本： CSS2

JavaScript 语法： object.style.direction="rtl"

## 205.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| ltr | 默认。文本方向从左到右。 |
| rtl | 文本方向从右到左。 |
| inherit | 规定应该从父元素继承 direction 属性的值。 |

# 第206章 CSS display 属性

使段落生出行内框：

p.inline

{

display:inline;

}

注释：如果规定了 !DOCTYPE，则 Internet Explorer 8 （以及更高版本）支持属性值 "inline-table"、"run-in"、"table"、"table-caption"、"table-cell"、"table-column"、"table-column-group"、"table-row"、"table-row-group"、以及 "inherit"。

## 206.1 定义和用法

display 属性规定元素应该生成的框的类型。

说明

这个属性用于定义建立布局时元素生成的显示框类型。对于 HTML 等文档类型，如果使用 display 不谨慎会很危险，因为可能违反 HTML 中已经定义的显示层次结构。对于 XML，由于 XML 没有内置的这种层次结构，所有 display 是绝对必要的。

注释：CSS2 中有值 compact 和 marker，不过由于缺乏广泛的支持，已经从 CSS2.1 中去除了。

默认值： inline

继承性： no

版本： CSS1

JavaScript 语法： object.style.display="inline"

## 206.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 此元素不会被显示。 |
| block | 此元素将显示为块级元素，此元素前后会带有换行符。 |
| inline | 默认。此元素会被显示为内联元素，元素前后没有换行符。 |
| inline-block | 行内块元素。（CSS2.1 新增的值） |
| list-item | 此元素会作为列表显示。 |
| run-in | 此元素会根据上下文作为块级元素或内联元素显示。 |
| compact | CSS 中有值 compact，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。 |
| marker | CSS 中有值 marker，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。 |
| table | 此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。 |
| inline-table | 此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。 |
| table-row-group | 此元素会作为一个或多个行的分组来显示（类似 <tbody>）。 |
| table-header-group | 此元素会作为一个或多个行的分组来显示（类似 <thead>）。 |
| table-footer-group | 此元素会作为一个或多个行的分组来显示（类似 <tfoot>）。 |
| table-row | 此元素会作为一个表格行显示（类似 <tr>）。 |
| table-column-group | 此元素会作为一个或多个列的分组来显示（类似 <colgroup>）。 |
| table-column | 此元素会作为一个单元格列显示（类似 <col>） |
| table-cell | 此元素会作为一个表格单元格显示（类似 <td> 和 <th>） |
| table-caption | 此元素会作为一个表格标题显示（类似 <caption>） |
| inherit | 规定应该从父元素继承 display 属性的值。 |

# 第207章 CSS empty-cells 属性

隐藏表格中空单元格上的边框和背景：

table

{

border-collapse:separate;

empty-cells:hide;

}

## 207.1 定义和用法

empty-cells 属性设置是否显示表格中的空单元格（仅用于“分离边框”模式）。

注释：某些版本的 IE 浏览器不支持此属性。

说明

该属性定义了不包含任何内容的表单元格如何表示。如果显示，就会绘制出单元格的边框和背景。除非 border-collapse 设置为 separate，否则将忽略这个属性。

默认值： show

继承性： yes

版本： CSS2

JavaScript 语法： object.style.emptyCells="hide"

## 207.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| hide | 不在空单元格周围绘制边框。 |
| show | 在空单元格周围绘制边框。默认。 |
| inherit | 规定应该从父元素继承 empty-cells 属性的值。 |

# 第208章 CSS filter 属性

把所有图像都改为黑白（100% 灰色）：

img {

filter: grayscale(100%);

}

## 208.1 定义和用法

filter 属性定义元素（通常是 <img>）的视觉效果（如模糊和饱和度）。

默认值： none

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.filter="grayscale(100%)"

## 208.2 CSS 语法

filter: none | blur() | brightness() | contrast() | drop-shadow() | grayscale() | hue-rotate() | invert() | opacity() | saturate() | sepia() | url();

提示：如需使用多个滤镜，请用空格分隔每个滤镜（参见页面下方的更多实例）。

## 208.3 滤镜函数

注释：使用百分比值（例如 75％）的滤镜，也接受该值是十进制（例如 0.75）。

|  |  |
| --- | --- |
| 滤镜 | 描述 |
| none | 默认值。规定无效果。 |
| blur(px) | 对图像应用模糊效果。较大的值将产生更多的模糊。  如果为指定值，则使用 0。 |
| brightness(%) | 调整图像的亮度。  0％ 将使图像完全变黑。  默认值是 100％，代表原始图像。  值超过 100％ 将提供更明亮的结果。 |
| contrast(%) | 调整图像的对比度。  0％ 将使图像完全变黑。  默认值是 100％，代表原始图像。  超过 100％ 的值将提供更具对比度的结果。 |
| drop-shadow(h-shadow v-shadow blur spread color) | 对图像应用阴影效果。  可能的值：  h-shadow - 必需。指定水平阴影的像素值。负值会将阴影放置在图像的左侧。  v-shadow - 必需。指定垂直阴影的像素值。负值会将阴影放置在图像上方。  blur -可选。这是第三个值，单位必须用像素。为阴影添加模糊效果。值越大创建的模糊就越多（阴影会变得更大更亮）。不允许负值。如果未规定值，会使用 0（阴影的边缘很锐利）。  spread - 可选。这是第四个值，单位必须用像素。正值将导致阴影扩展并增大，负值将导致阴影缩小。如果未规定值，会使用 0（阴影与元素的大小相同）。  注释：Chrome、Safari 和 Opera，也许还有其他浏览器，不支持第 4 个长度；如果添加，则不会呈现。  color - 可选。为阴影添加颜色。如果未规定，则颜色取决于浏览器（通常为黑色）。  这个例子创建了红色的阴影，水平和垂直方向均为 8px，带有 10px 的模糊效果：  filter: drop-shadow(8px 8px 10px red);  提示：这个滤镜类似 box-shadow 属性。 |
| grayscale(%) | 将图像转换为灰阶。  0% (0) 是默认值，代表原始图像。  100％ 将使图像完全变灰（用于黑白图像）。  注释：不允许负值。 |
| hue-rotate(deg) | 在图像上应用色相旋转。该值定义色环的度数。默认值为 0deg，代表原始图像。  注释：最大值是 360deg。 |
| invert(%) | 反转图像中的样本。  0% (0) 是默认值，代表原始图像。  100％将使图像完全反转。  注释：不允许负值。 |
| opacity(%) | 设置图像的不透明度级别。opacity-level 描述了透明度级别，其中：  0% 为完全透明。  100% (1) 是默认值，代表原始图像（不透明）。  注释：不允许负值。  提示：这个滤镜类似 opacity 属性。 |
| saturate(%) | 设置图像的饱和度。  0% (0) will make the image completely un-saturated.  100% is default and represents the original image.  Values over 100% provides super-saturated results.  注释：不允许负值。 |
| sepia(%) | 将图像转换为棕褐色。  0% (0) 是默认值，代表原始图像。  100％ 将使图像完全变为棕褐色。  注释：不允许负值。 |
| url() | url() 函数接受规定 SVG 滤镜的 XML 文件的位置，并且可以包含指向特定滤镜元素的锚点。实例：  filter: url(svg-url#element-id) |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

模糊实例

为图像添加模糊效果：

img {

filter: blur(5px);

}

模糊实例 2

应用模糊的背景图片：

img.background {

filter: blur(35px);

}

亮度实例

调整图像的亮度：

img {

filter: brightness(200%);

}

对比度实例

调整图像的对比度：

img {

filter: contrast(200%);

}

阴影实例

为图像应用阴影效果：

img {

filter: drop-shadow(8px 8px 10px gray);

}

灰阶实例

将图像转换为灰阶：

img {

filter: grayscale(50%);

}

色相旋转实例

在图像上应用色相旋转：

img {

filter: hue-rotate(90deg);

}

反转实例

反转图像中的样本：

img {

filter: invert(100%);

}

不透明度实例

设置图像的不透明度级别：

img {

filter: opacity(30%);

}

饱和度实例

调整图像的饱和度：

img {

filter: saturate(800%);

}

棕褐色实例

将图像转换为棕褐色：

img {

filter: sepia(100%);

}

使用多重滤镜

要使用多个滤镜，请用空格分隔每个滤镜。请注意，顺序很重要（例如在 sepia() 之后使用 grayscale() 将产生完全灰色的图像）：

img {

filter: contrast(200%) brightness(150%);

}

所有滤镜

演示所有滤镜函数：

.blur {

filter: blur(4px);

}

.brightness {

filter: brightness(0.30);

}

.contrast {

filter: contrast(180%);

}

.grayscale {

filter: grayscale(100%);

}

.huerotate {

filter: hue-rotate(180deg);

}

.invert {

filter: invert(100%);

}

.opacity {

filter: opacity(50%);

}

.saturate {

filter: saturate(7);

}

.sepia {

filter: sepia(100%);

}

.shadow {

filter: drop-shadow(8px 8px 10px green);

}

# CSS flex 属性

不管其内容如何，使所有弹性项目的长度均相同：

#main div {

-ms-flex: 1; /\* IE 10 \*/

flex: 1;

}

## 209.1 定义和用法

flex 是以下属性的简写属性：

flex-grow

flex-shrink

flex-basis

flex 设置的是弹性项目的弹性长度。

注释：如果元素不是弹性项目，则 flex 属性无效。

默认值： 0 1 auto

继承： 否

动画制作： 支持。请查看单独的属性。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.flex="1"

## 209.2 CSS 语法

flex: flex-grow flex-shrink flex-basis|auto|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| flex-grow | 数字，规定项目相对于其余弹性项目的增长量。 |
| flex-shrink | 数字，规定项目相对于其余弹性项目的收缩量。 |
| flex-basis | 项目的长度。  合法值："auto"、"inherit"，或单位为 "%", "px", "em" 的值，或任何其他长度单位。 |
| auto | 等同于 1 1 auto。 |
| initial | 等同于 0 1 auto。参阅 initial。 |
| none | 等同于 0 0 auto。 |
| inherit | 从其父元素继承该属性。参阅 inherit。 |

更多实例

结合使用 flex 和媒体查询为不同的屏幕尺寸/设备创建不同的布局：

.flex-container {

display: flex;

flex-wrap: wrap;

}

.flex-item-left {

flex: 50%;

}

.flex-item-right {

flex: 50%;

}

/\* 响应式布局 - 制作单列布局（100％）而不是两列布局（50％） \*/

@media (max-width: 800px) {

.flex-item-right, .flex-item-left {

flex: 100%;

}

}

# 第210章 CSS flex-basis 属性

把第二个弹性项目的初始长度设置为 100 像素：

div:nth-of-type(2) {

flex-basis: 100px;

}

## 210.1 定义和用法

flex-basis 属性规定弹性项目的初始长度。

注释：如果元素不是弹性项目，则 flex 属性无效。

默认值： auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.flexBasis="200px"

## 210.2 CSS 语法

flex-basis: number|auto|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 长度单位或百分百，规定弹性项目的初始长度。 |
| auto | 默认值。长度等于弹性项目的长度。如果该项目未规定长度，则长度将依据其内容。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第211章 CSS flex-direction 属性

以相反的顺序设置 <div> 元素内的弹性项目的方向：

div {

display: flex;

flex-direction: row-reverse;

}

## 211.1 定义和用法

flex-direction 属性规定弹性项目的方向。

注释：如果元素不是弹性项目，则 flex 属性无效。

默认值： row

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.flexDirection="column-reverse"

## 211.2 CSS 语法

flex-direction: row|row-reverse|column|column-reverse|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| row | 默认值。作为一行，水平地显示弹性项目。 |
| row-reverse | 等同行，但方向相反。 |
| column | 作为列，垂直地显示弹性项目。 |
| column-reverse | 等同列，但方向相反。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

结合使用 flex-direction 和媒体查询为不同的屏幕尺寸/设备创建不同的布局：

.flex-container {

display: flex;

flex-direction: row;

}

/\* 响应式布局 - 制作单列布局（100％）而不是两列布局（50％） \*/

@media (max-width: 800px) {

.flex-container {

flex-direction: column;

}

}

# 第212章 CSS flex-flow 属性

使弹性项目以反方向显示，并在需要时换行：

div {

display: flex;

flex-flow: row-reverse wrap;

}

## 212.1 定义和用法

flex-flow 属性是以下属性的简写属性：

flex-direction

flex-wrap

注释：如果元素不是弹性项目，则 flex 属性无效。

默认值： row nowrap

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.flexFlow="column nowrap"

## 212.2 CSS 语法

flex-flow: flex-direction flex-wrap|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| flex-direction | 可能的值：  row  row-reverse  column  column-reverse  initial  inherit  默认值为 "row"。  规定弹性项目的方向。 |
| flex-wrap | 可能的值：  nowrap  wrap  wrap-reverse  initial  inherit  默认值为 "nowrap"。  规定弹性项目是否应换行。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第213章 CSS flex-grow 属性

让第二个弹性项目的宽度成为其余弹性项目的三倍：

div:nth-of-type(1) {flex-grow: 1;}

div:nth-of-type(2) {flex-grow: 3;}

div:nth-of-type(3) {flex-grow: 1;}

## 213.1 定义和用法

flex-grow 属性规定在相同的容器中，项目相对于其余弹性项目的增长量。

注释：如果元素不是弹性项目，则 flex 属性无效。

默认值： 0

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.flexGrow="5"

## 213.2 CSS 语法

flex-grow: number|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 数字值，规定该项目相对于其余弹性项目的增长量。默认为 0。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第214章 CSS flex-shrink 属性

实例

使第二个弹性项目相比其余项目收缩三倍：

div:nth-of-type(2) {

flex-shrink: 3;

}

## 214.1 定义和用法

flex-shrink 属性固定在相同的容器中，项目相对于其余弹性项目的收缩量。

注释：如果元素不是弹性项目，则 flex 属性无效。

默认值： 1

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.flexShrink="5"

## 214.2 CSS 语法

flex-shrink: number|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 数字，规定项目相对于其余弹性项目的收缩量。默认值为 1。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第215章 CSS flex-wrap 属性

实例

使弹性项目在需要时换行：

div {

display: flex;

flex-wrap: wrap;

}

## 215.1 定义和用法

flex-wrap 属性规定弹性项目是否应换行。

注释：如果元素不是弹性项目，则 flex 属性无效。

默认值： nowrap

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.flexWrap="nowrap"

## 215.2 CSS 语法

flex-wrap: nowrap|wrap|wrap-reverse|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| nowrap | 默认值。规定弹性项目不会换行。 |
| wrap | 规定弹性项目会在需要时换行。 |
| wrap-reverse | 规定弹性项目会在需要时换行，以反方向。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第216章 CSS float 属性

实例

把图像向右浮动：

img

{

float:right;

}

## 216.1 定义和用法

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。

默认值： none

继承性： no

版本： CSS1

JavaScript 语法： object.style.cssFloat="left"

## 216.2 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。 |

## 216.3 更多实例

**float 属性的简单应用**

<html>

<head>

<style type="text/css">

img

{

float:right

}

</style>

</head>

<body>

<p>在下面的段落中，我们添加了一个样式为 <b>float:right</b> 的图像。结果是这个图像会浮动到段落的右侧。</p>

<p>

<img src="/i/eg\_cute.gif" />

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

</body>

</html>

**将带有边框和边界的图像浮动于段落的右侧**

<html>

<head>

<style type="text/css">

img

{

float:right;

border:1px dotted black;

margin:0px 0px 15px 20px;

}

</style>

</head>

<body>

<p>在下面的段落中，图像会浮动到右侧，并且添加了点状的边框。我们还为图像添加了边距，这样就可以把文本推离图像：上和右外边距是 0px，下外边距是 15px，而图像左侧的外边距是 20px。</p>

<p>

<img src="/i/eg\_cute.gif" />

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

</body>

</html>

**带标题的图像浮动于右侧**

<html>

<head>

<style type="text/css">

div

{

float:right;

width:120px;

margin:0 0 15px 20px;

padding:15px;

border:1px solid black;

text-align:center;

}

</style>

</head>

<body>

<div>

<img src="/i/eg\_cute.gif" /><br />

CSS is fun!

</div>

<p>

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

<p>

在上面的段落中，div 元素的宽度是 120 像素，它其中包含图像。div 元素浮动到右侧。我们向 div 元素添加了外边距，这样就可以把 div 推离文本。同时，我们还向 div 添加了边框和内边距。

</p>

</body>

</html>

**使段落的首字母浮动于左侧**

<html>

<head>

<style type="text/css">

span

{

float:left;

width:0.7em;

font-size:400%;

font-family:algerian,courier;

line-height:80%;

}

</style>

</head>

<body>

<p>

<span>T</span>his is some text.

This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

</p>

<p>

在上面的段落中，文本的第一个字母包含在一个 span 元素中。这个 span 元素的宽度是当前字体尺寸的 0.7 倍。span 元素的字体尺寸是 400%，行高是 80%。span 中的字母字体是 "Algerian"

</p>

</body>

</html>

**创建水平菜单**

<html>

<head>

<style type="text/css">

ul

{

float:left;

width:100%;

padding:0;

margin:0;

list-style-type:none;

}

a

{

float:left;

width:7em;

text-decoration:none;

color:white;

background-color:purple;

padding:0.2em 0.6em;

border-right:1px solid white;

}

a:hover {background-color:#ff3300}

li {display:inline}

</style>

</head>

<body>

<ul>

<li><a href="#">Link one</a></li>

<li><a href="#">Link two</a></li>

<li><a href="#">Link three</a></li>

<li><a href="#">Link four</a></li>

</ul>

<p>

在上面的例子中，我们把 ul 元素和 a 元素浮向左浮动。li 元素显示为行内元素（元素前后没有换行）。这样就可以使列表排列成一行。ul 元素的宽度是 100%，列表中的每个超链接的宽度是 7em（当前字体尺寸的 7 倍）。我们添加了颜色和边框，以使其更漂亮。

</p>

</body>

</html>

**创建无表格的首页**

<html>

<head>

<style type="text/css">

div.container

{

width:100%;

margin:0px;

border:1px solid gray;

line-height:150%;

}

div.header,div.footer

{

padding:0.5em;

color:white;

background-color:gray;

clear:left;

}

h1.header

{

padding:0;

margin:0;

}

div.left

{

float:left;

width:160px;

margin:0;

padding:1em;

}

div.content

{

margin-left:190px;

border-left:1px solid gray;

padding:1em;

}

</style>

</head>

<body>

<div class="container">

<div class="header"><h1 class="header">W3School.com.cn</h1></div>

<div class="left"><p>"Never increase, beyond what is necessary, the number of entities required to explain anything." William of Ockham (1285-1349)</p></div>

<div class="content">

<h2>Free Web Building Tutorials</h2>

<p>At W3School.com.cn you will find all the Web-building tutorials you need,

from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.</p>

<p>W3School.com.cn - The Largest Web Developers Site On The Net!</p></div>

<div class="footer">Copyright 2008 by YingKe Investment.</div>

</div>

</body>

</html>

# 第217章 CSS font 属性

实例

在一个声明中设置所有字体属性：

p.ex1

{

font:italic arial,sans-serif;

}

p.ex2

{

font:italic bold 12px/20px arial,sans-serif;

}

## 217.1 定义和用法

font 简写属性在一个声明中设置所有字体属性。

注释：此属性也有第六个值："line-height"，可设置行间距。

说明

这个简写属性用于一次设置元素字体的两个或更多方面。使用 icon 等关键字可以适当地设置元素的字体，使之与用户计算机环境中的某个方面一致。注意，如果没有使用这些关键词，至少要指定字体大小和字体系列。

可以按顺序设置如下属性：

font-style

font-variant

font-weight

font-size/line-height

font-family

可以不设置其中的某个值，比如 font:100% verdana; 也是允许的。未设置的属性会使用其默认值。

默认值： not specified

继承性： yes

版本： CSS1

JavaScript 语法： object.style.font="italic small-caps bold 12px arial,sans-serif"

## 217.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| font-style | 规定字体样式。参阅：font-style 中可能的值。 |
| font-variant | 规定字体异体。参阅：font-variant 中可能的值。 |
| font-weight | 规定字体粗细。参阅：font-weight 中可能的值。 |
| font-size/line-height | 规定字体尺寸和行高。参阅：font-size 和 line-height 中可能的值。 |
| font-family | 规定字体系列。参阅：font-family 中可能的值。 |
| caption | 定义被标题控件（比如按钮、下拉列表等）使用的字体。 |
| icon | 定义被图标标记使用的字体。 |
| menu | 定义被下拉列表使用的字体。 |
| message-box | 定义被对话框使用的字体。 |
| small-caption | caption 字体的小型版本。 |
| status-bar | 定义被窗口状态栏使用的字体。 |

# 第218章 CSS @font-face 规则

实例

指定一种名为 "myFirstFont" 的字体，并规定可以找到它的 URL：

@font-face {

font-family: myFirstFont;

src: url(sansation\_light.woff);

}

## 218.1 定义和用法

通过使用 @font-face 规则，Web 设计师不必再使用任何一种“网络安全”字体。

在 @font-face 规则中，您必须首先定义字体名称（例如 myFirstFont），然后指向字体文件。

提示：字体的 URL 使用小写字母。大写字母可能会在 IE 中产生意外结果！

如需将字体用于 HTML 元素，请通过 font-family 属性引用字体的名称（myFirstFont）：

div {

font-family: myFirstFont;

}

## 218.2 语法

@font-face {

font-properties

}

|  |  |  |
| --- | --- | --- |
| 字体描述符 | 值 | 描述 |
| font-family | name | 必需。定义字体名称。 |
| src | URL | 必需。定义下载字体的 URL。 |
| font-stretch | normal  condensed  ultra-condensed  extra-condensed  semi-condensed  expanded  semi-expanded  extra-expanded  ultra-expanded | 可选。定义应如何拉伸字体。默认值是 "normal"。 |
| font-style | normal  italic  oblique | 可选。定义字体样式。默认值是 "normal"。 |
| font-weight | normal  bold  100  200  300  400  500  600  700  800  900 | 可选。定义字体的粗细。默认值是 "normal"。 |
| unicode-range | unicode-range | 可选。定义字体支持的 Unicode 字符范围。默认值是 "U+0-10FFFF"。 |

## 218.3 更多实例

实例

您必须添加另一个 @font-face 规则，其中包含粗体文本的描述符：

@font-face {

font-family: myFirstFont;

src: url(sansation\_bold.woff);

font-weight: bold;

}

文件 "sansation\_bold.woff" 是另一个字体文件，其中包含 Sansation 字体的粗体字符。

每当设置字体族 "myFirstFont" 的一段文本应呈现为粗体时，浏览器就会使用它。

这样，您可以为同一字体设置多个 @font-face 规则。

# 第219章 CSS font-family 属性

实例

为段落设置字体：

p

{

font-family:"Times New Roman",Georgia,Serif;

}

## 219.1 定义和用法

font-family 规定元素的字体系列。

font-family 可以把多个字体名称作为一个“回退”系统来保存。如果浏览器不支持第一个字体，则会尝试下一个。也就是说，font-family 属性的值是用于某个元素的字体族名称或/及类族名称的一个优先表。浏览器会使用它可识别的第一个值。

有两种类型的字体系列名称：

指定的系列名称：具体字体的名称，比如："times"、"courier"、"arial"。

通常字体系列名称：比如："serif"、"sans-serif"、"cursive"、"fantasy"、"monospace"

提示：使用逗号分割每个值，并始终提供一个类族名称作为最后的选择。

注意：使用某种特定的字体系列（Geneva）完全取决于用户机器上该字体系列是否可用；这个属性没有指示任何字体下载。因此，强烈推荐使用一个通用字体系列名作为后路。

默认值： not specified

继承性： yes

版本： CSS1

JavaScript 语法： object.style.fontFamily="arial,sans-serif"

## 219.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| amily-name  generic-family | 用于某个元素的字体族名称或/及类族名称的一个优先表。  默认值：取决于浏览器。 |
| inherit | 规定应该从父元素继承字体系列。 |

# 220章 CSS font-feature-settings 属性

实例

规定 font-feature-settings：

/\* 启用 small-caps \*/

.ex1 { font-feature-settings: "smcp" on; }

/\* 将大写和小写都转换为小型大写字母 \*/

.ex2 { font-feature-settings: "c2sc", "smcp"; }

/\* 没有常见的连字 \*/

.ex3 { font-feature-settings: "liga" 0; }

/\* 启用自动分数 \*/

.ex4 { font-feature-settings: "frac"; }

## 220.1 定义和用法

font-feature-settings 属性允许控制 OpenType 字体中的高级印刷特性。

默认值： normal

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.fontFeatureSettings="normal"

## 220.2 CSS 语法

font-feature-settings: normal|feature-value;

属性值

|  |  |  |
| --- | --- | --- |
| 值 | 描述 | 值 |
| normal | 默认。使用默认设置来布置文本。 | normal |
| feature-value | 格式：string [1|0|on|off]，字符串始终为 4 个 ASCII 字符。 | feature-value |

# 第221章 CSS font-kerning 属性

实例

规定应用字体字距调整：

div {

font-kerning: normal;

}

## 221.1 定义和用法

font-kerning 属性控制字体中存储的字距信息的使用。

提示：字距定义字母间距。

注释：对于不包含字距调整数据的字体，此属性将没有可见效果。

默认值： auto

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.fontKerning="normal"

## 221.2 CSS 语法

font-kerning: auto|normal|none;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。浏览器确定是否应应用字体字距调整。 |
| normal | 规定应用字体字距调整。 |
| none | 规定不应用字体字距调整。 |

# 第222章 CSS font-size 属性

实例

设置不同的 HTML 元素的尺寸：

h1 {font-size:250%;}

h2 {font-size:200%;}

p {font-size:100%;}

## 222.1 定义和用法

font-size 属性可设置字体的尺寸。

说明

该属性设置元素的字体大小。注意，实际上它设置的是字体中字符框的高度；实际的字符字形可能比这些框高或矮（通常会矮）。

各关键字对应的字体必须比一个最小关键字相应字体要高，并且要小于下一个最大关键字对应的字体。

默认值： medium

继承性： yes

版本： CSS1

JavaScript 语法： object.style.fontSize="larger"

222.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| xx-small  x-small  small  medium  large  x-large  xx-large | 把字体的尺寸设置为不同的尺寸，从 xx-small 到 xx-large。  默认值：medium。 |
| smaller | 把 font-size 设置为比父元素更小的尺寸。 |
| larger | 把 font-size 设置为比父元素更大的尺寸。 |
| length | 把 font-size 设置为一个固定的值。 |
| % | 把 font-size 设置为基于父元素的一个百分比值。 |
| inherit | 规定应该从父元素继承字体尺寸。 |

# 第223章 CSS font-size-adjust 属性

实例

设置不同的 HTML 元素的 font-size-adjust 属性：

h1

{

font-size-adjust:0.58;

}

p

{

font-size-adjust:0.60;

}

## 223.1 定义和用法

font-size-adjust 属性为某个元素规定一个 aspect 值，这样就可以保持首选字体的 x-height。

说明

字体的小写字母 "x" 的高度与 "font-size" 高度之间的比率被称为一个字体的 aspect 值。当字体拥有高的 aspect 值时，那么当此字体被设置为很小的尺寸时会更易阅读。举例：Verdana 的 aspect 值是 0.58（意味着当字体尺寸为 100px 时，它的 x-height 是 58px ）。Times New Roman 的 aspect 值是 0.46。这就意味着 Verdana 在小尺寸时比 Times New Roman 更易阅读。

默认值： none

继承性： yes

版本： CSS2

JavaScript 语法： object.style.fontSizeAdjust="0.70"

## 223.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。如果此字体不可用，则不保持此字体的 x-height。 |
| number | 定义字体的 aspect 值比率。  可使用的公式：  首选字体的字体尺寸 \* （font-size-adjust 值 / 可用字体的 aspect 值）=可应用到可用字体的字体尺寸  举例：  如果 14px 的 Verdana（aspect 值是 0.58）不可用，但是某个可用的字体的 aspect 值是 0.46，那么替代字体的尺寸将是 14 \* (0.58/0.46) = 17.65px。 |

# 第224章 CSS font-stretch 属性

实例

设置 HTML 元素的 font-stretch 属性：

h1

{

font-stretch:ultra-condensed;

}

## 224.1 定义和用法

font-stretch 属性可对当前的 font-family 进行伸缩变形。

默认值： normal

继承性： yes

版本： CSS2

JavaScript 语法： object.style.fontStretch="ultra-expanded"

## 224.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认值。把缩放比例设置为标准。 |
| wider | 把伸展比例设置为更进一步的伸展值 |
| narrower | 把收缩比例设置为更进一步的收缩值 |
| ultra-condensed  extra-condensed  condensed  semi-condensed  semi-expanded  expanded  extra-expanded  ultra-expanded | 设置 font-family 的缩放比例。  "ultra-condensed" 是最宽的值，而 "ultra-expanded" 是最窄的值。 |

# 第225章 CSS font-style 属性

实例

为三个段落设置不同的字体样式：

p.normal {font-style:normal;}

p.italic {font-style:italic;}

p.oblique {font-style:oblique;}

## 225.1 定义和用法

font-style 属性定义字体的风格。

说明

该属性设置使用斜体、倾斜或正常字体。斜体字体通常定义为字体系列中的一个单独的字体。理论上讲，用户代理可以根据正常字体计算一个斜体字体。

默认值： normal

继承性： yes

版本： CSS1

JavaScript 语法： object.style.fontStyle="italic"

## 225.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认值。浏览器显示一个标准的字体样式。 |
| italic | 浏览器会显示一个斜体的字体样式。 |
| oblique | 浏览器会显示一个倾斜的字体样式。 |
| inherit | 规定应该从父元素继承字体样式。 |

# 第226章 CSS font-variant 属性

实例

把段落设置为小型大写字母字体：

p.small

{

font-variant:small-caps;

}

## 226.1 定义和用法

font-variant 属性设置小型大写字母的字体显示文本，这意味着所有的小写字母均会被转换为大写，但是所有使用小型大写字体的字母与其余文本相比，其字体尺寸更小。

说明

该属性主要用于定义小型大写字母文本。理论上，用户代理可以根据正常字体计算出小型大写字母字体。

默认值： normal

继承性： yes

版本： CSS1

JavaScript 语法： object.style.fontVariant="small-caps"

## 226.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认值。浏览器会显示一个标准的字体。 |
| small-caps | 浏览器会显示小型大写字母的字体。 |
| inherit | 规定应该从父元素继承 font-variant 属性的值。 |

# 第227章 CSS font-variant-caps 属性

实例

将段落设置为小写字体：

p.normal {font-variant-caps: normal;}

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

p.allsmall {font-variant-caps: all-small-caps;}

p.petite {font-variant-caps: petite-caps;}

p.allpetite {font-variant-caps: all-petite-caps;}

p.unicase {font-variant-caps: unicase;}

p.titling {font-variant-caps: titling-caps;}

## 227.1 定义和用法

font-variant-caps 属性控制大写字母的替代字形的使用。

默认值： normal

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.fontVariantCaps="small-caps"

## 227.2 CSS 语法

font-variant-caps: normal|small-caps|all-small-caps|petite-caps|all-petite-caps|unicase|titling-caps|initial|inherit|unset;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 停用备用字形的使用。 |
| small-caps | 显示小型大写字母。 |
| all-small-caps | 把大写和小写字母都显示为小型大写字母。 |
| petite-caps | 显示特小型大写字母。 |
| all-petite-caps | 把大写和小写字母都显示为特小型大写字母。 |
| unicase | 允许将大写字母转化为小型大写字母与普通小写字母的混用。 |
| titling-caps | 显示首字母大写。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |
| unset |  |

# 第228章 CSS font-weight 属性

实例

设置三个段落的字体的粗细：

p.normal {font-weight:normal;}

p.thick {font-weight:bold;}

p.thicker {font-weight:900;}

## 228.1 定义和用法

font-weight 属性设置文本的粗细。

说明

该属性用于设置显示元素的文本中所用的字体加粗。数字值 400 相当于 关键字 normal，700 等价于 bold。每个数字值对应的字体加粗必须至少与下一个最小数字一样细，而且至少与下一个最大数字一样粗。

默认值： normal

继承性： yes

版本： CSS1

JavaScript 语法： object.style.fontWeight="900"

228.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认值。定义标准的字符。 |
| bold | 定义粗体字符。 |
| bolder | 定义更粗的字符。 |
| lighter | 定义更细的字符。 |
| 100  200  300  400  500  600  700  800  900 | 定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold。 |
| inherit | 规定应该从父元素继承字体的粗细。 |

# 第229章 CSS grid 属性

制作一个三列网格布局，其中第一行高 150 像素：

.grid-container {

display: grid;

grid: 150px / auto auto auto;

}

## 229.1 定义和用法

grid 属性是以下属性的简写属性：

grid-template-rows

grid-template-columns

grid-template-areas

grid-auto-rows

grid-auto-columns

grid-auto-flow

默认值： none none none auto auto row

继承： 否

动画制作： yes, see individual properties. 请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.grid="250px / auto auto auto"

## 229.2 CSS 语法

grid: none|grid-template-rows / grid-template-columns|grid-template-areas|grid-template-rows / [grid-auto-flow] grid-auto-columns|[grid-auto-flow] grid-auto-rows / grid-template-columns|initial|inherit;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认值。不定义行或列的尺寸。 |
| grid-template-rows / grid-template-columns | 规定列和行的尺寸。 |
| grid-template-areas | 规定使用命名项目的网格布局。 |
| grid-template-rows / grid-auto-columns | 规定行的尺寸（高度），以及列的自动尺寸。 |
| grid-auto-rows / grid-template-columns | 规定行的自动尺寸，并设置 grid-template-columns 属性。 |
| grid-template-rows / grid-auto-flow grid-auto-columns | 规定行的尺寸（高度），以及如何放置自动就位的项目，和列的自动尺寸。 |
| grid-auto-flow grid-auto-rows / grid-template-columns | 规定如何放置自动就位的项目，和行的自动尺寸，以及设置 grid-template-columns 属性。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

实例

规定两行，其中 "item1" 跨越前两行中的前两列（采用五列网格布局）：

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid:

'myArea myArea . . .'

'myArea myArea . . .';

}

实例

命名所有项目，并制作一张现成的网页模板：

.item1 { grid-area: header; }

.item2 { grid-area: menu; }

.item3 { grid-area: main; }

.item4 { grid-area: right; }

.item5 { grid-area: footer; }

.grid-container {

display: grid;

grid:

'header header header header header'

'menu main main main right right'

'menu footer footer footer footer';

}

# 第230章 CSS grid-area 属性

使 "item1" 在行 2 列 1 开始，并横跨两行三列：

.item1 {

grid-area: 2 / 1 / span 2 / span 3;

}

## 230.1 定义和用法

grid-area 属性网格项目的尺寸以及在网格布局中的 位置，它是以下属性的简写属性：

grid-row-start

grid-column-start

grid-row-end

grid-column-end

grid-area 属性还可以用于为网格项分配名称。然后，可以通过网格容器的 grid-template-areas 属性引用命名的网格项目。请参阅下面的例子。

默认值： auto / auto / auto / auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridArea="1 / 2 / span 2 / span 3"

## 230.2 CSS 语法

grid-area: grid-row-start / grid-column-start / grid-row-end / grid-column-end | itemname;

|  |  |
| --- | --- |
| 值 | 描述 |
| grid-row-start | 规定从哪一行开始显示项目。 |
| grid-column-start | 规定从哪一列开始显示项目。 |
| grid-row-end | 规定在哪条行线停止显示项目，或跨越多少行。 |
| grid-column-end | 指定在哪条列线停止显示项目，或跨越多少列。 |
| itemname | 规定网格项目的项目。 |

更多实例

实例

Item1 被命名 "myArea"，并在五列网格布局中横跨所有五列：

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas: 'myArea myArea myArea myArea myArea';

}

实例

使 "myArea" 在五列网格布局中横跨两列（句号表示没有名称的项目）：

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas: 'myArea myArea . . .';

}

实例

使 "item1" 横跨两列和两行：

.grid-container {

grid-template-areas: 'myArea myArea . . .' 'myArea myArea . . .';

}

实例

命名所有项目，并制作一张现成的网页模板：

.item1 { grid-area: header; }

.item2 { grid-area: menu; }

.item3 { grid-area: main; }

.item4 { grid-area: right; }

.item5 { grid-area: footer; }

.grid-container {

grid-template-areas:

'header header header header header header'

'menu main main main right right'

'menu footer footer footer footer footer';

}

# 第231章 CSS grid-auto-columns 属性

为网格中的列设置默认尺寸：

.grid-container {

display: grid;

grid-auto-columns: 50px;

}

## 231.1 定义和用法

grid-auto-columns 属性设置网格容器中列的尺寸。

该属性只会影响未设置尺寸的列。

默认值： auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridAutoColumns="120px"

231.2 CSS 语法

grid-auto-columns: auto|max-content|min-content|length;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。由容器尺寸决定列的尺寸。 |
| fit-content() |  |
| max-content | 根据列中最大的项目设置每列的尺寸。 |
| min-content | 根据列中最小的项目设置每列的尺寸。 |
| minmax(min.max) | 设置大于或等于 min 且小于或等于 max 的尺寸范围。 |
| length | 设置列的尺寸，通过使用合法的长度值。参阅长度单位。 |
| % | 设置列的尺寸，通过使用百分比值。 |

# 第232章 CSS grid-auto-flow 属性

逐列插入自动放置的项目：

.grid-container {

display: grid;

grid-auto-flow: column;

}

## 232.1 定义和用法

grid-auto-flow 属性控制自动放置的项目如何插入网格中。

默认值： row

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridAutoFlow="row dense"

## 232.2 CSS 语法

grid-auto-flow: row|column|dense|row dense|column dense;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| row | 默认值。通过填充每一行来放置项目。 |
| column | 通过填充每一列来放置项目。 |
| dense | 放置项目以填充网格中的任何孔。 |
| row dense | 通过填充每一行来放置项目，并填充网格中的任何孔。 |
| column dense | 通过填充每一列来放置项目，并填充网格中的任何孔。 |

更多实例

通过添加 "dense" 值来填充网格中的所有孔：

.grid-container {

display: grid;

grid-auto-flow: row dense;

}

# 第233章 CSS grid-auto-rows 属性

设置网格中行的默认尺寸：

.grid-container {

display: grid;

grid-auto-rows: 150px;

}

## 233.1 定义和用法

grid-auto-rows 属性为网格容器中的行设置尺寸。

该属性只会影响未设置尺寸的行。

默认值： auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridAutoRows="60px"

## 233.2 CSS 语法

grid-auto-rows: auto|max-content|min-content|length;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。由行中的最大项目的尺寸决定行的尺寸。 |
| max-content | 根据行中的最大项目设置每行的尺寸。 |
| min-content | 根据行中的最大项目设置每行的尺寸。 |
| length | 设置行的尺寸，通过使用合法的长度值。参阅长度单位。 |

# 第234章 CSS grid-column 属性

实例

使 "item1" 在列 1 开始并横跨两列：

.item1 {

grid-column: 1 / span 2;

}

## 234.1 定义和用法

grid-column 属性规定网格项目的尺寸以及在网格布局中的位置，它是以下属性的简写属性：

grid-column-start

grid-column-end

默认值： auto / auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridColumn="2 / span 2"

## 234.2 CSS 语法

grid-column: grid-column-start / grid-column-end;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| grid-column-start | 规定从哪一列开始显示项目。 |
| grid-column-end | 规定在哪一条列线（column-line）上停止显示项目，或跨越多少列。 |

更多实例

您可以使用列线值代替要跨越的列数：

.item1 {

grid-column: 1 / 3;

}

# 第235章 CSS grid-column-end 属性

使 "item1" 横跨三列：

.item1 {

grid-column-end: span 3;

}

## 235.1 定义和用法

grid-column-end 属性项目将横跨多少列，或者项目会在哪条列线（column-line）上结束。

请参阅页尾的实例。

默认值： auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridColumnEnd="5"

## 235.2 CSS 语法

grid-column-end: auto|span n|column-line;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。项目将横跨一列。 |
| span n | 规定项目将横跨的列数。 |
| column-line | 规定在哪列上停止显示项目。 |

更多实例

您可以使用列线值代替要横跨的列数：

.item1 {

grid-column-end: 3;

}

# 第236章 CSS grid-column-gap 属性

将列间隙设置为 50 像素：

.grid-container {

grid-column-gap: 50px;

}

## 236.1 定义和用法

grid-column-gap 属性定义网格布局中列间隙的尺寸。

默认值： 0

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridColumnGap="50px"

## 236.2 CSS 语法

grid-column-gap: length;

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 任何合法长度值，比如像素或百分百。默认值为 0。参阅长度单位。 |

# 第237章 CSS grid-column-start 属性

使 "item1" 从列 2 开始：

.item1 {

grid-column-start: 2;

}

## 237.1 定义和用法

grid-column-start 属性定义项目将在哪条列线上开始。

默认值： auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridColumnStart="3"

## 237.2 CSS 语法

grid-column-start: auto|span n|column-line;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。项目将随着流放置。 |
| span n | 规定项目将横跨的列数。 |
| column-line | 规定从哪列开始显示项目。 |

# 第238章 CSS grid-gap 属性

将行与列之间的简写设置为 50 像素：

.grid-container {

grid-gap: 50px;

}

## 238.1 定义和用法

grid-gap 属性定义网格布局中行与列之间间隙的尺寸，它是以下属性的简写属性：

grid-row-gap

grid-column-gap

默认值： 0 0

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridGap="50px 100px"

## 238.2 CSS 语法

grid-gap: grid-row-gap grid-column-gap;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| grid-row-gap | 设置网格布局中行间隙的尺寸。默认值为 0。 |
| grid-column-gap | 设置列间隙的尺寸。默认值为 0。 |

更多实例

设置行间隙为 20 像素，列间隙为 50 像素：

.grid-container {

grid-gap: 20px 50px;

}

# 第239章 CSS grid-row 属性

使 "item1" 在行 1 开始并横跨两行：

.item1 {

grid-row: 1 / span 2;

}

## 239.1 定义和用法

grid-row 属性规定网格项目的尺寸以及在网格布局中的位置，它是以下属性的简写属性：

grid-row-start

grid-row-end

默认值： auto / auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridRow="2 / span 2"

## 239.2 CSS 语法

grid-row: grid-row-start / grid-row-end;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| grid-row-start | 规定从哪行开始显示项目。 |
| grid-row-end | 规定在哪条行线上停止显示项目，或者横跨多少行。 |

更多实例

您可以使用行线值代替要横跨的行数：

.item1 {

grid-row: 1 / 3;

}

# 第240章 CSS grid-row-end 属性

使 "item1" 横跨三行：

.item1 {

grid-row-end: span 3;

}

## 240.1 定义和用法

grid-row-end 属性规定项目将横跨多少行，或者项目将在在哪条行线上结束。

请参见页尾的例子。

默认值： auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridRowEnd="4"

## 240.2 CSS 语法

grid-row-end: auto|row-line|span n;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。项目将横跨一行。 |
| span n | 规定项目将横跨的行数。 |
| column-line | 规定从哪行结束显示项目。 |

更多实例

您可以使用行线值代替要横跨的行数：

.item1 {

grid-row-end: 3;

}

# 第241章 CSS grid-row-gap 属性

将行间隙设置为 50 像素：

.grid-container {

grid-row-gap: 50px;

}

## 241.1 定义和用法

grid-row-gap 属性定义网格布局中行间隙的尺寸。

默认值： 0

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridRowGap="50px"

## 241.2 CSS 语法

grid-row-gap: length;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 任何合法的长度值。比如像素或百分百。默认值为 0。 |

# 第242章 CSS grid-row-start 属性

使 "item1" 从行 2 开始：

.item1 {

grid-row-start: 2;

}

## 242.1 定义和用法

grid-row-start 属性定义项目将在哪条行线上开始。

默认值： auto

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridRowStart="3"

## 242.2 CSS 语法

grid-row-start: auto|row-line;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。项目将随着流定位。 |
| row-line | 规定从哪行开始显示项目。 |

# 第243章 CSS grid-template 属性

制作第一列 150 像素高的三列网格布局：

.grid-container {

display: grid;

grid-template: 150px / auto auto auto;

}

## 243.1 定义和用法

grid-template 属性是以下属性的简写属性：

grid-template-rows

grid-template-columns

grid-template-areas

默认值： none none none

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridTemplate="250px / auto auto"

## 243.2 CSS 语法

grid-template: none|grid-template-rows / grid-template-columns|grid-template-areas|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认值。不规定列或行的尺寸。 |
| grid-template-rows / grid-template-columns | 规定列和行的尺寸。 |
| grid-template-areas | 规定使用命名项目的网格布局。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

实例

规定两行，其中 "item1" 横跨前两行中的前两列（在一个五列网格布局中）：

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template:

'myArea myArea . . .'

'myArea myArea . . .';

}

实例

命名所有项目，然后制作一张现成的网页模板：

.item1 { grid-area: header; }

.item2 { grid-area: menu; }

.item3 { grid-area: main; }

.item4 { grid-area: right; }

.item5 { grid-area: footer; }

.grid-container {

display: grid;

grid-template:

'header header header header header header'

'menu main main main right right'

'menu footer footer footer footer';

}

# 第244章 CSS grid-template-areas 属性

使命名项目 "myArea" 在五列网格布局中横跨两列：

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas: "myArea myArea . . .";

}

## 244.1 定义和用法

grid-template-areas 属性在网格布局中规定区域。

您可以使用 grid-area 属性命名网格项目，然后在 grid-template-areas 属性中引用该名称。

每个区域由撇号定义。请使用句号来引用没有名称的网格项目。

默认值： none

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridTemplateAreas=". . . myArea myArea"

## 244.2 CSS 语法

grid-template-areas: none|itemnames;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认值。未命名网格区域（grid areas）。 |
| itemnames | 规定每列和每行应如何显示的序列。 |

更多实例

实例

规定两行，其中 "item1" 在前两行中横跨前两列（在五列网格布局中）：

.item1 {

grid-area: myArea;

}

.grid-container {

display: grid;

grid-template-areas:

'myArea myArea . . .'

'myArea myArea . . .';

}

实例

命名所有项目，并制作一张现成网页模板：

.item1 { grid-area: header; }

.item2 { grid-area: menu; }

.item3 { grid-area: main; }

.item4 { grid-area: right; }

.item5 { grid-area: footer; }

.grid-container {

display: grid;

grid-template-areas:

'header header header header header'

'menu main main main right right'

'menu footer footer footer footer';

}

# 第245章 CSS grid-template-columns 属性

制作四列的网格容器：

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto;

}

## 245.1 定义和用法

grid-template-columns 属性规定网格布局中的列数（和宽度）。

这些值是一个用空格分隔的列表，其中每个值指定相应列的尺寸。

默认值： none

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridTemplateColumns="50px 50px 50px"

## 245.2 CSS 语法

grid-template-columns: none|auto|max-content|min-content|length|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认值。在需要时创建列。 |
| auto | 列的尺寸取决于容器的大小以及列中项目内容的大小。 |
| max-content | 根据列中最大的项目设置每列的尺寸。 |
| min-content | 根据列中最小的项目设置每列的尺寸。 |
| length | 设置列的尺寸，通过使用合法的长度值。参阅长度单位。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

实例

制作一个四列的网格布局，并规定每个列的尺寸：

.grid-container {

display: grid;

grid-template-columns: 30px 200px auto 100px;

}

# 第246章 CSS grid-template-rows 属性

规定 row-size（高度）：

.grid-container {

display: grid;

grid-template-rows: 100px 300px;

}

## 246.1 定义和用法

grid-template-rows 规定网格布局中的行数（和高度）。

值是用空格分隔的列表，其中每个值指定相应行的高度。

默认值： none

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS Grid Layout Module Level 1

JavaScript 语法： object.style.gridTemplateRows="50px 200px"

## 246.2 CSS 语法

grid-template-rows: none|auto|max-content|min-content|length|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 未设置尺寸。在需要时创建行。 |
| auto | 行的尺寸取决于容器的大小以及行中项目内容的大小。 |
| max-content | 根据行中的最大项目设置每行的尺寸。 |
| min-content | 根据行中的最小项目设置每行的尺寸。 |
| length | 设置行的尺寸，通过使用合法的长度值。参阅长度单位。 |

# CSS hanging-punctuation 属性

在 p 元素首行的开始边缘之外放置一个标点符号：

p

{

hanging-punctuation:first;

}

## 246.1 定义和用法

hanging-punctuation 属性规定把标点符号放在文本整行的开头还是结尾的行框之外。

默认值： none

继承性： yes

版本： CSS3

JavaScript 语法： object.style.hangingPunctuation="first"

## 246.2 语法

hanging-punctuation: none|first|last|allow-end|force-end;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 不在文本整行的开头还是结尾的行框之外放置标签符号。 |
| first | 标点附着在首行开始边缘之外。 |
| last | 标点附着在首行结尾边缘之外。 |
| allow-end |  |
| force-end |  |

# 第247章 CSS height 属性

设置段落的高度和宽度：

p

{

height:100px;

width:100px;

}

## 247.1 定义和用法

height 属性设置元素的高度。

说明

这个属性定义元素内容区的高度，在内容区外面可以增加内边距、边框和外边距。

行内非替换元素会忽略这个属性。

默认值： auto

继承性： no

版本： CSS1

JavaScript 语法： object.style.height="50px"

## 247.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。浏览器会计算出实际的高度。 |
| length | 使用 px、cm 等单位定义高度。 |
| % | 基于包含它的块级对象的百分比高度。 |
| inherit | 规定应该从父元素继承 height 属性的值 |

# 第248章 CSS hyphens 属性

设置不同的连字符：

div.a {

-webkit-hyphens: none;

-ms-hyphens: none;

hyphens: none;

}

div.b {

-webkit-hyphens: manual;

-ms-hyphens: manual;

hyphens: manual;

}

div.c {

-webkit-hyphens: auto;

-ms-hyphens: auto;

hyphens: auto;

}

## 248.1 定义和用法

hyphens 属性定义是否允许在一行文本中使用连字符创建更多的自动换行机会。

默认值： manual

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.hyphens="none"

## 248.2 CSS 语法

hyphens: none|manual|auto|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 单词不用连字符（不换行）。 |
| manual | 默认。单词只在 &hyphen; 或 &shy; 处有连字符（如果需要）。 |
| auto | 在算法确定的位置插入单词连字符（如果需要）。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第249章 CSS @import 规则

实例

把 "navigation.css" 样式表导入当前样式表中：

@import "navigation.css"; /\* 使用字符串 \*/

或者

@import url("navigation.css"); /\* 使用 url \*/

可以在页面下方找到更多 TIY 实例。

## 249.1 定义和用法

@import 规则允许您将样式表导入另一张样式表中。

@import 规则必须位于文档顶部（但是在任何 @charset 声明之后）。

@import 规则还支持媒体查询，因此可以允许依赖媒体的导入。

## 249.2 CSS 语法

@import url|string list-of-mediaqueries;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| url|string | url 或字符串，代表要导入的资源的位置。url 可以是绝对或相对的。 |
| list-of-mediaqueries | 以逗号分隔的媒体查询列表，决定通过 URL 引入的 CSS 规则在什么条件下应用。 |
|  |  |

更多实例

实例

只在媒体为 print 时导入 "printstyle.css" 样式表：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: translate(50px,100px); /\* IE 9 \*/

transform: translate(50px,100px); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>translate() 方法</h1>

<p>translate() 方法从元素当前位置对其进行移动：</p>

<div>

该 div 元素从其当前位置向右移动 50 个像素，并向下移动 100 个像素。

</div>

</body>

</html>

实例

只在媒体为 screen 且视口最大宽度 768 像素时导入 "mobstyle.css" 样式表：

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: translate(50px,100px); /\* IE 9 \*/

transform: translate(50px,100px); /\* 标准语法 \*/

}

</style>

</head>

<body>

<h1>translate() 方法</h1>

<p>translate() 方法从元素当前位置对其进行移动：</p>

<div>

该 div 元素从其当前位置向右移动 50 个像素，并向下移动 100 个像素。

</div>

</body>

</html>

# 第250章 CSS isolation 属性

为 id="e" 的元素创建新的堆叠上下文：

#e {

isolation: isolate;

}

## 250.1 定义和用法

isolation 属性定义元素是否必须创建新的堆叠内容。

提示：与 background-blend-mode 或 mix-blend-mode 一起使用时，isolation 属性很有用。

默认值： auto

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.isolation="isolate"

## 250.2 CSS 语法

isolation: auto|isolate|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。只在应用于元素的属性之一需要它时，才创建新的堆叠上下文。 |
| isolate | 必须创建新的堆叠上下文。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第251章 CSS justify-content 属性

在容器中央对齐弹性项目：

div {

display: flex;

justify-content: center;

}

可以在页面下方找到更多 TIY 实例。

## 251.1 定义和用法

justify-content 属性（水平）对齐弹性容器的项目，当项目不占用主轴上所有可用空间时。

提示：请使用 align-items 属性垂直对齐项目。

默认值： flex-start

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.justifyContent="space-between"

## 251.2 CSS 语法

justify-content: flex-start|flex-end|center|space-between|space-around|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| flex-start | 默认值。项目位于容器的开头。 |
| flex-end | 项目位于容器的结尾。 |
| center | 项目位于容器中央。 |
| space-between | 项目在行与行之间留有间隔。 |
| space-around | 项目在行之前、行之间和行之后留有空间。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

实例

在容器的开头对齐弹性项目（默认）：

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 400px;

height: 100px;

border: 1px solid #c3c3c3;

display: flex;

justify-content: flex-end;

}

#main div {

width: 70px;

height: 70px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>The "justify-content: flex-end;" 在容器末尾对齐弹性项目：</p>

<div id="main">

<div style="background-color:coral;">1</div>

<div style="background-color:lightblue;">2</div>

<div style="background-color:pink;">3</div>

</div>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 justify-content 属性。</p>

</body>

</html>

实例

在容器末尾对齐弹性项目：

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 400px;

height: 100px;

border: 1px solid #c3c3c3;

display: flex;

justify-content: space-between;

}

#main div {

width: 70px;

height: 70px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>The "justify-content: space-between;" 在行之间为弹性项目留出空间：</p>

<div id="main">

<div style="background-color:coral;">1</div>

<div style="background-color:lightblue;">2</div>

<div style="background-color:pink;">3</div>

</div>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 justify-content 属性。</p>

</body>

</html>

实例

在行之间显示带有间隔的弹性项目：

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 400px;

height: 100px;

border: 1px solid #c3c3c3;

display: flex;

justify-content: flex-start;

}

#main div {

width: 70px;

height: 70px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>The "justify-content: flex-start;" 在容器开头对齐弹性项目（默认）：</p>

<div id="main">

<div style="background-color:coral;">1</div>

<div style="background-color:lightblue;">2</div>

<div style="background-color:pink;">3</div>

</div>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 justify-content 属性。</p>

</body>

</html>

实例

在行之前、行之间和行之后显示带有间隔的弹性项目：

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 400px;

height: 100px;

border: 1px solid #c3c3c3;

display: flex;

justify-content: space-around;

}

#main div {

width: 70px;

height: 70px;

}

</style>

</head>

<body>

<h1>justify-content 属性</h1>

<p>"justify-content: space-around;" 在行之前、行之间和行之后为弹性项目留出空间：</p>

<div id="main">

<div style="background-color:coral;">1</div>

<div style="background-color:lightblue;">2</div>

<div style="background-color:pink;">3</div>

</div>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 justify-content 属性。</p>

</body>

</html>

# 第252章 CSS @keyframes 规则

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s infinite;

-moz-animation:mymove 5s infinite; /\* Firefox \*/

-webkit-animation:mymove 5s infinite; /\* Safari and Chrome \*/

-o-animation:mymove 5s infinite; /\* Opera \*/

}

@keyframes mymove

{

from {top:0px;}

to {top:200px;}

}

@-moz-keyframes mymove /\* Firefox \*/

{

from {top:0px;}

to {top:200px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

from {top:0px;}

to {top:200px;}

}

@-o-keyframes mymove /\* Opera \*/

{

from {top:0px;}

to {top:200px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

<div></div>

</body>

</html>

## 252.1 定义和用法

通过 @keyframes 规则，您能够创建动画。

创建动画的原理是，将一套 CSS 样式逐渐变化为另一套样式。

在动画过程中，您能够多次改变这套 CSS 样式。

以百分比来规定改变发生的时间，或者通过关键词 "from" 和 "to"，等价于 0% 和 100%。

0% 是动画的开始时间，100% 动画的结束时间。

为了获得最佳的浏览器支持，您应该始终定义 0% 和 100% 选择器。

注释：请使用动画属性来控制动画的外观，同时将动画与选择器绑定。

## 252.2 语法

@keyframes animationname {keyframes-selector {css-styles;}}

|  |  |
| --- | --- |
| 值 | 描述 |
| animationname | 必需。定义动画的名称。 |
| keyframes-selector | 必需。动画时长的百分比。  合法的值：  0-100%  from（与 0% 相同）  to（与 100% 相同） |
| css-styles | 必需。一个或多个合法的 CSS 样式属性。 |

实例 1

在一个动画中添加多个 keyframe 选择器：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s infinite;

-moz-animation:mymove 5s infinite; /\* Firefox \*/

-webkit-animation:mymove 5s infinite; /\* Safari and Chrome \*/

-o-animation:mymove 5s infinite; /\* Opera \*/

}

@keyframes mymove

{

0% {top:0px;}

25% {top:200px;}

75% {top:50px}

100% {top:100px;}

}

@-moz-keyframes mymove /\* Firefox \*/

{

0% {top:0px;}

25% {top:200px;}

75% {top:50px}

100% {top:100px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

0% {top:0px;}

25% {top:200px;}

75% {top:50px}

100% {top:100px;}

}

@-o-keyframes mymove /\* Opera \*/

{

0% {top:0px;}

25% {top:200px;}

75% {top:50px}

100% {top:100px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

<div></div>

</body>

</html>

在一个动画中改变多个 CSS 样式：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s infinite;

-moz-animation:mymove 5s infinite; /\* Firefox \*/

-webkit-animation:mymove 5s infinite; /\* Safari and Chrome \*/

-o-animation:mymove 5s infinite; /\* Opera \*/

}

@keyframes mymove

{

0% {top:0px; background:red; width:100px;}

100% {top:200px; background:yellow; width:300px;}

}

@-moz-keyframes mymove /\* Firefox \*/

{

0% {top:0px; background:red; width:100px;}

100% {top:200px; background:yellow; width:300px;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

0% {top:0px; background:red; width:100px;}

100% {top:200px; background:yellow; width:300px;}

}

@-o-keyframes mymove /\* Opera \*/

{

0% {top:0px; background:red; width:100px;}

100% {top:200px; background:yellow; width:300px;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

<div></div>

</body>

</html>

带有多个 CSS 样式的多个 keyframe 选择器：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:red;

position:relative;

animation:mymove 5s infinite;

-moz-animation:mymove 5s infinite; /\* Firefox \*/

-webkit-animation:mymove 5s infinite; /\* Safari and Chrome \*/

-o-animation:mymove 5s infinite; /\* Opera \*/

}

@keyframes mymove

{

0% {top:0px; left:0px; background:red;}

25% {top:0px; left:100px; background:blue;}

50% {top:100px; left:100px; background:yellow;}

75% {top:100px; left:0px; background:green;}

100% {top:0px; left:0px; background:red;}

}

@-moz-keyframes mymove /\* Firefox \*/

{

0% {top:0px; left:0px; background:red;}

25% {top:0px; left:100px; background:blue;}

50% {top:100px; left:100px; background:yellow;}

75% {top:100px; left:0px; background:green;}

100% {top:0px; left:0px; background:red;}

}

@-webkit-keyframes mymove /\* Safari and Chrome \*/

{

0% {top:0px; left:0px; background:red;}

25% {top:0px; left:100px; background:blue;}

50% {top:100px; left:100px; background:yellow;}

75% {top:100px; left:0px; background:green;}

100% {top:0px; left:0px; background:red;}

}

@-o-keyframes mymove /\* Opera \*/

{

0% {top:0px; left:0px; background:red;}

25% {top:0px; left:100px; background:blue;}

50% {top:100px; left:100px; background:yellow;}

75% {top:100px; left:0px; background:green;}

100% {top:0px; left:0px; background:red;}

}

</style>

</head>

<body>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

<div></div>

</body>

</html>

# 第253章 CSS left 属性

把图像的左边缘设置在其包含元素左边缘向右 100 像素的位置：

<html>

<head>

<style type="text/css">

img

{

position:absolute;

left:100px;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<img src="/i/w3school\_logo\_white.gif" />

</body>

</html>

## 253.1 定义和用法

left 属性规定元素的左边缘。该属性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。

注释：如果 "position" 属性的值为 "static"，那么设置 "left" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，left 的计算值始终等于 right。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.left="100px"

## 253.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。通过浏览器计算左边缘的位置。 |
| % | 设置以包含元素的百分比计的左边位置。可使用负值。 |
| length | 使用 px、cm 等单位设置元素的左边位置。可使用负值。 |
| inherit | 规定应该从父元素继承 left 属性的值。 |

# 第254章 CSS letter-spacing 属性

设置 h1 和 h2 元素的字母间距：

<html>

<head>

<style type="text/css">

h1 {letter-spacing: -0.5em}

h4 {letter-spacing: 20px}

</style>

</head>

<body>

<h1>This is header 1</h1>

<h4>This is header 4</h4>

</body>

</html>

## 254.1 定义和用法

letter-spacing 属性增加或减少字符间的空白（字符间距）。

该属性定义了在文本字符框之间插入多少空间。由于字符字形通常比其字符框要窄，指定长度值时，会调整字母之间通常的间隔。因此，normal 就相当于值为 0。

注释：允许使用负值，这会让字母之间挤得更紧。

默认值： normal

继承性： yes

版本： CSS1

JavaScript 语法： object.style.letterSpacing="3px"

## 254.1 可能的值

|  |  |
| --- | --- |
| 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。 |

# 第255章 CSS line-height 属性

<html>

<head>

<style type="text/css">

p.small {line-height: 90%}

p.big {line-height: 200%}

</style>

</head>

<body>

<p>

这是拥有标准行高的段落。

在大多数浏览器中默认行高大约是 110% 到 120%。

这是拥有标准行高的段落。

这是拥有标准行高的段落。

这是拥有标准行高的段落。

这是拥有标准行高的段落。

这是拥有标准行高的段落。

</p>

<p class="small">

这个段落拥有更小的行高。

这个段落拥有更小的行高。

这个段落拥有更小的行高。

这个段落拥有更小的行高。

这个段落拥有更小的行高。

这个段落拥有更小的行高。

这个段落拥有更小的行高。

</p>

<p class="big">

这个段落拥有更大的行高。

这个段落拥有更大的行高。

这个段落拥有更大的行高。

这个段落拥有更大的行高。

这个段落拥有更大的行高。

这个段落拥有更大的行高。

这个段落拥有更大的行高。

</p>

</body>

</html>

## 255.1 定义和用法

line-height 属性设置行间的距离（行高）。

注释：不允许使用负值。

说明

该属性会影响行框的布局。在应用到一个块级元素时，它定义了该元素中基线之间的最小距离而不是最大距离。

line-height 与 font-size 的计算值之差（在 CSS 中成为“行间距”）分为两半，分别加到一个文本行内容的顶部和底部。可以包含这些内容的最小框就是行框。

原始数字值指定了一个缩放因子，后代元素会继承这个缩放因子而不是计算值。

默认值： normal

继承性： yes

版本： CSS1

JavaScript 语法： object.style.lineHeight="2"

## 255.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认。设置合理的行间距。 |
| number | 设置数字，此数字会与当前的字体尺寸相乘来设置行间距。 |
| length | 设置固定的行间距。 |
| % | 基于当前字体尺寸的百分比行间距。 |
| inherit | 规定应该从父元素继承 line-height 属性的值。 |

# 第256章 CSS list-style 属性

<html>

<head>

<style type="text/css">

ul

{

list-style: square inside url('/i/eg\_arrow.gif')

}

</style>

</head>

<body>

<ul>

<li>咖啡</li>

<li>茶</li>

<li>可口可乐</li>

</ul>

</body>

</html>

## 256.1 定义和用法

list-style 简写属性在一个声明中设置所有的列表属性。

说明

该属性是一个简写属性，涵盖了所有其他列表样式属性。由于它应用到所有 display 为 list-item 的元素，所以在普通的 HTML 和 XHTML 中只能用于 li 元素，不过实际上它可以应用到任何元素，并由 list-item 元素继承。

可以按顺序设置如下属性：

list-style-type

list-style-position

list-style-image

可以不设置其中的某个值，比如 "list-style:circle inside;" 也是允许的。未设置的属性会使用其默认值。

默认值： disc outside none

继承性： yes

版本： CSS1

JavaScript 语法： object.style.listStyle="decimal inside"

## 256.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| list-style-type | 设置列表项标记的类型。参阅：list-style-type 中可能的值。 |
| list-style-position | 设置在何处放置列表项标记。参阅：list-style-position 中可能的值。 |
| list-style-image | 使用图像来替换列表项的标记。参阅：list-style-image 中可能的值。 |
| inherit | 规定应该从父元素继承 list-style 属性的值。 |

# 第257章 CSS list-style-image 属性

<html>

<head>

<style type="text/css">

ul

{

list-style-image: url('/i/eg\_arrow.gif')

}

</style>

</head>

<body>

<ul>

<li>咖啡</li>

<li>茶</li>

<li>可口可乐</li>

</ul>

</body>

</html>

## 257.1 定义和用法

list-style-image 属性使用图像来替换列表项的标记。

说明

这个属性指定作为一个有序或无序列表项标志的图像。图像相对于列表项内容的放置位置通常使用 list-style-position 属性控制。

注释：请始终规定一个 "list-style-type" 属性以防图像不可用。

默认值： none

继承性： yes

版本： CSS1

JavaScript 语法： object.style.listStyleImage="url('/images/blueball.gif')"

## 258.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| URL | 图像的路径。 |
| none | 默认。无图形被显示。 |
| inherit | 规定应该从父元素继承 list-style-image 属性的值。 |

# CSS list-style-position 属性

<html>

<head>

<style type="text/css">

ul.inside

{

list-style-position: inside

}

ul.outside

{

list-style-position: outside

}

</style>

</head>

<body>

<p>该列表的 list-style-position 的值是 "inside"：</p>

<ul class="inside">

<li>Earl Grey Tea - 一种黑颜色的茶</li>

<li>Jasmine Tea - 一种神奇的“全功能”茶</li>

<li>Honeybush Tea - 一种令人愉快的果味茶</li>

</ul>

<p>该列表的 list-style-position 的值是 "outside"：</p>

<ul class="outside">

<li>Earl Grey Tea - 一种黑颜色的茶</li>

<li>Jasmine Tea - 一种神奇的“全功能”茶</li>

<li>Honeybush Tea - 一种令人愉快的果味茶</li>

</ul>

</body>

</html>

## 258.1 定义和用法

list-style-position 属性设置在何处放置列表项标记。

说明

该属性用于声明列表标志相对于列表项内容的位置。外部 (outside) 标志会放在离列表项边框边界一定距离处，不过这距离在 CSS 中未定义。内部 (inside) 标志处理为好像它们是插入在列表项内容最前面的行内元素一样。

默认值： outside

继承性： yes

版本： CSS1

JavaScript 语法： object.style.listStylePosition="inside"

## 258.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| inside | 列表项目标记放置在文本以内，且环绕文本根据标记对齐。 |
| outside | 默认值。保持标记位于文本的左侧。列表项目标记放置在文本以外，且环绕文本不根据标记对齐。 |
| inherit | 规定应该从父元素继承 list-style-position 属性的值。 |

# 第259章 CSS list-style-type 属性

<html>

<head>

<style type="text/css">

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

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

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

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

</style>

</head>

<body>

<p>Type circle:</p>

<ul class="circle">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<p>Type square:</p>

<ul class="square">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<p>Type upper-roman:</p>

<ol class="upper-roman">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<p>Type lower-alpha:</p>

<ol class="lower-alpha">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

</body>

</html>

## 259.1 定义和用法

list-style-type 属性设置列表项标记的类型。

默认值： disc

继承性： yes

版本： CSS1

JavaScript 语法： object.style.listStyleType="square"

## 259.2 可能的值

CSS2 的值：

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 无标记。 |
| disc | 默认。标记是实心圆。 |
| circle | 标记是空心圆。 |
| square | 标记是实心方块。 |
| decimal | 标记是数字。 |
| decimal-leading-zero | 0开头的数字标记。(01, 02, 03, 等。) |
| lower-roman | 小写罗马数字(i, ii, iii, iv, v, 等。) |
| upper-roman | 大写罗马数字(I, II, III, IV, V, 等。) |
| lower-alpha | 小写英文字母The marker is lower-alpha (a, b, c, d, e, 等。) |
| upper-alpha | 大写英文字母The marker is upper-alpha (A, B, C, D, E, 等。) |
| lower-greek | 小写希腊字母(alpha, beta, gamma, 等。) |
| lower-latin | 小写拉丁字母(a, b, c, d, e, 等。) |
| upper-latin | 大写拉丁字母(A, B, C, D, E, 等。) |
| hebrew | 传统的希伯来编号方式 |
| armenian | 传统的亚美尼亚编号方式 |
| georgian | 传统的乔治亚编号方式(an, ban, gan, 等。) |
| cjk-ideographic | 简单的表意数字 |
| hiragana | 标记是：a, i, u, e, o, ka, ki, 等。（日文片假名） |
| katakana | 标记是：A, I, U, E, O, KA, KI, 等。（日文片假名） |
| hiragana-iroha | 标记是：i, ro, ha, ni, ho, he, to, 等。（日文片假名） |
| katakana-iroha | 标记是：I, RO, HA, NI, HO, HE, TO, 等。（日文片假名） |

**CSS2.1 的值：**

disc | circle | square | decimal | decimal-leading-zero |

lower-roman | upper-roman | lower-greek | lower-latin | upper-latin |

armenian | georgian | none | inherit

# 第260章 CSS margin 属性

<html>

<head>

<style type="text/css">

p.margin {margin: 2cm 4cm 3cm 4cm}

</style>

</head>

<body>

<p>这个段落没有指定外边距。</p>

<p class="margin">这个段落带有指定的外边距。这个段落带有指定的外边距。这个段落带有指定的外边距。这个段落带有指定的外边距。这个段落带有指定的外边距。</p>

<p>这个段落没有指定外边距。</p>

</body>

</html>

## 261.1 定义和用法

margin 简写属性在一个声明中设置所有外边距属性。该属性可以有 1 到 4 个值。

说明

这个简写属性设置一个元素所有外边距的宽度，或者设置各边上外边距的宽度。

块级元素的垂直相邻外边距会合并，而行内元素实际上不占上下外边距。行内元素的的左右外边距不会合并。同样地，浮动元素的外边距也不会合并。允许指定负的外边距值，不过使用时要小心。

注释：允许使用负值。

例子 1

margin:10px 5px 15px 20px;

上外边距是 10px

右外边距是 5px

下外边距是 15px

左外边距是 20px

例子 2

margin:10px 5px 15px;

上外边距是 10px

右外边距和左外边距是 5px

下外边距是 15px

例子 3

margin:10px 5px;

上外边距和下外边距是 10px

右外边距和左外边距是 5px

例子 4

margin:10px;

所有 4 个外边距都是 10px

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.margin="10px 5px"

## 261.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 浏览器计算外边距。 |
| length | 规定以具体单位计的外边距值，比如像素、厘米等。默认值是 0px。 |
| % | 以包含元素宽度的百分比指定外边距。 |
| inherit | 规定应该从父元素继承外边距。 |

# 第262章 CSS margin-bottom 属性

<html>

<head>

<style type="text/css">

p.bottommargin {margin-bottom: 2cm}

</style>

</head>

<body>

<p>这个段落没有指定外边距。</p>

<p class="bottommargin">这个段落带有指定的下外边距。</p>

<p>这个段落没有指定外边距。</p>

</body>

</html>

## 262.1 定义和用法

margin-bottom 属性设置元素的下外边距。

注释：允许使用负值。

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.marginBottom="10px"

## 262.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 浏览器计算下外边距。 |
| length | 规定以具体单位计的下外边距值，比如像素、厘米等。默认值是 0px。 |
| % | 以包含元素宽度的百分比指定下外边距。 |
| inherit | 规定应该从父元素继承下外边距。 |

# 第263章 CSS margin-left 属性

<html>

<head>

<style type="text/css">

p.leftmargin {margin-left: 2cm}

</style>

</head>

<body>

<p>这个段落没有指定外边距。</p>

<p class="leftmargin">这个段落带有指定的左外边距。</p>

</body>

</html>

## 263.1 定义和用法

margin-left 属性设置元素的左外边距。

注释：允许使用负值。

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.marginLeft="10px"

## 263.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 浏览器设置的左外边距。 |
| length | 定义固定的左外边距。默认值是0。 |
| % | 以包含元素宽度的百分比指定左外边距。 |
| inherit | 规定应该从父元素继承左外边距。 |

# 第264章 CSS margin-right 属性

<html>

<head>

<style type="text/css">

p.rightmargin {margin-right: 8cm}

</style>

</head>

<body>

<p>这个段落没有指定外边距。</p>

<p class="rightmargin">这个段落带有指定的右外边距。这个段落带有指定的右外边距。这个段落带有指定的右外边距。</p>

</body>

</html>

## 264.1 定义和用法

margin-right 属性设置元素的右外边距。

注释：允许使用负值。

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.marginRight="10px"

## 264.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 浏览器设置的右外边距。 |
| length | 定义固定的右外边距。默认值是 0。 |
| % | 以包含元素宽度的百分比指定右外边距。 |
| inherit | 规定应该从父元素继承右外边距。 |

# 第265章 CSS margin-top 属性

<html>

<head>

<style type="text/css">

p.topmargin {margin-top: 5cm}

</style>

</head>

<body>

<p>这个段落没有指定外边距。</p>

<p class="topmargin">这个段落带有指定的上外边距。</p>

</body>

</html>

## 265.1 定义和用法

margin-top 属性设置元素的上外边距。

注释：允许使用负值。

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.marginTop="10px"

## 265.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 浏览器设置的上外边距。 |
| length | 定义固定的上外边距。默认值是 0。 |
| % | 以包含元素宽度的百分比指定上外边距。 |
| inherit | 规定应该从父元素继承上外边距。 |

# 第266章 CSS max-height 属性

<html>

<head>

<style type="text/css">

p

{

max-height: 10px

}

</style>

</head>

<body>

<p>这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。</p>

<img src="/i/eg\_smile.gif" />

</body>

</html>

## 266.1 定义和用法

max-height 属性设置元素的最大高度。

说明

该属性值会对元素的高度设置一个最高限制。因此，元素可以比指定值矮，但不能比其高。不允许指定负值。

注释：max-height 属性不包括外边距、边框和内边距。

默认值： none

继承性： no

版本： CSS2

JavaScript 语法： object.style.maxHeight="50px"

## 266.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。定义对元素被允许的最大高度没有限制。 |
| length | 定义元素的最大高度值。 |
| % | 定义基于包含它的块级对象的百分比最大高度。 |
| inherit | 规定应该从父元素继承 max-height 属性的值。 |

# 第267章 CSS max-width 属性

<html>

<head>

<style type="text/css">

p

{

max-width: 300px

}

</style>

</head>

<body>

<p>这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。</p>

</body>

</html>

## 267.1 定义和用法

max-width 定义元素的最大宽度。

说明

该属性值会对元素的宽度设置一个最高限制。因此，元素可以比指定值窄，但不能比其宽。不允许指定负值。

默认值： none

继承性： no

版本： CSS2

JavaScript 语法： object.style.maxWidth="50px"

## 267.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。定义对元素的最大宽度没有限制。 |
| length | 定义元素的最大宽度值。 |
| % | 定义基于包含它的块级对象的百分比最大宽度。 |
| inherit | 规定应该从父元素继承 max-width 属性的值。 |

# 第268章 CSS @media 规则

如果浏览器窗口的宽度为 600px 或更小时，把 <body> 元素的背景颜色更改为“浅蓝色”：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

background-color: yellow;

}

@media only screen and (max-width: 600px) {

body {

background-color: lightblue;

}

}

</style>

</head>

<body>

<h1>@media 规则</h1>

<p>请调整浏览器窗口的大小。如果此文档的宽度为 600 像素或更小，背景颜色为“浅蓝色”，否则为“黄色”。</p>

</body>

</html>

## 268.1 定义和用法

@media 规则在媒体查询中用于为不同的媒体类型/设备应用不同的样式。

媒体查询可用于检查许多事情，诸如：

视口的宽度和高度

设备的宽度和高度

方向（手机或平板电脑处于横屏还是竖屏模式？）

分辨率

使用媒体查询是一种流行的技术，用于向台式机、笔记本电脑、平板电脑和手机提供定制的样式表（响应式网页设计）。

您还可以使用媒体查询来规定某些样式仅适用于打印的文档或屏幕阅读器（mediatype：print、screen 或 speech）。

除了媒体类型之外，还有媒体特性。媒体特性通过允许测试用户代理或显示设备的具体特性，为媒体查询提供了更多特定细节。例如，您可以将样式仅应用于大于或小于特定宽度的屏幕。

## 268.2 CSS 语法

@media not|only mediatype and (mediafeature and|or|not mediafeature) {

CSS-Code;

}

## 268.3 not、only 和 and 关键字的含义：

not：not 关键字反正整个媒体查询的含义。

only：only 关键字可防止旧版浏览器应用指定的样式，这些浏览器不支持带媒体特性的媒体查询。它对现代浏览器没有影响。

and：and 关键字将媒体特性与媒体类型或其他媒体特性组合在一起。

它们都是可选的。但是，如果使用 not 或 only，则还必须指定媒体类型。

您还可以针对不同的媒体使用不同的样式表，就像这样：

<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">

<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">

....

## 268.4 媒体类型

|  |  |
| --- | --- |
| 值 | 描述 |
| all | 默认。用于所有媒体类型设备。 |
| print | 用于打印机。 |
| screen | 用于计算机屏幕、平板电脑、智能手机等。 |
| speech | 用于朗读页面的屏幕阅读器。 |

268.5 媒体特性

|  |  |
| --- | --- |
| 值 | 描述 |
| any-hover | 是否有任何可用的输入机制允许用户（将鼠标等）悬停在元素上？  在 Media Queries Level 4 中被添加。 |
| any-pointer | 可用的输入机制中是否有任何指针设备，如果有，它的精度如何？  在 Media Queries Level 4 中被添加。 |
| aspect-ratio | 视口（viewport）的宽高比。 |
| color | 输出设备每个像素的比特值，常见的有 8、16、32 位。  如果设备不支持输出彩色，则该值为 0。 |
| color-gamut | 用户代理和输出设备大致程度上支持的色域。  在 Media Queries Level 4 中被添加。 |
| color-index | 输出设备的颜色查询表（color lookup table）中的条目数量。  如果设备不使用颜色查询表，则该值为 0 |
| device-aspect-ratio | 输出设备的宽高比。  已在 Media Queries Level 4 中被弃用。 |
| device-height | 输出设备渲染表面（如屏幕）的高度。  已在 Media Queries Level 4 中被弃用。 |
| device-width | 输出设备渲染表面（如屏幕）的宽度。  已在 Media Queries Level 4 中被弃用。 |
| display-mode | 应用程序的显示模式，如 web app 的 manifest 中的 display 成员所指定  在 Web App Manifest spec 被定义。 |
| forced-colors | 检测是用户代理否限制调色板。  在 Media Queries Level 5 中被添加。 |
| grid | 输出设备使用网格屏幕还是点阵屏幕？ |
| height | 视口（viewport）的高度。 |
| hover | 主输入机制是否允许用户将鼠标悬停在元素上？  在 Media Queries Level 4 中被添加。 |
| inverted-colors | 浏览器或者底层操作系统是否反转了颜色。  在 Media Queries Level 5 中被添加。 |
| light-level | 当前环境光水平。  在 Media Queries Level 5 中被添加。 |
| max-aspect-ratio | 显示区域的宽度和高度之间的最大比例。 |
| max-color | 输出设备每个颜色分量的最大位数。 |
| max-color-index | 设备可以显示的最大颜色数。 |
| max-height | 显示区域的最大高度，例如浏览器窗口。 |
| max-monochrome | 单色（灰度）设备上每种“颜色”的最大位数。 |
| max-resolution | 设备的最大分辨率，使用 dpi 或 dpcm。 |
| max-width | 显示区域的最大宽度，例如浏览器窗口。 |
| min-aspect-ratio | 显示区域的宽度和高度之间的最小比例。 |
| min-color | 输出设备每个颜色分量的最小位数。 |
| min-color-index | 设备可以显示的最小颜色数。 |
| min-height | 显示区域的最小高度，例如浏览器窗口。 |
| min-monochrome | 单色（灰度）设备上每种“颜色”的最小位数。 |
| min-resolution | 设备的最低分辨率，使用 dpi 或 dpcm。 |
| min-width | 显示区域的最小宽度，例如浏览器窗口。 |
| monochrome | 输出设备单色帧缓冲区中每个像素的位深度。  如果设备并非黑白屏幕，则该值为 0。 |
| orientation | 视窗（viewport）的旋转方向（横屏还是竖屏模式）。 |
| overflow-block | 输出设备如何处理沿块轴溢出视口(viewport)的内容。  在 Media Queries Level 4 中被添加。 |
| overflow-inline | 沿内联轴溢出视口(viewport)的内容是否可以滚动？  在 Media Queries Level 4 中被添加。 |
| pointer | 主要输入机制是一个指针设备吗？如果是，它的精度如何？  在 Media Queries Level 4 中被添加。 |
| prefers-color-scheme | 探测用户倾向于选择亮色还是暗色的配色方案。  在 Media Queries Level 5 中被添加。 |
| prefers-contrast | 探测用户是否有向系统要求提高或降低相近颜色之间的对比度。  在 Media Queries Level 5 中被添加。 |
| prefers-reduced-motion | 用户是否希望页面上出现更少的动态效果。  在 Media Queries Level 5 中被添加。 |
| prefers-reduced-transparency | 用户是否倾向于选择更低的透明度。  在 Media Queries Level 5 中被添加。 |
| resolution | 输出设备的分辨率，使用 dpi 或 dpcm。 |
| scan | 输出设备的扫描过程（适用于电视等）。 |
| scripting | 探测脚本（例如 JavaScript）是否可用。  在 Media Queries Level 5 中被添加。 |
| update | 输出设备更新内容的渲染结果的频率。  在 Media Queries Level 4 中被添加。 |
| width | 视窗（viewport）的宽度。 |

更多实例

实例

当浏览器的宽度为 600px 或更小时，隐藏元素：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

div.example {

background-color: yellow;

padding: 20px;

}

@media screen and (max-width: 600px) {

div.example {

display: none;

}

}

</style>

</head>

<body>

<h2>在不同的屏幕尺寸上隐藏元素</h2>

<div class="example">示例 DIV</div>

<p>请调整浏览器窗口的大小以查看响应效果。如果浏览器宽度小于或等于 600 像素，则隐藏元素。</p>

</body>

</html>

实例

如果视口的宽度为 800 像素或更宽，请使用媒体查询将背景色设置为淡紫色；如果视口的宽度介于 400 至 799 像素之间，则使用媒体查询将背景色设置为浅绿色。如果视口小于 400 像素，则背景色为浅蓝色：

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: lightblue;

}

@media screen and (min-width: 400px) {

body {

background-color: lightgreen;

}

}

@media screen and (min-width: 800px) {

body {

background-color: lavender;

}

}

</style>

</head>

<body>

<h1>请调整浏览器窗口大小以查看效果！</h1>

<p>如果视口的宽度为 800 像素或更宽，使用媒体查询将背景色设置为淡紫色；如果视口的宽度介于 400 至 799 像素之间，背景色为浅绿色。如果视口小于 400 像素，则背景色为浅蓝色。</p>

</body>

</html>

实例

创建一个响应式导航菜单（在大屏幕上水平显示，在小屏幕上垂直显示）：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 设置顶部导航栏的样式 \*/

.topnav {

overflow: hidden;

background-color: #333;

}

/\* 设置 topnav 链接的样式 \*/

.topnav a {

float: left;

display: block;

color: #f2f2f2;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

/\* 在鼠标悬停时改变颜色 \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

/\* 如果屏幕等于或小于 600 像素，则菜单链接堆叠而不是并排 \*/

@media screen and (max-width: 600px) {

.topnav a {

float: none;

width: 100%;

}

}

</style>

</head>

<body>

<h2>响应式导航菜单</h2>

<p>请调整浏览器窗口大小来查看效果：如果屏幕小于 600 像素，导航菜单将垂直而不是水平显示。</p>

<div class="topnav">

<a href="#">链接</a>

<a href="#">链接</a>

<a href="#">链接</a>

</div>

</body>

</html>

实例

使用媒体查询来创建响应式列布局：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 创建并排浮动的四列 \*/

.column {

float: left;

width: 25%;

padding: 20px;

}

/\* 在列后清除浮动 \*/

.row:after {

content: "";

display: table;

clear: both;

}

/\* 如果屏幕宽度为 992 像素或更窄，则四列变为两列 \*/

@media screen and (max-width: 992px) {

.column {

width: 50%;

}

}

/\* 如果屏幕宽度为 600 像素或更窄，则列堆叠而不是并排 \*/

@media screen and (max-width: 600px) {

.column {

width: 100%;

}

}

</style>

</head>

<body>

<h2>响应式四列布局</h2>

<p>请调整浏览器窗口的大小以查看响应效果。在宽度为 992px 或更窄的屏幕上，列会调整尺寸，将从四列变为为两列。在宽度为 600 像素或更小的屏幕上，这些列将堆叠，而不是并排。</p>

<div class="row">

<div class="column" style="background-color:#aaa;">

<h2>列 1</h2>

<p>一些文本..</p>

</div>

<div class="column" style="background-color:#bbb;">

<h2>列 2</h2>

<p>一些文本..</p>

</div>

<div class="column" style="background-color:#ccc;">

<h2>列 3</h2>

<p>一些文本..</p>

</div>

<div class="column" style="background-color:#ddd;">

<h2>列 4</h2>

<p>一些文本..</p>

</div>

</div>

</body>

</html>

实例

使用媒体查询来创建响应式网站：

<!DOCTYPE html>

<html>

<head>

<title>页面标题</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

\* {

box-sizing: border-box;

}

/\* 设置 body 的样式 \*/

body {

font-family: Arial;

margin: 0;

}

/\* 页眉/LOGO \*/

.header {

padding: 60px;

text-align: center;

background: #1abc9c;

color: white;

}

/\* 设置顶部导航栏的样式 \*/

.navbar {

display: flex;

background-color: #333;

}

/\* 设置导航栏链接的样式 \*/

.navbar a {

color: white;

padding: 14px 20px;

text-decoration: none;

text-align: center;

}

/\* 当鼠标悬停时改变颜色 \*/

.navbar a:hover {

background-color: #ddd;

color: black;

}

/\* 列容器 \*/

.row {

display: flex;

flex-wrap: wrap;

}

/\* 创建两个并排的等列 \*/

/\* Sidebar/left column \*/

.side {

flex: 30%;

background-color: #f1f1f1;

padding: 20px;

}

/\* 主列 \*/

.main {

flex: 70%;

background-color: white;

padding: 20px;

}

/\* 伪图像，仅供示例 \*/

.fakeimg {

background-color: #aaa;

width: 100%;

padding: 20px;

}

/\* 页脚 \*/

.footer {

padding: 20px;

text-align: center;

background: #ddd;

}

/\* 响应式布局 - 当屏幕宽度小于 700 像素时，使两列堆叠，而不是并排 \*/

@media (max-width: 700px) {

.row, .navbar {

flex-direction: column;

}

}

</style>

</head>

<body>

<!-- 注释 -->

<div style="background:yellow;padding:5px">

<h4 style="text-align:center">请调整浏览器窗口的大小以查看响应效果。</h4>

</div>

<!-- Header -->

<div class="header">

<h1>我的网站</h1>

<p>拥有<b>弹性</b>布局。</p>

</div>

<!-- 导航栏 -->

<div class="navbar">

<a href="#">链接</a>

<a href="#">链接</a>

<a href="#">链接</a>

<a href="#">链接</a>

</div>

<!-- 弹性网格（内容） -->

<div class="row">

<div class="side">

<h2>关于我</h2>

<h5>我的照片：</h5>

<div class="fakeimg" style="height:200px;">图像</div>

<p>hello world!</p>

<h3>更多文本</h3>

<p>Welcome to Shanghai！</p>

<div class="fakeimg" style="height:60px;">图像</div><br>

<div class="fakeimg" style="height:60px;">图像</div><br>

<div class="fakeimg" style="height:60px;">图像</div>

</div>

<div class="main">

<h2>标题</h2>

<h5>标题描述, Dec 7, 2021</h5>

<div class="fakeimg" style="height:200px;">图像</div>

<p>一些文本..</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<br>

<h2>标题</h2>

<h5>标题描述, Sep 2, 2021</h5>

<div class="fakeimg" style="height:200px;">图像</div>

<p>一些文本..</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

</div>

<!-- 页脚 -->

<div class="footer">

<h2>页脚</h2>

</div>

</body>

</html>

实例

媒体查询还可以用于根据浏览器的方向更改页面的布局。您可以写一组 CSS 属性，这些属性仅在浏览器窗口的宽度大于其高度（即“横屏”方向）时才适用。

如果方向处于横屏模式，则使用浅蓝色背景色：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

background-color: lightgreen;

}

@media only screen and (orientation: landscape) {

body {

background-color: lightblue;

}

}

</style>

</head>

<body>

<p>请调整浏览器窗口大小来查看效果：如果文档宽度大于高度，则背景色为浅蓝色，否则为浅绿色。</p>

</body>

</html>

实例

使用媒体查询将文本颜色设置为在屏幕上显示文档时为绿色，在打印时为黑色：

<!DOCTYPE html>

<html>

<head>

<style>

@media screen {

body {

color: green;

}

}

@media print {

body {

color: black;

}

}

</style>

</head>

<body>

<h1>@media 规则</h1>

<p>在屏幕上显示文档时，使用媒体查询将文本颜色设置为绿色，在打印时为黑色。</p>

</body>

</html>

实例

逗号分隔的列表：使用逗号将另一个媒体查询添加到已存在的媒体查询中（它的行为类似 OR 运算符）：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

@media screen and (max-width: 900px) and (min-width: 600px), (min-width: 1100px) {

div.example {

font-size: 50px;

padding: 50px;

border: 8px solid black;

background: yellow;

}

}

</style>

</head>

<body>

<h2>改变 DIV 在不同屏幕尺寸上的外观</h2>

<div class="example">Example DIV.</div>

<p>请调整浏览器窗口的大小以查看响应效果。如果浏览器宽度在 600 到 900 像素之间或超过 1100 像素，则改变 DIV 的外观。</p>

</body>

</html>

# 第269章 CSS min-height 属性

<html>

<head>

<style type="text/css">

p

{

min-height: 100px

}

</style>

</head>

<body>

<p>这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。</p>

<img src="/i/eg\_smile.gif" />

</body>

</html>

## 269.1 定义和用法

min-height 属性设置元素的最小高度。

说明

该属性值会对元素的高度设置一个最低限制。因此，元素可以比指定值高，但不能比其矮。不允许指定负值。

默认值： 0

继承性： no

版本： CSS2

JavaScript 语法： object.style.minHeight="50px"

## 269.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 定义元素的最小高度。默认值是 0。 |
| % | 定义基于包含它的块级对象的百分比最小高度。 |
| inherit | 规定应该从父元素继承 min-height 属性的值。 |

# 第270章 CSS min-width 属性

<html>

<head>

<style type="text/css">

p

{

min-width: 1000px

}

</style>

</head>

<body>

<p>这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。

这是一些文本。这是一些文本。这是一些文本。</p>

<img src="/i/eg\_smile.gif" />

</body>

</html>

## 270.1 定义和用法

min-width 属性设置元素的最小宽度。

说明

该属性值会对元素的宽度设置一个最小限制。因此，元素可以比指定值宽，但不能比其窄。不允许指定负值。

默认值： none

继承性： no

版本： CSS2

JavaScript 语法： object.style.minWidth="50px"

## 270.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 定义元素的最小宽度值。默认值：取决于浏览器。 |
| % | 定义基于包含它的块级对象的百分比最小宽度。 |
| inherit | 规定应该从父元素继承 min-width 属性的值。 |

# 第271章 CSS mix-blend-mode 属性

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

.container {

background-color: red;

padding: 15px;

}

.container img {

mix-blend-mode: darken;

}

</style>

</head>

<body>

<h1>mix-blend-mode 属性</h1>

<p>mix-blend-mode 属性规定元素内容如何与其背景混合：</p>

<div class="container">

<img src="/i/photo/tulip.jpg" alt="Tulip" width="300" height="300">

</div>

</body>

</html>

## 271.1 定义和用法

mix-blend-mode 属性规定元素的内容应如何与其直接父背景混合。

默认值： normal

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

JavaScript 语法： object.style.mixBlendMode = "darken"

## 271.2 CSS 语法

mix-blend-mode: normal|multiply|screen|overlay|darken|lighten|color-dodge|color-burn|difference|exclusion|hue|saturation|color|luminosity;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 这是默认值。将混合模式设置为普通。 |
| multiply | 将混合模式设置为 multiply。 |
| screen | 将混合模式设置为 screen。 |
| overlay | 将混合模式设置为 overlay。 |
| darken | 将混合模式设置为 darken。 |
| lighten | 将混合模式设置为 lighten。 |
| color-dodge | 将混合模式设置为 color-dodge。 |
| color-burn | 将混合模式设置为 color-burn。 |
| difference | 将混合模式设置为 difference。 |
| exclusion | 将混合模式设置为 exclusion。 |
| hue | 将混合模式设置为 hue。 |
| saturation | 将混合模式设置为 saturation。 |
| color | 将混合模式设置为 color。 |
| luminosity | 将混合模式设置为 luminosity。 |

更多实例

实例

演示所有值：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

.container {

background-color: red;

height: 5000px;

}

img {

width: 33.33%;

height: auto;

float: left;

}

.normal {mix-blend-mode: normal;}

.multiply {mix-blend-mode: multiply;}

.screen {mix-blend-mode: screen;}

.overlay {mix-blend-mode: overlay;}

.darken {mix-blend-mode: darken;}

.lighten {mix-blend-mode: lighten;}

.color-dodge {mix-blend-mode: color-dodge;}

.color-burn {mix-blend-mode: color-burn;}

.difference {mix-blend-mode: difference;}

.exclusion {mix-blend-mode: exclusion;}

.hue {mix-blend-mode: hue;}

.saturation {mix-blend-mode: saturation;}

.color {mix-blend-mode: color;}

.luminosity {mix-blend-mode: luminosity;}

</style>

</head>

<body>

<h1>所有 mix-blend-mode 值</h1>

<p>mix-blend-mode 属性规定元素内容如何与其背景混合：</p>

<div class="container">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="normal" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="multiply" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="screen" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="overlay" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="darken" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="lighten" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="color-dodge" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="color-burn" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="difference" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="exclusion" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="hue" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="saturation" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="color" width="300" height="300">

<img src="/i/photo/tulip.jpg" alt="Tulip" class="luminosity" width="300" height="300">

</div>

</body>

</html>

使用 mix-blend-mode 来创建响应式 cutout/knockout 文本（抠图文本）：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body {font-family: Arial, Helvetica, sans-serif;}

.image-container {

background-image: url("paris.jpg");

background-size: cover;

position: relative;

height: 300px;

}

.text {

background-color: white;

color: black;

font-size: 10vw;

font-weight: bold;

margin: 0 auto;

padding: 10px;

width: 50%;

text-align: center;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

mix-blend-mode: screen;

}

</style>

</head>

<body>

<h1>响应式文本抠图效果</h1>

<div class="image-container">

<div class="text">SHANGHAI</div>

</div>

<p>本例创建了一种响应式的抠图文本，文本会剪切背景图像。</p>

<p>请调整 浏览器窗口大小来查看响应式效果。</p>

<p><b>注释：</b>此例在 Internet Explorer 或 Edge 中无效。</p>

</body>

</html>

# 第272章 CSS object-fit 属性

裁剪图像的两边，保留长宽比，然后填充空间：

<!DOCTYPE html>

<html>

<head>

<style>

img.a {

width: 200px;

height: 400px;

object-fit: cover;

}

</style>

</head>

<body>

<h1>object-fit 属性</h1>

<h2>object-fit: cover：</h2>

<p>剪切图像的侧边，保留长宽比，并填充空间：</p>

<img class="a" src="paris.jpg" alt="Shanghai" width="400" height="300">

<h2>原始图像 ：</h2>

<img src="paris.jpg" alt="Shanghai" width="400" height="300">

<p>注释：Internet Explorer/Edge 15 或更早的版本 不支持 object-fit 属性。</p>

</body>

</html>

## 272.1 定义和用法

object-fit 属性用于规定应如何调整 <img> 或 <video> 的大小来适应其容器。

这个属性告知内容以各种方式填充容器。例如“保留长宽比”或“展开并占用尽可能多的空间”。

默认值： see individual properties

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.objectFit="cover"

## 272.2 CSS 语法

object-fit: fill|contain|cover|scale-down|none|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| fill | 默认值。调整替换内容的大小来填充元素的内容框。如有必要，对象将被拉伸或挤压。 |
| contain | 缩放被替换的内容以保持其宽高比，同时适合元素的内容框。 |
| cover | 调整被替换内容的大小，以在填充元素的整个内容框时保持其长宽比。该对象将被裁剪。 |
| none | 替换的内容不会调整大小。 |
| scale-down | 内容的尺寸与 none 或 contain 中的一个相同，取决于它们两个之间谁得到的对象尺寸会更小一些。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第273章 CSS object-position 属性

调整图像的大小以适合其内容框，并将图像在内容框内的位置从左侧移动 5px，从顶部移动 10％：

<!DOCTYPE html>

<html>

<head>

<style>

img.a {

width: 200px;

height: 400px;

object-fit: none;

object-position: 5px 10%;

border: 5px solid red;

}

</style>

</head>

<body>

<h1>object-position 属性</h1>

<p>调整图像的大小以适合其内容框，然后在内容框内将图像从左侧移 5px，从顶部移 10％：</p>

<img class="a" src="paris.jpg" alt="Shanghai" width="400" height="300">

<h2>原始图像 ：</h2>

<img src="paris.jpg" alt="Shanghai" width="400" height="300">

<p>注释：Internet Explorer/Edge 15 以及更早的版本不支持 object-position 和 object-fit 属性。</p>

</body>

</html>

## 273.1 定义和用法

object-position 属性与 object-fit 一起使用，可规定应如何在其“自己的内容框”内使用 x/y 坐标定位 <img> 或 <video>。

默认值： 50% 50%

继承： 是

动画制作： 支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.objectPosition="0 10%"

## 273.2 CSS 语法

object-position: position|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| position | 规定图像或视频在其内容框中的位置。  第一个值控制 x 轴，第二个值控制 y 轴。  可以是字符串（left、center 或 right）或数字（以 px 或 ％ 为单位）。允许负值。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第274章 CSS opacity 属性

设置 div 元素的不透明级别：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

background-color:red;

opacity:0.5;

filter:Alpha(opacity=50); /\* IE8 以及更早的浏览器 \*/

}

</style>

</head>

<body>

<div>本元素的不透明度是 0.5。请注意，文本和背景色都受到不透明级别的影响。</div>

</body>

</html>

## 274.1 定义和用法

opacity 属性设置元素的不透明级别。

默认值： 1

继承性： no

版本： CSS3

JavaScript 语法： object.style.opacity=0.5

## 274.2 语法

opacity: value|inherit;

|  |  |
| --- | --- |
| 值 | 描述 |
| value | 规定不透明度。从 0.0 （完全透明）到 1.0（完全不透明）。 |
| inherit | 应该从父元素继承 opacity 属性的值。 |

# 第275章 CSS order 属性

设置弹性项目的顺序：

<!DOCTYPE html>

<html>

<head>

<style>

#main {

width: 400px;

height: 150px;

border: 1px solid #c3c3c3;

display: -webkit-flex; /\* Safari \*/

display: flex;

}

#main div {

width: 70px;

height: 70px;

}

/\* Safari 6.1+ \*/

div#myRedDIV {-webkit-order: 2;}

div#myBlueDIV {-webkit-order: 4;}

div#myGreenDIV {-webkit-order: 3;}

div#myPinkDIV {-webkit-order: 1;}

/\* 标准语法 \*/

div#myRedDIV {order: 2;}

div#myBlueDIV {order: 4;}

div#myGreenDIV {order: 3;}

div#myPinkDIV {order: 1;}

</style>

</head>

<body>

<h1>order 属性</h1>

<div id="main">

<div style="background-color:coral;" id="myRedDIV"></div>

<div style="background-color:lightblue;" id="myBlueDIV"></div>

<div style="background-color:lightgreen;" id="myGreenDIV"></div>

<div style="background-color:pink;" id="myPinkDIV"></div>

</div>

<p><b>注释：</b>Internet Explorer 10 以及更早的版本不支持 order 属性。</p>

<p><b>注释：</b>Safari 6.1（以及更新的版本）支持替代的 -webkit-order 属性。</p>

</body>

</html>

## 275.1 定义和用法

order 属性规定弹性项目相对于同一容器内其余弹性项目的顺序。

注释：如果元素不是弹性项目，则 order 属性无效。

默认值： 0

继承： no

动画制作： 支持。请参阅单独的属性。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.order="2"

## 275.2 CSS 语法

order: number|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 默认值为 0。规定弹性项目的顺序。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第276章 CSS outline 属性

设置 4 个边框的样式：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<style type="text/css">

p

{

border:red solid thin;

outline:#00ff00 dotted thick;

}

</style>

</head>

<body>

<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高版本） 才支持 outline 属性。</p>

</body>

</html>

## 276.1 定义和用法

outline （轮廓）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。

注释：轮廓线不会占据空间，也不一定是矩形。

outline 简写属性在一个声明中设置所有的轮廓属性。

可以按顺序设置如下属性：

outline-color

outline-style

outline-width

如果不设置其中的某个值，也不会出问题，比如 outline:solid #ff0000; 也是允许的。

默认值： invert none medium

继承性： no

版本： CSS2

JavaScript 语法： object.style.outline="#0000FF dotted thin"

## 276.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| outline-color | 规定边框的颜色。参阅：outline-color 中可能的值。 |
| outline-style | 规定边框的样式。参阅：outline-style 中可能的值。 |
| outline-width | 规定边框的宽度。参阅：outline-width 中可能的值。 |
| inherit | 规定应该从父元素继承 outline 属性的设置。 |

# 第277章 CSS outline-color 属性

设置点状轮廓的颜色：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<style type="text/css">

p

{

border:red solid thin;

outline-style:dotted;

outline-color:#00ff00;

}

</style>

</head>

<body>

<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高版本） 才支持 outline-color 属性。</p>

</body>

</html>

## 277.1 定义和用法

outline （轮廓）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。outline 属性可设置元素周围的轮廓线。

注释：请始终在 outline-color 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

注释：轮廓线不会占据空间，也不一定是矩形。

outline-color 属性设置一个元素整个轮廓中可见部分的颜色。要记住，轮廓的样式不能是 none，否则轮廓不会出现。

默认值： invert

继承性： no

版本： CSS2

JavaScript 语法： object.style.outlineColor="#0000FF"

## 277.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| color\_name | 规定颜色值为颜色名称的轮廓颜色（比如 red）。 |
| hex\_number | 规定颜色值为十六进制值的轮廓颜色（比如 #ff0000）。 |
| rgb\_number | 规定颜色值为 rgb 代码的轮廓颜色（比如 rgb(255,0,0)）。 |
| invert | 默认。执行颜色反转（逆向的颜色）。可使轮廓在不同的背景颜色中都是可见。 |
| inherit | 规定应该从父元素继承轮廓颜色的设置。 |

# 第278章 CSS outline-offset 属性

规定边框边缘之外 15 像素处的轮廓：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

margin:20px;

width:150px;

padding:10px;

height:70px;

border:2px solid black;

outline:2px solid red;

outline-offset:15px;

}

</style>

</head>

<body>

<p><b>注释：</b>Internet Explorer 和 Opera 不支持 support outline-offset 属性。</p>

<div>这个 div 在边框边缘之外 15 像素处有一个轮廓。</div>

</body>

</html>

## 278.1 定义和用法

outline-offset 属性对轮廓进行偏移，并在边框边缘进行绘制。

轮廓在两方面与边框不同：

轮廓不占用空间

轮廓可能是非矩形

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.outlineOffset="15px"

## 278.2 语法

outline-offset: length|inherit;

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 轮廓与边框边缘的距离。 |
| inherit | 规定应从父元素继承 outline-offset 属性的值。 |

# 第279章 CSS outline-style 属性

设置轮廓的样式：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<style type="text/css">

p

{

border:red solid thin;

outline-style:dotted;

outline-color:#00ff00;

}

</style>

</head>

<body>

<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高版本） 才支持 outline-color 属性。</p>

</body>

</html>

## 279.1 定义和用法

outline-style 属性用于设置元素的整个轮廓的样式。样式不能是 none，否则轮廓不会出现。

outline （轮廓）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。outline 属性设置元素周围的轮廓线。

注释：请始终在 outline-color 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

注释：轮廓线不会占据空间，也不一定是矩形。

默认值： none

继承性： no

版本： CSS2

JavaScript 语法： object.style.outlineStyle="dotted"

## 279.1 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。定义无轮廓。 |
| dotted | 定义点状的轮廓。 |
| dashed | 定义虚线轮廓。 |
| solid | 定义实线轮廓。 |
| double | 定义双线轮廓。双线的宽度等同于 outline-width 的值。 |
| groove | 定义 3D 凹槽轮廓。此效果取决于 outline-color 值。 |
| ridge | 定义 3D 凸槽轮廓。此效果取决于 outline-color 值。 |
| inset | 定义 3D 凹边轮廓。此效果取决于 outline-color 值。 |
| outset | 定义 3D 凸边轮廓。此效果取决于 outline-color 值。 |
| inherit | 规定应该从父元素继承轮廓样式的设置。 |

# 第280章 CSS outline-width 属性

设置点状轮廓的颜色：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<style type="text/css">

p.one

{

border:red solid thin;

outline-style:solid;

outline-width:thin;

}

p.two

{

border:red solid thin;

outline-style:dotted;

outline-width:3px;

}

</style>

</head>

<body>

<p class="one">This is some text in a paragraph.</p>

<p class="two">This is some text in a paragraph.</p>

<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高版本） 才支持 outline-width 属性。</p>

</body>

</html>

## 280.1 定义和用法

outline-width 属性设置元素整个轮廓的宽度，只有当轮廓样式不是 none 时，这个宽度才会起作用。如果样式为 none，宽度实际上会重置为 0。不允许设置负长度值。

outline（轮廓）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。outline 属性设置元素周围的轮廓线。

注释：请始终在 outline-width 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

注释：轮廓线不会占据空间，也不一定是矩形。

默认值： medium

继承性： no

版本： CSS2

JavaScript 语法： object.style.outlineWidth="thin"

## 280.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| thin | 规定细轮廓。 |
| medium | 默认。规定中等的轮廓。 |
| thick | 规定粗的轮廓。 |
| length | 允许您规定轮廓粗细的值。 |
| inherit | 规定应该从父元素继承轮廓宽度的设置。 |

# 第281章 CSS overflow 属性

设置 overflow 属性：

<html>

<head>

<style type="text/css">

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>

<div>

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。

</div>

</body>

</html>

## 281.1 定义和用法

overflow 属性规定当内容溢出元素框时发生的事情。

说明

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： visible

继承性： no

版本： CSS2

JavaScript 语法： object.style.overflow="scroll"

## 281.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| visible | 默认值。内容不会被修剪，会呈现在元素框之外。 |
| hidden | 内容会被修剪，并且其余内容是不可见的。 |
| scroll | 内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。 |
| auto | 如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。 |
| inherit | 规定应该从父元素继承 overflow 属性的值。 |

# 第282章 CSS overflow-x 属性

裁剪 div 元素中内容的左/右边缘 - 如果溢出元素的内容区域的话：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:110px;

height:110px;

border:thin solid black;

overflow-x:hidden;

overflow-y:hidden;

}

</style>

</head>

<body>

<div><p style="width:140px">

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

</p></div>

<p>Overflow-x 是否对内容的左/右边缘进行裁剪。</p>

<p>Overflow-y 是否对内容的上/下边缘进行裁剪。</p>

</body>

</html>

## 282.1 定义和用法

overflow-x 属性规定是否对内容的左/右边缘进行裁剪 - 如果溢出元素内容区域的话。

提示：使用 overflow-y 属性来确定对上/下边缘的裁剪。

默认值： visible

继承性： no

版本： CSS3

JavaScript 语法： object.style.overflowX="scroll"

## 282.2 语法

overflow-x: visible|hidden|scroll|auto|no-display|no-content;

|  |  |
| --- | --- |
| 值 | 描述 |
| visible | 不裁剪内容，可能会显示在内容框之外。 |
| hidden | 裁剪内容 - 不提供滚动机制。 |
| scroll | 裁剪内容 - 提供滚动机制。 |
| auto | 如果溢出框，则应该提供滚动机制。 |
| no-display | 如果内容不适合内容框，则删除整个框。 |
| no-content | 如果内容不适合内容框，则隐藏整个内容。 |

# 第283章 CSS overflow-y 属性

裁剪 div 元素中内容的左/右边缘 - 如果溢出元素的内容区域的话：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:110px;

height:110px;

border:thin solid black;

overflow-x:hidden;

overflow-y:hidden;

}

</style>

</head>

<body>

<div><p style="width:140px">

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

这是一个段落。这是一个段落。这是一个段落。这是一个段落。

</p></div>

<p>Overflow-x 是否对内容的左/右边缘进行裁剪。</p>

<p>Overflow-y 是否对内容的上/下边缘进行裁剪。</p>

</body>

</html>

## 283.1 定义和用法

overflow-y 属性规定是否对内容的上/下边缘进行裁剪 - 如果溢出元素内容区域的话。

提示：使用 overflow-x 属性来确定对左/右边缘的裁剪。

默认值： visible

继承性： no

版本： CSS3

JavaScript 语法： object.style.overflowY="scroll"

## 283.2 语法

overflow-y: visible|hidden|scroll|auto|no-display|no-content;

|  |  |
| --- | --- |
| 值 | 描述 |
| visible | 不裁剪内容，可能会显示在内容框之外。 |
| hidden | 裁剪内容 - 不提供滚动机制。 |
| scroll | 裁剪内容 - 提供滚动机制。 |
| auto | 如果溢出框，则应该提供滚动机制。 |
| no-display | 如果内容不适合内容框，则删除整个框。 |
| no-content | 如果内容不适合内容框，则隐藏整个内容。 |

# 第284章 CSS padding 属性

设置 p 元素的 4 个内边距：

<html>

<head>

<style type="text/css">

td.test1 {padding: 1.5cm}

td.test2 {padding: 0.5cm 2.5cm}

</style>

</head>

<body>

<table border="1">

<tr>

<td class="test1">

这个表格单元的每个边拥有相等的内边距。

</td>

</tr>

</table>

<br />

<table border="1">

<tr>

<td class="test2">

这个表格单元的上和下内边距是 0.5cm，左和右内边距是 2.5cm。

</td>

</tr>

</table>

</body>

</html>

## 284.1 定义和用法

padding 简写属性在一个声明中设置所有内边距属性。

说明

这个简写属性设置元素所有内边距的宽度，或者设置各边上内边距的宽度。行内非替换元素上设置的内边距不会影响行高计算；因此，如果一个元素既有内边距又有背景，从视觉上看可能会延伸到其他行，有可能还会与其他内容重叠。元素的背景会延伸穿过内边距。不允许指定负边距值。

注释：不允许使用负值。

例子 1

padding:10px 5px 15px 20px;

上内边距是 10px

右内边距是 5px

下内边距是 15px

左内边距是 20px

例子 2

padding:10px 5px 15px;

上内边距是 10px

右内边距和左内边距是 5px

下内边距是 15px

例子 3

padding:10px 5px;

上内边距和下内边距是 10px

右内边距和左内边距是 5px

例子 4

padding:10px;

所有 4 个内边距都是 10px

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.padding="10px 5px"

284.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 浏览器计算内边距。 |
| length | 规定以具体单位计的内边距值，比如像素、厘米等。默认值是 0px。 |
| % | 规定基于父元素的宽度的百分比的内边距。 |
| inherit | 规定应该从父元素继承内边距。 |

# 第285章 CSS padding-bottom 属性

设置 p 元素的下内边距：

<html>

<head>

<style type="text/css">

td {padding-bottom: 2cm}

</style>

</head>

<body>

<table border="1">

<tr>

<td>

这个表格单元拥有下内边距。

</td>

</tr>

</table>

</body>

</html>

## 285.1 定义和用法

padding-bottom 属性设置元素的下内边距（底部空白）。

说明

该属性设置元素下内边距的宽度。行内非替换元素上设置的下内边距不会影响行高计算，因此，如果一个元素既有内边距又有背景，从视觉上看可能延伸到其他行，有可能还会与其他内容重叠。不允许指定负内边距值。

注释：不允许使用负值。

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.paddingBottom="10px"

## 285.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 规定以具体单位计的固定的下内边距值，比如像素、厘米等。默认值是 0px。 |
| % | 定义基于父元素宽度的百分比下内边距。此值不会如预期地那样工作于所有的浏览器中。 |
| inherit | 规定应该从父元素继承下内边距。 |

# 第286章 CSS padding-left 属性

设置 p 元素的左内边距：

<html>

<head>

<style type="text/css">

td {padding-left: 2cm}

</style>

</head>

<body>

<table border="1">

<tr>

<td>

这个表格单元拥有左内边距。

</td>

</tr>

</table>

</body>

</html>

## 286.1 定义和用法

padding-left 属性设置元素左内边距（空白）。

说明

该属性设置元素左内边距的宽度。行内非替换元素上设置的左内边距仅在元素所生成的第一个行内框的左边出现.

注释：不允许使用负值。

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.paddingLeft="10px"

## 286.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 规定以具体单位计的固定的左内边距值，比如像素、厘米等。默认值是 0px。 |
| % | 定义基于父元素宽度的百分比左内边距。此值不会如预期地那样工作于所有的浏览器中。 |
| inherit | 规定应该从父元素继承左内边距。 |

# 第287章 CSS padding-right 属性

设置 p 元素的右内边距：

<html>

<head>

<style type="text/css">

td {padding-right: 5cm}

</style>

</head>

<body>

<table border="1">

<tr>

<td>

这个表格单元拥有右内边距。

</td>

</tr>

</table>

</body>

</html>

## 287.1 定义和用法

padding-right 属性设置元素右内边距（空白）。

注释：不允许使用负值。

说明

该属性设置元素右内边距的宽度。行内非替换元素上设置的右内边距仅在元素所生成的第一个行内框的右边出现.

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.paddingRight="10px"

## 287.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 规定以具体单位计的固定的右内边距值，比如像素、厘米等。默认值是 0px。 |
| % | 定义基于父元素宽度的百分比右内边距。此值不会如预期地那样工作于所有的浏览器中。 |
| inherit | 规定应该从父元素继承右内边距。 |

# 第288章 CSS padding-top 属性

设置 p 元素的上内边距：

<html>

<head>

<style type="text/css">

td {padding-top: 2cm}

</style>

</head>

<body>

<table border="1">

<tr>

<td>

这个表格单元拥有上内边距。

</td>

</tr>

</table>

</body>

</html>

## 288.1 定义和用法

padding-top 属性设置元素的上内边距（空间）。

说明

该属性设置元素上内边距的宽度。行内非替换元素上设置的上内边距不会影响行高计算，因此，如果一个元素既有内边距又有背景，从视觉上看可能延伸到其他行，有可能还会与其他内容重叠。不允许指定负内边距值。

注释：不允许使用负值。

默认值： 0

继承性： no

版本： CSS1

JavaScript 语法： object.style.paddingTop="10px"

## 288.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| length | 规定以具体单位计的固定的上内边距值，比如像素、厘米等。默认值是 0px。 |
| % | 定义基于父元素宽度的百分比上内边距。此值不会如预期的那样工作于所有的浏览器中。 |
| inherit | 规定应该从父元素继承上内边距。 |

# 第289章 CSS page-break-after 属性

设置在表格元素之后始终进行分页的分页行为：

<html>

<head>

<style>

@media print

{

table {page-break-after:always;}

}

</style>

</head>

<body>

....

</body>

</html>

## 289.1 定义和用法

page-break-after 属性设置元素后的 page-breaking 行为。

尽管可以用 always 强制放上分页符，但是无法保证避免分页符的插入，创作人员最多只能要求用户代理尽可能避免插入分页。

应用于：position 值为 relative 或 static 的非浮动块级元素。

注释：请尽可能少地使用分页属性，并且避免在表格、浮动元素、带有边框的块元素中使用分页属性。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.pageBreakAfter="always"

## 289.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。如果必要则在元素后插入分页符。 |
| always | 在元素后插入分页符。 |
| avoid | 避免在元素后插入分页符。 |
| left | 在元素之后足够的分页符，一直到一张空白的左页为止。 |
| right | 在元素之后足够的分页符，一直到一张空白的右页为止。 |
| inherit | 规定应该从父元素继承 page-break-after 属性的设置。 |

# 第290章 CSS page-break-before 属性

实例

设置在表格元素之后始终进行分页的分页行为：

<html>

<head>

<style>

@media print

{

table {page-break-before:always;}

}

</style>

</head>

<body>

....

</body>

</html>

## 290.1 定义和用法

page-break-before 属性设置元素前的 page-breaking 行为。

尽管可以用 always 强制放上分页符，但是无法保证避免分页符的插入，创作人员最多只能要求用户代理尽可能避免插入分页。

应用于：position 值为 relative 或 static 的非浮动块级元素。

注释：请尽可能少地使用分页属性，并且避免在表格、浮动元素、带有边框的块元素中使用分页属性。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.pageBreakBefore="always"

## 290.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。如果必要则在元素前插入分页符。 |
| always | 在元素前插入分页符。 |
| avoid | 避免在元素前插入分页符。 |
| left | 在元素之前足够的分页符，一直到一张空白的左页为止。 |
| right | 在元素之前足够的分页符，一直到一张空白的右页为止。 |
| inherit | 规定应该从父元素继承 page-break-before 属性的设置。 |

# 第291章 CSS page-break-inside 属性

实例

设置在表格元素内部避免进行分页的分页行为：

<html>

<head>

<style>

@media print

{

table {page-break-inside:avoid;}

}

</style>

</head>

<body>

....

</body>

</html>

## 291.1 定义和用法

page-break-inside 属性设置元素内部的 page-breaking 行为。

尽管可以用 always 强制放上分页符，但是无法保证避免分页符的插入，创作人员最多只能要求用户代理尽可能避免插入分页。

应用于：position 值为 relative 或 static 的非浮动块级元素。

注释：请尽可能少地使用分页属性，并且避免在表格、浮动元素、带有边框的块元素中使用分页属性。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.pageBreakInside="avoid"

## 291.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。如果必要则在元素内部插入分页符。 |
| avoid | 避免在元素内部插入分页符。 |
| inherit | 规定应该从父元素继承 page-break-inside 属性的设置。 |

# 第292章 CSS perspective 属性

设置元素被查看位置的视图：

<!DOCTYPE html>

<html>

<head>

<style>

#div1

{

position: relative;

height: 150px;

width: 150px;

margin: 50px;

padding:10px;

border: 1px solid black;

perspective:150;

-webkit-perspective:150; /\* Safari and Chrome \*/

}

#div2

{

padding:50px;

position: absolute;

border: 1px solid black;

background-color: yellow;

transform: rotateX(45deg);

-webkit-transform: rotateX(45deg); /\* Safari and Chrome \*/

}

</style>

</head>

<body>

<div id="div1">

<div id="div2">HELLO</div>

</div>

</body>

</html>

## 292.1 定义和用法

perspective 属性定义 3D 元素距视图的距离，以像素计。该属性允许您改变 3D 元素查看 3D 元素的视图。

当为元素定义 perspective 属性时，其子元素会获得透视效果，而不是元素本身。

注释：perspective 属性只影响 3D 转换元素。

提示：请与 perspective-origin 属性一同使用该属性，这样您就能够改变 3D 元素的底部位置。

默认值： none

继承性： yes

版本： CSS3

JavaScript 语法： object.style.perspective=500

## 292.2 语法

perspective: number|none;

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 元素距离视图的距离，以像素计。 |
| none | 默认值。与 0 相同。不设置透视。 |

# 第293章 CSS perspective-origin 属性

设置 3D 元素的基点位置：

<!DOCTYPE html>

<html>

<head>

<style>

#div1

{

position: relative;

height: 150px;

width: 150px;

margin: 50px;

padding:10px;

border: 1px solid black;

perspective:150;

-webkit-perspective:150; /\* Safari and Chrome \*/

}

#div2

{

padding:50px;

position: absolute;

border: 1px solid black;

background-color: yellow;

transform: rotateX(45deg);

-webkit-transform: rotateX(45deg); /\* Safari and Chrome \*/

}

</style>

</head>

<body>

<div id="div1">

<div id="div2">HELLO</div>

</div>

</body>

</html>

## 293.1 定义和用法

perspective-origin 属性定义 3D 元素所基于的 X 轴和 Y 轴。该属性允许您改变 3D 元素的底部位置。

当为元素定义 perspective-origin 属性时，其子元素会获得透视效果，而不是元素本身。

注释：该属性必须与 perspective 属性一同使用，而且只影响 3D 转换元素。

默认值： 50% 50%

继承性： no

版本： CSS3

JavaScript 语法： object.style.perspectiveOrigin="10% 10%"

## 293.2 语法

perspective-origin: x-axis y-axis;

|  |  |
| --- | --- |
| 值 | 描述 |
| x-axis | 定义该视图在 x 轴上的位置。默认值：50%。  可能的值：  left  center  right  length  % |
| y-axis | 定义该视图在 y 轴上的位置。默认值：50%。  可能的值：  top  center  bottom  length  % |

# 第294章 CSS pointer-events 属性

实例

设置元素是否对指针事件做出反应：

<!DOCTYPE html>

<html>

<head>

<style>

div.ex1 {

pointer-events: none;

}

div.ex2 {

pointer-events: auto;

}

</style>

</head>

<body>

<h1>pointer-events 属性</h1>

<p>请把鼠标指针移至下面的链接，查看是否对指针事件做出反应：</p>

<h2>pointer-events: none：</h2>

<div class="ex1">访问 <a href="https://www.w3school.com.cn/html/">HTML 教程</a></div>

<h2>pointer-events: auto（默认）</h2>

<div class="ex2">访问 <a href="https://www.w3school.com.cn/css/">CSS 教程</a></div>

</body>

</html>

## 294.1 定义和用法

pointer-events 属性定义元素是否对指针事件做出反应。

默认值： auto

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.pointerEvents="none"

## 294.2 CSS 语法

pointer-events: auto|none;

属性值

|  |  |
| --- | --- |
| 属性值 | 描述 |
| auto | 默认值。元素对指针事件做出反应，比如 :hover 和 click。 |
| none | 元素不对指针事件做出反应。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第295章 CSS position 属性

定位 h2 元素：

<html>

<head>

<style type="text/css">

h2.pos\_abs

{

position:absolute;

left:100px;

top:150px

}

</style>

</head>

<body>

<h2 class="pos\_abs">这是带有绝对定位的标题</h2>

<p>通过绝对定位，元素可以放置到页面上的任何位置。下面的标题距离页面左侧 100px，距离页面顶部 150px。</p>

</body>

</html>

## 295.1 定义和用法

position 属性规定元素的定位类型。

说明

这个属性定义建立元素布局所用的定位机制。任何元素都可以定位，不过绝对或固定元素会生成一个块级框，而不论该元素本身是什么类型。相对定位元素会相对于它在正常流中的默认位置偏移。

默认值： static

继承性： no

版本： CSS2

JavaScript 语法： object.style.position="absolute"

## 295.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| absolute | 生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。  元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。 |
| fixed | 生成绝对定位的元素，相对于浏览器窗口进行定位。  元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。 |
| relative | 生成相对定位的元素，相对于其正常位置进行定位。  因此，"left:20" 会向元素的 LEFT 位置添加 20 像素。 |
| static | 默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明） |
| inherit | 规定应该从父元素继承 position 属性的值。 |
|  |  |

# 第296章 CSS quotes 属性

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html lang="en">

<head>

<style type="text/css">

q:lang(en)

{

quotes: '"' '"' "'" "'"

}

</style>

</head>

<body>

<p><q>This is a <q>big</q> quote.</q></p>

<p><b>注释：</b>如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 quotes 属性。</p>

</body>

</html>

## 296.1 定义和用法

quotes 属性设置嵌套引用（embedded quotation）的引号类型。

默认值： not specified

继承性： yes

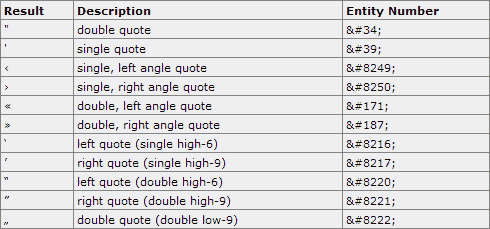
版本： CSS2

JavaScript 语法： object.style.quotes="none"

## 296.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 规定 "content" 属性的 "open-quote" 和 "close-quote" 的值不会产生任何引号。 |
| string string string string | 定义要使用的引号。  前两个值规定第一级引用嵌套，后两个值规定下一级引号嵌套。 |
| inherit | 规定应该从父元素继承 quotes 属性的值。 |

## 296.3 引号字符



# 第297章 CSS resize 属性

规定可以由用户调整 div 元素的大小：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

border:2px solid;

padding:10px 40px;

width:300px;

resize:both;

overflow:auto;

}

</style>

</head>

<body>

<div>resize 属性规定是否可由用户调整元素尺寸。</div>

<p><b>注释：</b> Firefox 4+、Safari 以及 Chrome 支持 resize 属性。</p>

</body>

</html>

## 297.1 定义和用法

resize 属性规定是否可由用户调整元素的尺寸。

注释：如果希望此属性生效，需要设置元素的 overflow 属性，值可以是 auto、hidden 或 scroll。

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.resize="both"

## 297.2 语法

resize: none|both|horizontal|vertical;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 用户无法调整元素的尺寸。 |
| both | 用户可调整元素的高度和宽度。 |
| horizontal | 用户可调整元素的宽度。 |
| vertical | 用户可调整元素的高度。 |

# 第298章 CSS right 属性

把图像的右边缘设置在其包含元素右边缘向左 5 像素的位置：

<html>

<head>

<style type="text/css">

img

{

position:absolute;

right:50px;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<img src="/i/w3school\_logo\_white.gif" />

</body>

</html>

## 298.1 定义和用法

right 属性规定元素的右边缘。该属性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。

注释：如果 "position" 属性的值为 "static"，那么设置 "right" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，left 的计算值始终等于 right。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.right="50px"

## 298.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。通过浏览器计算右边缘的位置。 |
| % | 设置以包含元素的百分比计的右边位置。可使用负值。 |
| length | 使用 px、cm 等单位设置元素的右边位置。可使用负值。 |
| inherit | 规定应该从父元素继承 right 属性的值。 |

# 第299章 CSS scroll-behavior 属性

为文档添加平滑滚动效果：

<!DOCTYPE html>

<html>

<head>

<style>

html {

scroll-behavior: smooth;

}

#section1 {

height: 600px;

background-color: pink;

}

#section2 {

height: 600px;

background-color: yellow;

}

</style>

</head>

<body>

<h1>平滑的滚动</h1>

<div class="main" id="section1">

<h2>第一章</h2>

<p>请点击链接来查看平滑的滚动效果。</p>

<a href="#section2">点击我会平滑滚动到下面的第二章</a>

<p>注释：删除 scroll-behavior 属性可清楚平滑滚动。</p>

</div>

<div class="main" id="section2">

<h2>第二章</h2>

<a href="#section1">点击我可平滑滚动到上面的第一章</a>

</div>

<p><b>注释：</b>Internet Explorer 不支持 scroll-behavior 属性。</p>

</body>

</html>

## 299.1 定义和用法

scroll-behavior 属性规定当用户单击可滚动框中的链接时，是否平滑地（具动画效果）滚动位置，而不是直线跳转。

默认值： auto

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSSOM View Module（工作草案）

JavaScript 语法： object.style.scrollBehavior="smooth"

## 299.2 CSS 语法

scroll-behavior: auto|smooth|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。允许在滚动框内的元素间直接跳转的“滚动效果”。 |
| smooth | 允许在滚动框内的元素间平滑的“滚动效果”。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |
|  |  |

# 第300章 CSS tab-size 属性

将制表符的宽度设置为 16 个空格：

<!DOCTYPE html>

<html>

<head>

<style>

#t1 {

-moz-tab-size: 4; /\* Firefox \*/

tab-size: 4;

}

#t2 {

-moz-tab-size: 16; /\* Firefox \*/

tab-size: 16;

}

</style>

</head>

<body>

<h1>tab-size 属性</h1>

<pre id="t1">

I use tab-size 4

</pre>

<pre id="t2">

I use tab-size 16

</pre>

<p><b>注释：</b>Firefox 支持替代的 -moz-tab-size 属性。</p>

</body>

</html>

## 300.1 定义和用法

tab-size 属性规定制表符的宽度。

在 HTML 中，制表符通常显示为单个空格，除了某些元素（比如 <textarea> 和 <pre>）外，并且 tab-size 属性的结果仅对那些元素可见。

默认值： 8

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.tabSize="16"

## 300.2 CSS 语法

tab-size: number|length|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| number | 每个制表符要显示的空格字符数。默认值是 8。 |
| length | 制表符的长度。主流浏览器都不支持这个属性值。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第301章 CSS table-layout 属性

设置表格布局算法：

<html>

<head>

<style type="text/css">

table.one

{

table-layout: automatic

}

table.two

{

table-layout: fixed

}

</style>

</head>

<body>

<table class="one" border="1" width="100%">

<tr>

<td width="20%">1000000000000000000000000000</td>

<td width="40%">10000000</td>

<td width="40%">100</td>

</tr>

</table>

<br />

<table class="two" border="1" width="100%">

<tr>

<td width="20%">1000000000000000000000000000</td>

<td width="40%">10000000</td>

<td width="40%">100</td>

</tr>

</table>

</body>

</html>

## 301.1 定义和用法

tableLayout 属性用来显示表格单元格、行、列的算法规则。

固定表格布局：

固定表格布局与自动表格布局相比，允许浏览器更快地对表格进行布局。

在固定表格布局中，水平布局仅取决于表格宽度、列宽度、表格边框宽度、单元格间距，而与单元格的内容无关。

通过使用固定表格布局，用户代理在接收到第一行后就可以显示表格。

自动表格布局：

在自动表格布局中，列的宽度是由列单元格中没有折行的最宽的内容设定的。

此算法有时会较慢，这是由于它需要在确定最终的布局之前访问表格中所有的内容。

说明

该属性指定了完成表布局时所用的布局算法。固定布局算法比较快，但是不太灵活，而自动算法比较慢，不过更能反映传统的 HTML 表。

默认值： auto

继承性： yes

版本： CSS2

JavaScript 语法： object.style.tableLayout="fixed"

## 301.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| automatic | 默认。列宽度由单元格内容设定。 |
| fixed | 列宽由表格宽度和列宽度设定。 |
| inherit | 规定应该从父元素继承 table-layout 属性的值。 |

# 第302章 CSS text-align 属性

设置 h1、h2、h3 元素的文本对齐方式：

<html>

<head>

<style type="text/css">

h1 {text-align: center}

h2 {text-align: left}

h3 {text-align: right}

</style>

</head>

<body>

<h1>这是标题 1</h1>

<h2>这是标题 2</h2>

<h3>这是标题 3</h3>

</body>

</html>

## 302.1 定义和用法

text-align 属性规定元素中的文本的水平对齐方式。

该属性通过指定行框与哪个点对齐，从而设置块级元素内文本的水平对齐方式。通过允许用户代理调整行内容中字母和字之间的间隔，可以支持值 justify；不同用户代理可能会得到不同的结果。

默认值： 如果 direction 属性是 ltr，则默认值是 left；如果 direction 是 rtl，则为 right。

继承性： yes

版本： CSS1

JavaScript 语法： object.style.textAlign="right"

## 302.2 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。

# 第303章 CSS text-align-last 属性

对齐三个 <div> 元素中的最后一行：

<!DOCTYPE html>

<html>

<head>

<style>

div.a {

text-align: justify; /\* For Edge \*/

-moz-text-align-last: right; /\* 针对 58.0 之前的 Firefox \*/

text-align-last: right;

}

div.b {

text-align: justify; /\* For Edge \*/

-moz-text-align-last: center; /\* 针对 58.0 之前的 Firefox \*/

text-align-last: center;

}

div.c {

text-align: justify; /\* For Edge \*/

-moz-text-align-last: justify; /\* 针对 58.0 之前的 Firefox \*/

text-align-last: justify;

}

</style>

</head>

<body>

<h1>text-align-last 属性</h1>

<h2>text-align-last: right:</h2>

<div class="a">

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

<h2>text-align-last: center:</h2>

<div class="b">

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

<h2>text-align-last: justify:</h2>

<div class="c">

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

</body>

</html>

## 303.1 定义和用法

text-align-last 属性规定如何对齐文本的最后一行。

请注意，text-align-last 属性设置的是被选元素内的所有最末行。所以，如果一个 <div> 中有三个段落，text-align-last 会应用于每段的最后一行。如需在容器中的最后一段上使用 text-align-last，您可使用 :last child，请看下面的例子。

注释：在 Edge/Internet Explorer 中，text-align-last 属性仅适用于具有 "text-align: justify" 的文本。

默认值： auto

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.textAlignLast="right"

## 303.2 CSS 语法

text-align-last: auto|left|right|center|justify|start|end|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。每一行的对齐规则由 text-align 的值来确定。  当 text-align 的值是 justify，text-align-last 的表现和设置了 start 的表现是一样的。 |
| eft | 最后一行向左对齐。 |
| right | 最后一行向右对齐。 |
| center | 最后一行居中对齐。 |
| justify | 最后一行文字的开头与内容盒的左侧对齐，末尾与右侧对齐。 |
| start | 最后一行在行起点对齐。  如果 text-direction 为从左到右，则左对齐，如果 text-direction 为从右到左，则右对齐。 |
| end | 最后一行在行末尾对齐。  如果 text-direction 为从左到右，则右对齐，如果 text-direction 为从右到左，则左对齐。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

更多实例

在容器中的最后一行上使用 text-align-last：

<!DOCTYPE html>

<html>

<head>

<style>

div.a {

text-align: justify; /\* 针对 Edge \*/

-moz-text-align-last: center; /\* 针对 58.0 之前的 Firefox \*/

text-align-last: center;

}

div.b {

text-align: justify; /\* 针对 Edge \*/

}

div.b p:last-child {

-moz-text-align-last: center; /\* 针对 58.0 之前的 Firefox \*/

text-align-last: center;

}

</style>

</head>

<body>

<h1>text-align-last 属性</h1>

<h2>text-align-last: center（影响所有段落）：</h2>

<div class="a">

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

<h2>text-align-last: center;（只影响最后一段）：</h2>

<div class="b">

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

<p>Shanghai is one of the four direct-administered municipalities of the People's Republic of China. Welcome to Shanghai!</p>

</div>

</body>

</html>

# 第304章 CSS text-decoration 属性

设置 h1、h2、h3、h4 元素的文本修饰：

<html>

<head>

<style type="text/css">

h1 {text-decoration: overline}

h2 {text-decoration: line-through}

h3 {text-decoration: underline}

h4 {text-decoration:blink}

a {text-decoration: none}

</style>

</head>

<body>

<h1>这是标题 1</h1>

<h2>这是标题 2</h2>

<h3>这是标题 3</h3>

<h4>这是标题 4</h4>

<p><a href="http://www.w3school.com.cn/index.html">这是一个链接</a></p>

</body>

</html>

## 304.1 定义和用法

text-decoration 属性规定添加到文本的修饰。

注释：修饰的颜色由 "color" 属性设置。

说明

这个属性允许对文本设置某种效果，如加下划线。如果后代元素没有自己的装饰，祖先元素上设置的装饰会“延伸”到后代元素中。不要求用户代理支持 blink。

默认值： none

继承性： no

版本： CSS1

JavaScript 语法： object.style.textDecoration="overline"

## 304.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。定义标准的文本。 |
| underline | 定义文本下的一条线。 |
| overline | 定义文本上的一条线。 |
| line-through | 定义穿过文本下的一条线。 |
| blink | 定义闪烁的文本。 |
| inherit | 规定应该从父元素继承 text-decoration 属性的值。 |

# 第305章 CSS text-decoration-color 属性

把 text-decoration 的颜色设置为红色：

<!DOCTYPE html>

<html>

<head>

<style>

p {

text-decoration: underline;

-webkit-text-decoration-color: red; /\* Safari \*/

text-decoration-color: red;

}

</style>

</head>

<body>

<h1>text-decoration-color 属性</h1>

<p>现在，下划线的颜色应该是红色！</p>

<p><b>注释：</b>版本 79 之前的 Edge 不支持 text-decoration-color 属性。</p>

</body>

</html>

## 305.1 定义和用法

text-decoration-color 属性规定 text-decoration（下划线、上划线、穿线）的颜色。

默认值： currentColor

继承： 否

动画制作： 支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.textDecorationColor="red"

## 305.2 CSS 语法

text-decoration-color: color|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| color | 规定 text-decoration 的颜色。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第306章 CSS text-decoration-line 属性

设置不同的 text-decoration 线条类型：

<!DOCTYPE html>

<html>

<head>

<style>

div.a {

-webkit-text-decoration-line: overline; /\* Safari \*/

text-decoration-line: overline;

}

div.b {

-webkit-text-decoration-line: underline; /\* Safari \*/

text-decoration-line: underline;

}

div.c {

-webkit-text-decoration-line: line-through; /\* Safari \*/

text-decoration-line: line-through;

}

div.d {

-webkit-text-decoration-line: overline underline; /\* Safari \*/

text-decoration-line: overline underline;

}

</style>

</head>

<body>

<h1>text-decoration-line 属性</h1>

<div class="a">This is some text with a line on top.</div>

<br>

<div class="b">This is some text with an underline.</div>

<br>

<div class="c">This is some text with a line-through.</div>

<br>

<div class="d">This is some text with an overline and an underline.</div>

<p><b>注释：</b>版本 79 之前的 Edge 不支持 text-decoration-line 属性，Safari 只支持 -webkit- 前缀。</p>

</body>

</html>

## 306.1 定义和用法

text-decoration-line 属性设置要使用哪种文本装饰的类型（下划线、上划线、穿线）。

提示：同时请参阅 text-decoration 属性，它是 text-decoration-line、text-decoration-style 以及 text-decoration-color 的简写属性。

注释：您还可以组合多个值，比如 underline 和 overline，在文本上下方同时显示线条。

默认值： none

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.textDecorationLine="overline"

## 306.2 CSS 语法

text-decoration-line: none|underline|overline|line-through|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认值。规定 text-decoration 没有线条。 |
| underline | 规定在文本下方显示线条。 |
| overline | 规定在文本上方显示线条。 |
| line-through | 规定显示横穿文本的线条。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第307章 CSS text-decoration-style 属性

设置不同类型的 text-decoration 样式：

<!DOCTYPE html>

<html>

<head>

<style>

div.a {

text-decoration-line: underline;

text-decoration-style: solid;

}

div.b {

text-decoration-line: underline;

text-decoration-style: wavy;

}

div.c {

text-decoration-line: underline;

text-decoration-style: double;

}

div.d {

text-decoration-line: overline underline;

text-decoration-style: wavy;

}

</style>

</head>

<body>

<h1>text-decoration-style 属性</h1>

<div class="a">This is some text with a solid underline.</div>

<br>

<div class="b">This is some text with a wavy underline.</div>

<br>

<div class="c">This is some text with a double underline.</div>

<br>

<div class="d">This is some text with a wavy over- and underline.</div>

<p><b>注释：</b>版本 79 之前的 Edge 不支持 text-decoration-line 属性。</p>

</body>

</html>

## 307.1 定义和用法

text-decoration-style 属性设置文本装饰的类型（实线、波浪线、点线、虚线、双线）。

提示：同时请参阅 text-decoration 属性，它是这三个属性的简写属性：

text-decoration-line

text-decoration-style

text-decoration-color

默认值： solid

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.textDecorationStyle="wavy"

## 307.2 CSS 语法

text-decoration-style: solid|double|dotted|dashed|wavy|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| solid | 默认值。线条显示为单行。 |
| double | 线条显示为双线。 |
| dotted | 线条显示为点线。 |
| dashed | 线条显示为虚线。 |
| wavy | 线条显示为波浪线。 |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第308章 CSS text-indent 属性

将段落的第一行缩进 50 像素：

<html>

<head>

<style type="text/css">

p {text-indent: 1cm}

</style>

</head>

<body>

<p>

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

这是段落中的一些文本。

</p>

</body>

</html>

## 308.1 定义

text-indent 属性规定文本块中首行文本的缩进。

注释：允许使用负值。如果使用负值，那么首行会被缩进到左边。

注意：在 CSS 2.1 之前，text-indent 总是继承计算值，而不是声明值。

说明

用于定义块级元素中第一个内容行的缩进。这最常用于建立一个“标签页”效果。允许指定负值，这会产生一种“悬挂缩进”的效果。

默认值： not specified

继承性： yes

版本： CSS1

JavaScript 语法： object.style.textIndent="50px"

## 308.2 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。 |

# 第309章 CSS text-justify 属性

齐行改变单词间的间隔：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

text-align:justify;

text-justify:inter-word;

}

</style>

</head>

<body>

<h1>CSS text-justify 实例</h1>

<div>Shanghai is the largest city by population in the People's Republic of China (PRC) and the largest city proper by population in the world. It is one of the four province-level municipalities of the PRC, with a total population of over 23 million as of 2010. It is a global city, with influence in commerce, culture, finance, media, fashion, technology, and transport. It is a major financial center and the busiest container port in the world.</div>

<p><b>提示：</b>请调整浏览器窗口的大小，来查看齐行效果。</p>

<p><b>注释：</b>text-justify 属性只在 IE 中有效。</p>

</body>

</html>

## 309.1 定义和用法

text-justify 属性指定 text-align 设置为 "justify" 时文本的对齐方式。

该属性规定如何对齐行文本进行对齐和分隔。

默认值： auto

继承性： yes

版本： CSS3

JavaScript 语法： object.style.textJustify="inter-word"

## 309.2 语法

text-justify: auto|inter-word|inter-ideograph|inter-cluster|distribute|kashida|trim;

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 浏览器决定齐行算法。 |
| none | 禁用齐行。 |
| inter-word | 增加/减少单词间的间隔。 |
| inter-ideograph | 用表意文本来排齐内容。 |
| inter-cluster | 只对不包含内部单词间隔的内容（比如亚洲语系）进行排齐。 |
| distribute | 类似报纸版面，除了在东亚语系中最后一行是不齐行的。 |
| kashida | 通过拉伸字符来排齐内容。 |

# 第310章 CSS text-overflow 属性

使用 text-overflow 属性：

<!DOCTYPE html>

<html>

<head>

<style>

div.test

{

white-space:nowrap;

width:12em;

overflow:hidden;

border:1px solid #000000;

}

</style>

</head>

<body>

<p>下面两个 div 包含无法在框中容纳的长文本。正如您所见，文本被修剪了。</p>

<p>这个 div 使用 "text-overflow:ellipsis" ：</p>

<div class="test" style="text-overflow:ellipsis;">This is some long text that will not fit in the box</div>

<p>这个 div 使用 "text-overflow:clip"：</p>

<div class="test" style="text-overflow:clip;">This is some long text that will not fit in the box</div>

</body>

</html>

## 310.1 定义和用法

text-overflow 属性规定当文本溢出包含元素时发生的事情。

默认值： clip

继承性： no

版本： CSS3

JavaScript 语法： object.style.textOverflow="ellipsis"

## 310.2 语法

text-overflow: clip|ellipsis|string;

|  |  |
| --- | --- |
| 值 | 描述 |
| clip | 修剪文本。 |
| ellipsis | 显示省略符号来代表被修剪的文本。 |
| string | 使用给定的字符串来代表被修剪的文本。 |

# 第311章 CSS text-shadow 属性

基础的文本阴影效果：

<!DOCTYPE html>

<html>

<head>

<style>

h1

{

text-shadow: 5px 5px 5px #FF0000;

}

</style>

</head>

<body>

<h1>文本阴影效果！</h1>

</body>

</html>

## 311.1 定义和用法

text-shadow 属性向文本设置阴影。

默认值： none

继承性： yes

版本： CSS3

JavaScript 语法： object.style.textShadow="2px 2px #ff0000"

## 311.2 语法

text-shadow: h-shadow v-shadow blur color;

注释：text-shadow 属性向文本添加一个或多个阴影。该属性是逗号分隔的阴影列表，每个阴影有两个或三个长度值和一个可选的颜色值进行规定。省略的长度是 0。

|  |  |
| --- | --- |
| 值 | 描述 |
| h-shadow | 必需。水平阴影的位置。允许负值。 |
| v-shadow | 必需。垂直阴影的位置。允许负值。 |
| blur | 可选。模糊的距离。 |
| color | 可选。阴影的颜色。参阅 CSS 颜色值。 |

# 第312章 CSS text-transform 属性

转换不同元素中的文本：

<html>

<head>

<style type="text/css">

h1 {text-transform: uppercase}

p.uppercase {text-transform: uppercase}

p.lowercase {text-transform: lowercase}

p.capitalize {text-transform: capitalize}

</style>

</head>

<body>

<h1>This Is An H1 Element</h1>

<p class="uppercase">This is some text in a paragraph.</p>

<p class="lowercase">This is some text in a paragraph.</p>

<p class="capitalize">This is some text in a paragraph.</p>

</body>

</html>

## 312.1 定义和用法

text-transform 属性控制文本的大小写。

说明

这个属性会改变元素中的字母大小写，而不论源文档中文本的大小写。如果值为 capitalize，则要对某些字母大写，但是并没有明确定义如何确定哪些字母要大写，这取决于用户代理如何识别出各个“词”。

默认值： none

继承性： yes

版本： CSS1

JavaScript 语法： object.style.textTransform="uppercase"

## 312.2 提示和注释

注释：不同的用户代理可能会用不同的方法来确定单词从哪里开始，相应地确定哪些字母要大写。例如，文本 "w3-school" 可以用两种方式显示："W3-school" 和 "W3-School"。CSS 并没有规定哪一种是正确的，所以这两种都是可以的。

## 312.3 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 默认。定义带有小写字母和大写字母的标准的文本。 |
| capitalize | 文本中的每个单词以大写字母开头。 |
| uppercase | 定义仅有大写字母。 |
| lowercase | 定义无大写字母，仅有小写字母。 |
| inherit | 规定应该从父元素继承 text-transform 属性的值。 |

# 第313章 CSS top 属性

把图像的上边缘设置在其包含元素上边缘之下 5 像素高的位置：

<html>

<head>

<style type="text/css">

img

{

position:absolute;

top:5px;

}

</style>

</head>

<body>

<img src="/i/w3school\_logo\_white.gif" />

<h1>This is a heading</h1>

</body>

</html>

## 313.1 定义和用法

top 属性规定元素的顶部边缘。该属性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。

注释：如果 "position" 属性的值为 "static"，那么设置 "top" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，如果 top 和 bottom 都是 auto，其计算值则都是 0；如果其中之一为 auto，则取另一个值的相反数；如果二者都不是 auto，bottom 将取 top 值的相反数。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.top="50px"

## 313.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。通过浏览器计算上边缘的位置。 |
| % | 设置以包含元素的百分比计的上边位置。可使用负值。 |
| length | 使用 px、cm 等单位设置元素的上边位置。可使用负值。 |
| inherit | 规定应该从父元素继承 top 属性的值。 |
|  |  |

# 第314章 CSS transform 属性

旋转 div 元素：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

margin:30px;

width:200px;

height:100px;

background-color:yellow;

/\* Rotate div \*/

transform:rotate(9deg);

-ms-transform:rotate(9deg); /\* Internet Explorer \*/

-moz-transform:rotate(9deg); /\* Firefox \*/

-webkit-transform:rotate(9deg); /\* Safari 和 Chrome \*/

-o-transform:rotate(9deg); /\* Opera \*/

}

</style>

</head>

<body>

<div>Hello World</div>

</body>

</html>

## 314.1 定义和用法

transform 属性向元素应用 2D 或 3D 转换。该属性允许我们对元素进行旋转、缩放、移动或倾斜。

为了更好地理解 transform 属性，请查看这个演示。

默认值： none

继承性： no

版本： CSS3

JavaScript 语法： object.style.transform="rotate(7deg)"

## 314.2 语法

transform: none|transform-functions;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 定义不进行转换。 |
| matrix(n,n,n,n,n,n) | 定义 2D 转换，使用六个值的矩阵。 |
| matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n) | 定义 3D 转换，使用 16 个值的 4x4 矩阵。 |
| translate(x,y) | 定义 2D 转换。 |
| translate3d(x,y,z) | 定义 3D 转换。 |
| translateX(x) | 定义转换，只是用 X 轴的值。 |
| translateY(y) | 定义转换，只是用 Y 轴的值。 |
| translateZ(z) | 定义 3D 转换，只是用 Z 轴的值。 |
| scale(x,y) | 定义 2D 缩放转换。 |
| scale3d(x,y,z) | 定义 3D 缩放转换。 |
| scaleX(x) | 通过设置 X 轴的值来定义缩放转换。 |
| scaleY(y) | 通过设置 Y 轴的值来定义缩放转换。 |
| scaleZ(z) | 通过设置 Z 轴的值来定义 3D 缩放转换。 |
| rotate(angle) | 定义 2D 旋转，在参数中规定角度。 |
| rotate3d(x,y,z,angle) | 定义 3D 旋转。 |
| rotateX(angle) | 定义沿着 X 轴的 3D 旋转。 |
| rotateY(angle) | 定义沿着 Y 轴的 3D 旋转。 |
| rotateZ(angle) | 定义沿着 Z 轴的 3D 旋转。 |
| skew(x-angle,y-angle) | 定义沿着 X 和 Y 轴的 2D 倾斜转换。 |
| skewX(angle) | 定义沿着 X 轴的 2D 倾斜转换。 |
| skewY(angle) | 定义沿着 Y 轴的 2D 倾斜转换。 |
| perspective(n) | 为 3D 转换元素定义透视视图。 |

# 第315章 CSS transform-origin 属性

设置旋转元素的基点位置：

<!DOCTYPE html>

<html>

<head>

<style>

#div1

{

position: relative;

height: 200px;

width: 200px;

margin: 100px;

padding:10px;

border: 1px solid black;

}

#div2

{

padding:50px;

position: absolute;

border: 1px solid black;

background-color: yellow;

transform: rotate(45deg);

transform-origin:20% 40%;

-ms-transform: rotate(45deg); /\* IE 9 \*/

-ms-transform-origin:20% 40%; /\* IE 9 \*/

-webkit-transform: rotate(45deg); /\* Safari and Chrome \*/

-webkit-transform-origin:20% 40%; /\* Safari and Chrome \*/

-moz-transform: rotate(45deg); /\* Firefox \*/

-moz-transform-origin:20% 40%; /\* Firefox \*/

-o-transform: rotate(45deg); /\* Opera \*/

-o-transform-origin:20% 40%; /\* Opera \*/

}

</style>

</head>

<body>

<div id="div1">

<div id="div2">HELLO</div>

</div>

</body>

</html>

## 315.1 定义和用法

transform-origin 属性允许您改变被转换元素的位置。

2D 转换元素能够改变元素 x 和 y 轴。3D 转换元素还能改变其 Z 轴。

为了更好地理解 transform-origin 属性，请查看这个演示。

Safari/Chrome 用户：为了更好地理解 transform-origin 属性用于 3D 转换的情况，请查看这个演示。

注释：该属性必须与 transform 属性一同使用。

为了更好地理解 transform 属性，请查看这个演示。

默认值： 50% 50% 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.transformOrigin="20% 40%"

## 315.2 语法

transform-origin: x-axis y-axis z-axis;

|  |  |
| --- | --- |
| 值 | 描述 |
| x-axis | 定义视图被置于 X 轴的何处。可能的值：  left  center  right  length  % |
| y-axis | 定义视图被置于 Y 轴的何处。可能的值：  top  center  bottom  length  % |
| z-axis | 定义视图被置于 Z 轴的何处。可能的值：  length |

# 第316章 CSS transform-style 属性

使被转换的子元素保留其 3D 转换：

<!DOCTYPE html>

<html>

<head>

<style>

#div1

{

position: relative;

height: 200px;

width: 200px;

margin: 100px;

padding:10px;

border: 1px solid black;

}

#div2

{

padding:50px;

position: absolute;

border: 1px solid black;

background-color: red;

transform: rotateY(60deg);

transform-style: preserve-3d;

-webkit-transform: rotateY(60deg); /\* Safari and Chrome \*/

-webkit-transform-style: preserve-3d; /\* Safari and Chrome \*/

}

#div3

{

padding:40px;

position: absolute;

border: 1px solid black;

background-color: yellow;

transform: rotateY(80deg);

-webkit-transform: rotateY(-60deg); /\* Safari and Chrome \*/

}

</style>

</head>

<body>

<div id="div1">

<div id="div2">HELLO

<div id="div3">YELLOW</div>

</div>

</div>

</body>

</html>

## 316.1 定义和用法

transform-style 属性规定如何在 3D 空间中呈现被嵌套的元素。

注释：该属性必须与 transform 属性一同使用。

默认值： flat

继承性： no

版本： CSS3

JavaScript 语法： object.style.transformStyle="preserve-3d"

## 316.2 语法

transform-style: flat|preserve-3d;

|  |  |
| --- | --- |
| 值 | 描述 |
| flat | 子元素将不保留其 3D 位置。 |
| preserve-3d | 子元素将保留其 3D 位置。 |

# 第317章 CSS transition 属性

把鼠标指针放到 div 元素上，其宽度会从 100px 逐渐变为 300px：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:blue;

transition:width 2s;

-moz-transition:width 2s; /\* Firefox 4 \*/

-webkit-transition:width 2s; /\* Safari and Chrome \*/

-o-transition:width 2s; /\* Opera \*/

}

div:hover

{

width:300px;

}

</style>

</head>

<body>

<div></div>

<p>请把鼠标指针移动到蓝色的 div 元素上，就可以看到过渡效果。</p>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

</body>

</html>

## 317.1 定义和用法

transition 属性是一个简写属性，用于设置四个过渡属性：

transition-property

transition-duration

transition-timing-function

transition-delay

注释：请始终设置 transition-duration 属性，否则时长为 0，就不会产生过渡效果。

默认值： all 0 ease 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.transition="width 2s"

## 317.2 语法

transition: property duration timing-function delay;

|  |  |
| --- | --- |
| 值 | 描述 |
| transition-property | 规定设置过渡效果的 CSS 属性的名称。 |
| transition-duration | 规定完成过渡效果需要多少秒或毫秒。 |
| transition-timing-function | 规定速度效果的速度曲线。 |
| transition-delay | 定义过渡效果何时开始。 |

# 第318章 CSS transition-delay 属性

在过渡效果开始前等待 2 秒：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:blue;

transition-property:width;

transition-duration:5s;

transition-delay:2s;

/\* Firefox 4 \*/

-moz-transition-property:width;

-moz-transition-duration:5s;

-moz-transition-delay:2s;

/\* Safari and Chrome \*/

-webkit-transition-property:width;

-webkit-transition-duration:5s;

-webkit-transition-delay:2s;

/\* Opera \*/

-o-transition-property:width;

-o-transition-duration:5s;

-o-transition-delay:2s;

}

div:hover

{

width:300px;

}

</style>

</head>

<body>

<div></div>

<p>请把鼠标指针移动到蓝色的 div 元素上，就可以看到过渡效果。</p>

<p><b>注释：</b>过渡效果会在开始前等待两秒钟。</p>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

</body>

</html>

## 318.1 定义和用法

transition-delay 属性规定过渡效果何时开始。

transition-delay 值以秒或毫秒计。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.transitionDelay="2s"

## 318.2 语法

transition-delay: time;

|  |  |
| --- | --- |
| 值 | 描述 |
| time | 规定在过渡效果开始之前需要等待的时间，以秒或毫秒计。 |

# 第319章 CSS transition-duration 属性

让过渡效果持续 5 秒：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:blue;

transition-property:width;

transition-duration:5s;

/\* Firefox 4 \*/

-moz-transition-property:width;

-moz-transition-duration:5s;

/\* Safari and Chrome \*/

-webkit-transition-property:width;

-webkit-transition-duration:5s;

/\* Opera \*/

-o-transition-property:width;

-o-transition-duration:5s;

}

div:hover

{

width:300px;

}

</style>

</head>

<body>

<div></div>

<p>请把鼠标指针移动到蓝色的 div 元素上，就可以看到过渡效果。</p>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

</body>

</html>

## 319.1 定义和用法

transition-duration 属性规定完成过渡效果需要花费的时间（以秒或毫秒计）。

默认值： 0

继承性： no

版本： CSS3

JavaScript 语法： object.style.transitionDuration="5s"

## 319.2 语法

transition-duration: time;

|  |  |
| --- | --- |
| 值 | 描述 |
| time | 规定完成过渡效果需要花费的时间（以秒或毫秒计）。  默认值是 0，意味着不会有效果。 |
|  |  |

# 第320章 CSS transition-property 属性

把鼠标指针放到 div 元素上，会产生带有平滑改变元素宽度的过渡效果：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:blue;

transition-property: width;

transition-duration: 2s;

-moz-transition-property: width; /\* Firefox 4 \*/

-moz-transition-duration: 2s; /\* Firefox 4 \*/

-webkit-transition-property: width; /\* Safari and Chrome \*/

-webkit-transition-duration: 2s; /\* Safari and Chrome \*/

-o-transition-property: width; /\* Opera \*/

-o-transition-duration: 2s; /\* Opera \*/

}

div:hover

{

width:300px;

}

</style>

</head>

<body>

<div></div>

<p>请把鼠标指针移动到蓝色的 div 元素上，就可以看到过渡效果。</p>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

</body>

</html>

## 320.1 定义和用法

transition-property 属性规定应用过渡效果的 CSS 属性的名称。（当指定的 CSS 属性改变时，过渡效果将开始）。

提示：过渡效果通常在用户将鼠标指针浮动到元素上时发生。

注释：请始终设置 transition-duration 属性，否则时长为 0，就不会产生过渡效果。

默认值： all

继承性： no

版本： CSS3

JavaScript 语法： object.style.transitionProperty="width,height"

## 320.2 语法

transition-property: none|all|property;

|  |  |
| --- | --- |
| 值 | 描述 |
| none | 没有属性会获得过渡效果。 |
| all | 所有属性都将获得过渡效果。 |
| property | 定义应用过渡效果的 CSS 属性名称列表，列表以逗号分隔。 |

# 第321章 CSS transition-timing-function 属性

以相同的速度从开始到结束的过渡效果：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:100px;

background:blue;

transition:width 2s;

transition-timing-function:linear;

/\* Firefox 4 \*/

-moz-transition:width 2s;

-moz-transition-timing-function:linear;

/\* Safari and Chrome \*/

-webkit-transition:width 2s;

-webkit-transition-timing-function:linear;

/\* Opera \*/

-o-transition:width 2s;

-o-transition-timing-function:linear;

}

div:hover

{

width:300px;

}

</style>

</head>

<body>

<div></div>

<p>请把鼠标指针移动到蓝色的 div 元素上，就可以看到过渡效果。</p>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

</body>

</html>

## 321.1 定义和用法

transition-timing-function 属性规定过渡效果的速度曲线。

该属性允许过渡效果随着时间来改变其速度。

默认值： ease

继承性： no

版本： CSS3

JavaScript 语法： object.style.transitionTimingFunction="linear"

## 321.2 语法

transition-timing-function: linear|ease|ease-in|ease-out|ease-in-out|cubic-

bezier(n,n,n,n);

|  |  |
| --- | --- |
| 值 | 描述 |
| linear | 规定以相同速度开始至结束的过渡效果（等于 cubic-bezier(0,0,1,1)）。 |
| ease | 规定慢速开始，然后变快，然后慢速结束的过渡效果（cubic-bezier(0.25,0.1,0.25,1)）。 |
| ease-in | 规定以慢速开始的过渡效果（等于 cubic-bezier(0.42,0,1,1)）。 |
| ease-out | 规定以慢速结束的过渡效果（等于 cubic-bezier(0,0,0.58,1)）。 |
| ease-in-out | 规定以慢速开始和结束的过渡效果（等于 cubic-bezier(0.42,0,0.58,1)）。 |
| cubic-bezier(n,n,n,n) | 在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值。 |

实例 1

为了更好地理解不同的函数值，请看下面带有五个不同值的五个不同的 div 元素：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:50px;

background:red;

color:white;

font-weight:bold;

transition:width 2s;

-moz-transition:width 2s; /\* Firefox 4 \*/

-webkit-transition:width 2s; /\* Safari and Chrome \*/

-o-transition:width 2s; /\* Opera \*/

}

#div1 {transition-timing-function: linear;}

#div2 {transition-timing-function: ease;}

#div3 {transition-timing-function: ease-in;}

#div4 {transition-timing-function: ease-out;}

#div5 {transition-timing-function: ease-in-out;}

/\* Firefox 4: \*/

#div1 {-moz-transition-timing-function: linear;}

#div2 {-moz-transition-timing-function: ease;}

#div3 {-moz-transition-timing-function: ease-in;}

#div4 {-moz-transition-timing-function: ease-out;}

#div5 {-moz-transition-timing-function: ease-in-out;}

/\* Safari and Chrome: \*/

#div1 {-webkit-transition-timing-function: linear;}

#div2 {-webkit-transition-timing-function: ease;}

#div3 {-webkit-transition-timing-function: ease-in;}

#div4 {-webkit-transition-timing-function: ease-out;}

#div5 {-webkit-transition-timing-function: ease-in-out;}

/\* Opera: \*/

#div1 {-o-transition-timing-function: linear;}

#div2 {-o-transition-timing-function: ease;}

#div3 {-o-transition-timing-function: ease-in;}

#div4 {-o-transition-timing-function: ease-out;}

#div5 {-o-transition-timing-function: ease-in-out;}

div:hover

{

width:300px;

}

</style>

</head>

<body>

<div id="div1" style="top:100px">linear</div>

<div id="div2" style="top:150px">ease</div>

<div id="div3" style="top:200px">ease-in</div>

<div id="div4" style="top:250px">ease-out</div>

<div id="div5" style="top:300px">ease-in-out</div>

<p>请把鼠标指针移动到红色的 div 元素上，就可以看到过渡效果。</p>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

</body>

</html>

实例 2

与上例相同，但通过 cubic-bezier 来规定速度曲线：

<!DOCTYPE html>

<html>

<head>

<style>

div

{

width:100px;

height:50px;

background:red;

color:white;

font-weight:bold;

transition:width 2s;

-moz-transition:width 2s; /\* Firefox 4 \*/

-webkit-transition:width 2s; /\* Safari and Chrome \*/

-o-transition:width 2s; /\* Opera \*/

}

#div1 {transition-timing-function: cubic-bezier(0,0,0.25,1);}

#div2 {transition-timing-function: cubic-bezier(0.25,0.1,0.25,1);}

#div3 {transition-timing-function: cubic-bezier(0.42,0,1,1);}

#div4 {transition-timing-function: cubic-bezier(0,0,0.58,1);}

#div5 {transition-timing-function: cubic-bezier(0.42,0,0.58,1);}

/\* Firefox 4: \*/

#div1 {-moz-transition-timing-function: cubic-bezier(0,0,0.25,1);}

#div2 {-moz-transition-timing-function: cubic-bezier(0.25,0.1,0.25,1);}

#div3 {-moz-transition-timing-function: cubic-bezier(0.42,0,1,1);}

#div4 {-moz-transition-timing-function: cubic-bezier(0,0,0.58,1);}

#div5 {-moz-transition-timing-function: cubic-bezier(0.42,0,0.58,1);}

/\* Safari and Chrome: \*/

#div1 {-webkit-transition-timing-function: cubic-bezier(0,0,0.25,1);}

#div2 {-webkit-transition-timing-function: cubic-bezier(0.25,0.1,0.25,1);}

#div3 {-webkit-transition-timing-function: cubic-bezier(0.42,0,1,1);}

#div4 {-webkit-transition-timing-function: cubic-bezier(0,0,0.58,1);}

#div5 {-webkit-transition-timing-function: cubic-bezier(0.42,0,0.58,1);}

/\* Opera: \*/

#div1 {-o-transition-timing-function: cubic-bezier(0,0,0.25,1);}

#div2 {-o-transition-timing-function: cubic-bezier(0.25,0.1,0.25,1);}

#div3 {-o-transition-timing-function: cubic-bezier(0.42,0,1,1);}

#div4 {-o-transition-timing-function: cubic-bezier(0,0,0.58,1);}

#div5 {-o-transition-timing-function: cubic-bezier(0.42,0,0.58,1);}

div:hover

{

width:300px;

}

</style>

</head>

<body>

<div id="div1" style="top:100px">linear</div>

<div id="div2" style="top:150px">ease</div>

<div id="div3" style="top:200px">ease-in</div>

<div id="div4" style="top:250px">ease-out</div>

<div id="div5" style="top:300px">ease-in-out</div>

<p>请把鼠标指针移动到红色的 div 元素上，就可以看到过渡效果。</p>

<p><b>注释：</b>本例在 Internet Explorer 中无效。</p>

</body>

</html>

# 第322章 CSS unicode-bidi 属性

重写文本：

<!DOCTYPE html>

<html>

<head>

<style>

div.ex1 {

direction: rtl;

unicode-bidi: bidi-override;

}

</style>

</head>

<body>

<h1>unicode-bidi 属性</h1>

<div>Some text. Default writing direction.</div>

<div class="ex1">Some text. Right-to-left direction.</div>

</body>

</html>

## 322.1 定义和用法

unicode-bidi 属性与 direction 属性一起使用，设置或返回是否应重写文本以支持同一文档中的多种语言。

默认值： normal

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS2

JavaScript 语法： object.style.unicodeBidi="bidi-override"

## 322.2 CSS 语法

unicode-bidi: normal|embed|bidi-override|initial|inherit;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 默认值。元素不会打开额外的嵌入级别。 |
| embed | 对于行内元素，此值将打开额外的嵌入级别。 |
| bidi-override | 对于行内元素，该值会创建一个覆盖；  对于块容器元素，该值将为不在另一个块容器元素内的行内级别的后代创建一个覆盖。 |
| isolate | 该元素与其同胞隔离。 |
| isolate-override |  |
| plaintext |  |
| initial | 将此属性设置为其默认值。参阅 initial。 |
| inherit | 从其父元素继承此属性。参阅 inherit。 |

# 第323章 CSS user-select 属性

防止选取 <div> 元素的文本：

<!DOCTYPE html>

<html>

<head>

<style>

div {

-webkit-user-select: none; /\* Safari \*/

-ms-user-select: none; /\* IE 10 and IE 11 \*/

user-select: none; /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>user-select 属性</h1>

<div>The text of this div element cannot be selected. If you double-click me, my text will not be highlighted.</div>

</body>

</html>

## 323.1 定义和用法

user-select 属性规定是否能选取元素的文本。

在 web 浏览器中，如果您在文本上双击，文本会被选取或高亮显示。此属性用于阻止这种行为。

默认值： auto

继承： 否

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.userSelect="none"

## 323.2 CSS 语法

user-select: auto|none|text|all;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。如果浏览器允许，则可以选择文本。 |
| none | 防止文本选取。 |
| text | 文本可被用户选取。 |
| all | 单击选取文本，而不是双击。 |

# 第324章 CSS vertical-align 属性

垂直对齐一幅图像：

<html>

<head>

<style type="text/css">

img.top {vertical-align:text-top}

img.bottom {vertical-align:text-bottom}

</style>

</head>

<body>

<p>

这是一幅<img class="top" border="0" src="/i/eg\_cute.gif" />位于段落中的图像。

</p>

<p>

这是一幅<img class="bottom" border="0" src="/i/eg\_cute.gif" />位于段落中的图像。

</p>

</body>

</html>

## 324.1 定义和用法

vertical-align 属性设置元素的垂直对齐方式。

说明

该属性定义行内元素的基线相对于该元素所在行的基线的垂直对齐。允许指定负长度值和百分比值。这会使元素降低而不是升高。在表单元格中，这个属性会设置单元格框中的单元格内容的对齐方式。

默认值： baseline

继承性： no

版本： CSS1

JavaScript 语法： object.style.verticalAlign="bottom"

324.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| baseline | 默认。元素放置在父元素的基线上。 |
| sub | 垂直对齐文本的下标。 |
| super | 垂直对齐文本的上标 |
| top | 把元素的顶端与行中最高元素的顶端对齐 |
| text-top | 把元素的顶端与父元素字体的顶端对齐 |
| middle | 把此元素放置在父元素的中部。 |
| bottom | 把元素的顶端与行中最低的元素的顶端对齐。 |
| text-bottom | 把元素的底端与父元素字体的底端对齐。 |
| length |  |
| % | 使用 "line-height" 属性的百分比值来排列此元素。允许使用负值。 |
| inherit | 规定应该从父元素继承 vertical-align 属性的值。 |

# 第325章 CSS visibility 属性

使 h2 元素不可见：

<html>

<head>

<style type="text/css">

h1.visible {visibility:visible}

h1.invisible {visibility:hidden}

</style>

</head>

<body>

<h1 class="visible">这是可见的标题</h1>

<h1 class="invisible">这是不可见的标题</h1>

</body>

</html>

## 325.1 定义和用法

visibility 属性规定元素是否可见。

提示：即使不可见的元素也会占据页面上的空间。请使用 "display" 属性来创建不占据页面空间的不可见元素。

说明

这个属性指定是否显示一个元素生成的元素框。这意味着元素仍占据其本来的空间，不过可以完全不可见。值 collapse 在表中用于从表布局中删除列或行。

默认值： visible

继承性： yes

版本： CSS2

JavaScript 语法： object.style.visibility="hidden"

## 325.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| visible | 默认值。元素是可见的。 |
| hidden | 元素是不可见的。 |
| collapse | 当在表格元素中使用时，此值可删除一行或一列，但是它不会影响表格的布局。被行或列占据的空间会留给其他内容使用。如果此值被用在其他的元素上，会呈现为 "hidden"。 |
| inherit | 规定应该从父元素继承 visibility 属性的值。 |

# 第326章 CSS white-space 属性

规定段落中的文本不进行换行：

<html>

<head>

<style type="text/css">

p

{

white-space: nowrap

}

</style>

</head>

<body>

<p>

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

这是一些文本。

</p>

</body>

</html>

## 326.1 定义和用法

white-space 属性设置如何处理元素内的空白。

这个属性声明建立布局过程中如何处理元素中的空白符。值 pre-wrap 和 pre-line 是 CSS 2.1 中新增的。

默认值： normal

继承性： yes

版本： CSS1

JavaScript 语法： object.style.whiteSpace="pre"

## 326.2 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。 |

# 第327章 CSS width 属性

设置段落的高度和宽度：

<html>

<head>

<style type="text/css">

img

{

width: 300px

}

</style>

</head>

<body>

<img src="/i/eg\_smile.gif" />

</body>

</html>

## 327.1 定义和用法

width 属性设置元素的宽度。

说明

这个属性定义元素内容区的宽度，在内容区外面可以增加内边距、边框和外边距。

行内非替换元素会忽略这个属性。

默认值： auto

继承性： no

版本： CSS1

JavaScript 语法： object.style.width="50px"

## 327.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认值。浏览器可计算出实际的宽度。 |
| length | 使用 px、cm 等单位定义宽度。 |
| % | 定义基于包含块（父元素）宽度的百分比宽度。 |
| inherit | 规定应该从父元素继承 width 属性的值。 |

# 第328章 CSS word-break 属性

在恰当的断字点进行换行：

<!DOCTYPE html>

<html>

<head>

<style>

p.test1

{

width:11em;

border:1px solid #000000;

word-break:hyphenate;

}

p.test2

{

width:11em;

border:1px solid #000000;

word-break:break-all;

}

</style>

</head>

<body>

<p class="test1">This is a veryveryveryveryveryveryveryveryveryvery long paragraph.</p>

<p class="test2">This is a veryveryveryveryveryveryveryveryveryvery long paragraph.</p>

<p><b>注释：</b>目前 Opera 不支持 word-break 属性。</p>

</body>

</html>

## 328.1 定义和用法

word-break 属性规定自动换行的处理方法。

提示：通过使用 word-break 属性，可以让浏览器实现在任意位置的换行。

默认值： normal

继承性： yes

版本： CSS3

JavaScript 语法： object.style.wordBreak="keep-all"

## 328.2 语法

word-break: normal|break-all|keep-all;

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 使用浏览器默认的换行规则。 |
| break-all | 允许在单词内换行。 |
| keep-all | 只能在半角空格或连字符处换行。 |

# 第329章 CSS word-spacing 属性

规定段落中的字间距是 25 像素：

<html>

<head>

<style type="text/css">

p.spread {word-spacing: 30px;}

p.tight {word-spacing: -0.5em;}

</style>

</head>

<body>

<p class="spread">This is some text. This is some text.</p>

<p class="tight">This is some text. This is some text.</p>

</body>

</html>

## 329.1 定义

word-spacing 属性增加或减少单词间的空白（即字间隔）。

该属性定义元素中字之间插入多少空白符。针对这个属性，“字” 定义为由空白符包围的一个字符串。如果指定为长度值，会调整字之间的通常间隔；所以，normal 就等同于设置为 0。允许指定负长度值，这会让字之间挤得更紧。

注释：允许使用负值。

默认值： normal

继承性： yes

版本： CSS1

JavaScript 语法： object.style.wordSpacing="10px"

## 329.2 提示和注释

注释：CSS 把“字（word）”定义为任何非空白符字符组成的串，并由某种空白字符包围。这个定义没有实际的语义，它只是假设一个文档包含由一个或多个空白字符包围的字。支持 CSS 的用户代理不一定能确定一个给定语言中哪些是合法的字，而哪些不是。尽管这个定义没有多大价值，不过它意味着采用象形文字的语言或非罗马书写体往往无法指定字间隔。

提示：利用这个属性，可能会创建字间隔太宽的文档，所以，使用 word-spacing 时要小心。

## 329.3 可能的值

|  |  |
| --- | --- |
| 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。 |

# 第330章 CSS word-wrap 属性

允许长单词换行到下一行：

<!DOCTYPE html>

<html>

<head>

<style>

p.test

{

width:11em;

border:1px solid #000000;

word-wrap:break-word;

}

</style>

</head>

<body>

<p class="test">This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.</p>

</body>

</html>

## 330.1 定义和用法

word-wrap 属性允许长单词或 URL 地址换行到下一行。

默认值： normal

继承性： yes

版本： CSS3

JavaScript 语法： object.style.wordWrap="break-word"

## 330.2 语法

word-wrap: normal|break-word;

|  |  |
| --- | --- |
| 值 | 描述 |
| normal | 只在允许的断字点换行（浏览器保持默认处理）。 |
| break-word | 在长单词或 URL 地址内部进行换行。 |

# 第331章 CSS writing-mode 属性

规定水平还是垂直地排布文本行：

<!DOCTYPE html>

<html>

<head>

<style>

p.test1 {

writing-mode: horizontal-tb;

}

p.test2 {

writing-mode: vertical-rl;

}

span.test2 {

writing-mode: vertical-rl;

}

</style>

</head>

<body>

<h1>writing-mode 属性</h1>

<p class="test1">Some text with default writing-mode.</p>

<p>Some text with a span element with a <span class="test2">vertical-rl</span> writing-mode.</p>

<p class="test2">Some text with writing-mode: vertical-rl.</p>

</body>

</html>

## 331.1 定义和用法

writing-mode 属性规定水平还是垂直地排布文本行。

默认值： horizontal-tb

继承： 是

动画制作： 不支持。请参阅：动画相关属性。

版本： CSS3

JavaScript 语法： object.style.writingMode="vertical-rl"

## 331.2 CSS 语法

writing-mode: horizontal-tb|vertical-rl|vertical-lr;

属性值

|  |  |
| --- | --- |
| 值 | 描述 |
| horizontal-tb | 让内容从左到右水平流动，从上到下垂直流动。 |
| vertical-rl | 让内容从上到下垂直流动，从右到左水平流动。 |
| vertical-lr | 让内容从上到下垂直流动，从左到右水平流动。 |

# 第332章 CSS z-index 属性

设置图像的 z-index：

<html>

<head>

<style type="text/css">

img

{

position:absolute;

left:0px;

top:0px;

z-index:-1;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<img src="/i/eg\_smile.gif" />

<p>由于图像的 z-index 是 -1，因此它在文本的后面出现。</p>

</body>

</html>

## 332.1 定义和用法

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;）！

说明

该属性设置一个定位元素沿 z 轴的位置，z 轴定义为垂直延伸到显示区的轴。如果为正数，则离用户更近，为负数则表示离用户更远。

默认值： auto

继承性： no

版本： CSS2

JavaScript 语法： object.style.zIndex="1"

## 332.2 可能的值

|  |  |
| --- | --- |
| 值 | 描述 |
| auto | 默认。堆叠顺序与父元素相等。 |
| number | 设置元素的堆叠顺序。 |
| inherit | 规定应该从父元素继承 z-index 属性的值。 |

# 第333章 CSS 简介

## 333.1 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

如果你希望首先学习这些项目，请在 首页 访问相关教程。

## 333.2 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

## 333.3 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。

## 333.4 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。

## 333.5 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。

## 333.6 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。

浏览器缺省设置

外部样式表

内部样式表（位于 <head> 标签内部）

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）。

# 第334章 CSS 基础语法

## 334.1 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。

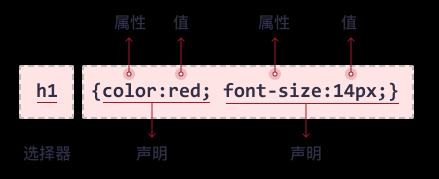
selector {property: value}

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;}

下面的示意图为您展示了上面这段代码的结构：



## 334.2 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。

## 334.3 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";}

## 334.4 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;}

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;

}

## 334.5 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

## 335.1 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;

}

## 335.2 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？

## 335.3 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。

## 335.4 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;

}

# 第336章 CSS 派生选择器

## 336.1 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>

# 第337章 CSS id 选择器

## 337.1 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。

## 337.2 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 <em></em> 或者 <span></span>，不过这样的用法是非法的，因为不可以在内联元素 <span> 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。

## 337.3 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;

}

# 第338章 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）。

# 第339章 如何创建 CSS

## 339.1 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：

### 339.1.1 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。

### 339.1.2 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>

### 339.1.3 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>

## 339.2 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。

# 第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。

## 340.1 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;}

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。

## 340.2 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：

1. radio {background-image: url(/i/eg\_bg\_07.gif);}

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。

## 340.3 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;

}

## 340.4 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。

## 340.5 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。

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

}

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。

|  |  |
| --- | --- |
| 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。 |

# 第341章 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 increased.

</p>

<p class="tight">

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

</p>

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>

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。

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。

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位置。

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。

|  |  |
| --- | --- |
| 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。 |

# 第342章 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

如果需要了解更多有关字体系列的知识，请阅读 CSS 字体系列。

指定字体系列

使用 font-family 属性 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;}

亲自试一试

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;}

亲自试一试

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;}

亲自试一试

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;}

亲自试一试

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>

亲自试一试

字体风格

font-style 属性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;}

亲自试一试

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

字体加粗

font-weight 属性设置文本的粗细。

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

字体大小

font-size 属性设置文本的大小。

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;}

亲自试一试

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 \*/

亲自试一试

在上面的例子中，以 em 为单位的文本大小与前一个例子中以像素计的文本是相同的。不过，如果使用 em 单位，则可以在所有浏览器中调整文本大小。

不幸的是，在 IE 中仍存在问题。在重设文本大小时，会比正常的尺寸更大或更小。

结合使用百分比和 EM

在所有浏览器中均有效的方案是为 body 元素（父元素）以百分比设置默认的 font-size 值：

实例

body {font-size:100%;}

h1 {font-size:3.75em;}

h2 {font-size:2.5em;}

p {font-size:0.875em;}

亲自试一试

我们的代码非常有效。在所有浏览器中，可以显示相同的文本大小，并允许所有浏览器缩放文本的大小。

|  |  |
| --- | --- |
| 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。 |

# 第343章 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}

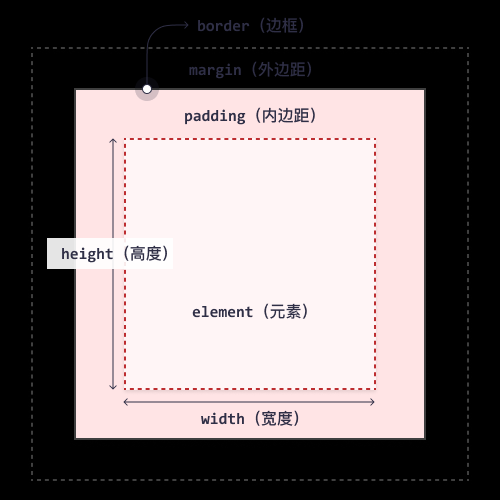
list-style 的值可以按任何顺序列出，而且这些值都可以忽略。只要提供了一个值，其它的就会填入其默认值。

|  |  |
| --- | --- |
| 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 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。

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：

\* {

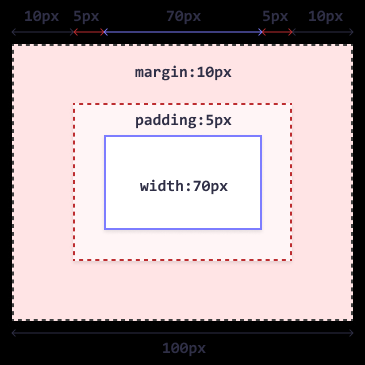
margin: 0;

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。

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：）

# 第345章 框模型：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。

|  |  |
| --- | --- |
| 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。 |

# 第346章 框模型：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;}

亲自试一试

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。

|  |  |
| --- | --- |
| 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。 |

# 第347章 框模型：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。

|  |  |
| --- | --- |
| 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。 |

# 第348章 框模型：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。

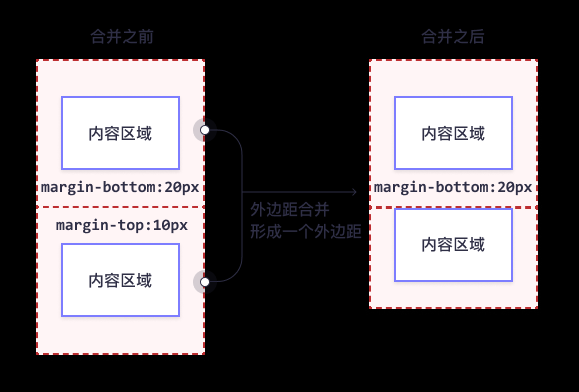
合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

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：



<html>

<head>

<style type="text/css">

\* {

margin:0;

padding:0;

border:0;

}

#d1 {

width:100px;

height:100px;

margin-top:20px;

margin-bottom:20px;

background-color:red;

}

#d2 {

width:100px;

height:100px;

margin-top:10px;

background-color:blue;

}

</style>

</head>

<body>

<div id="d1">

</div>

<div id="d2">

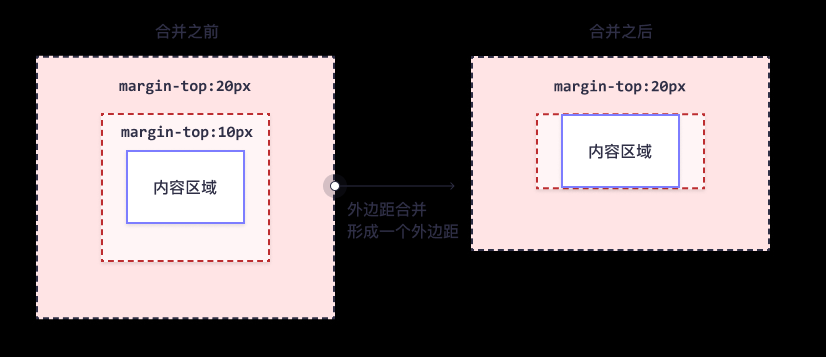
</div>

<p>请注意，两个 div 之间的外边距是 20px，而不是 30px（20px + 10px）。</p>

</body>

</html>

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：



<html>

<head>

<style type="text/css">

\* {

margin:0;

padding:0;

border:0;

}

#outer {

width:300px;

height:300px;

background-color:red;

margin-top:20px;

}

#inner {

width:50px;

height:50px;

background-color:blue;

margin-top:10px;

}

</style>

</head>

<body>

<div id="outer">

<div id="inner">

</div>

</div>

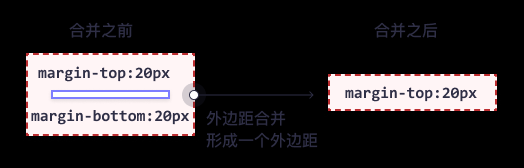
<p>注释：请注意，如果不设置 div 的内边距和边框，那么内部 div 的上外边距将与外部 div 的上外边距合并（叠加）。</p>

</body>

</html>

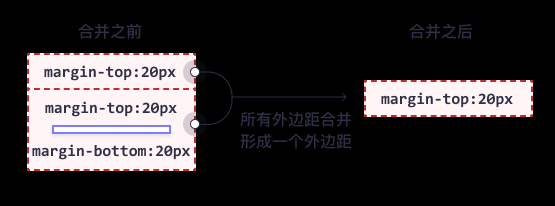
尽管看上去有些奇怪，但是外边距甚至可以与自身发生合并。

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：



CSS 外边距合并实例 3

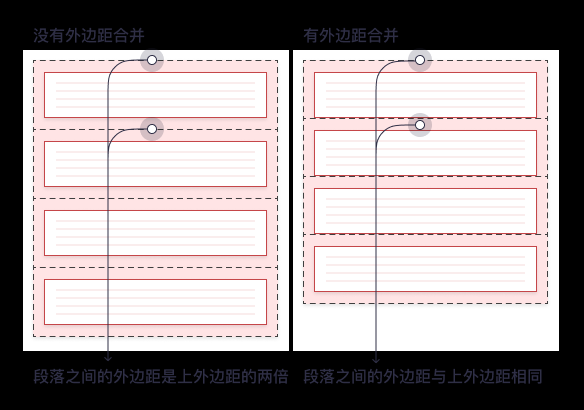
如果这个外边距遇到另一个元素的外边距，它还会发生合并：



CSS 外边距合并实例 4

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。

# 第350章 CSS 定位

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 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。

|  |  |
| --- | --- |
|  |  |
| 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。 |

# 第351章 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 {

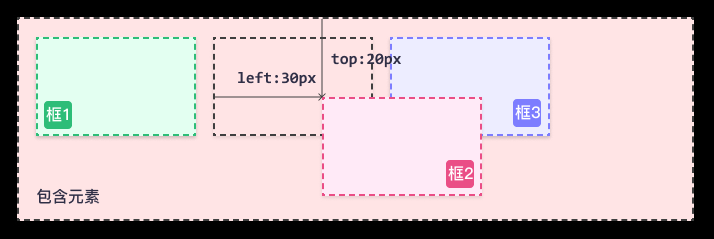
position: relative;

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。

# 第352章 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 {

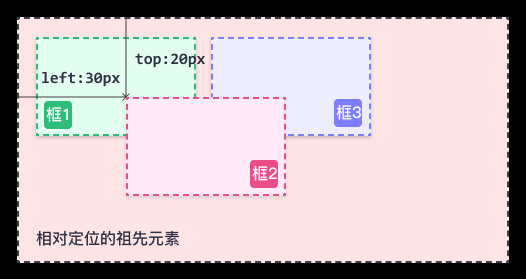
position: absolute;

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。

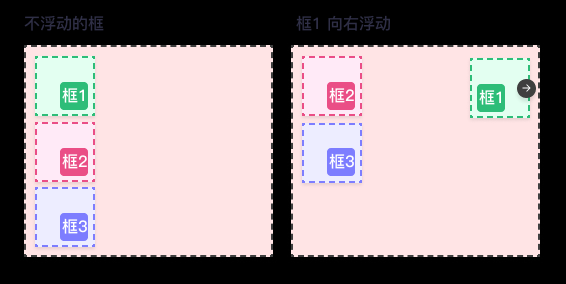
# 第353章 CSS 浮动

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。

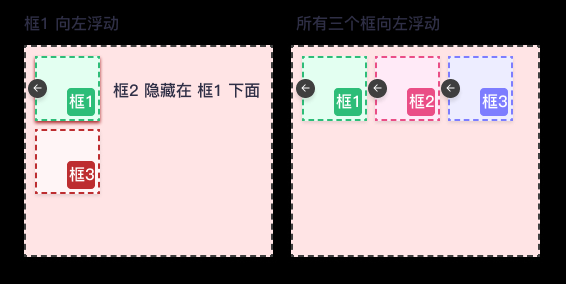
CSS 浮动

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：

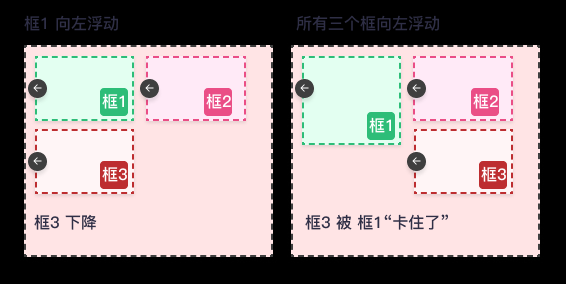


再请看下图，当框 1 向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框 2，使框 2 从视图中消失。

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

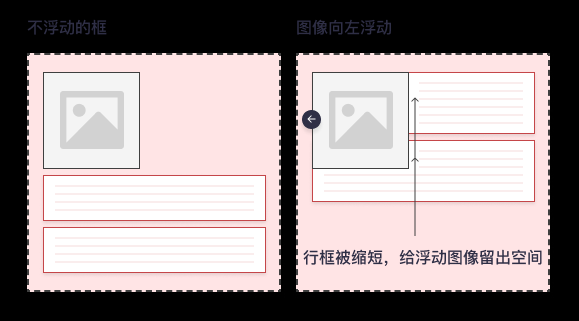
在 CSS 中，我们通过 float 属性实现元素的浮动。

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

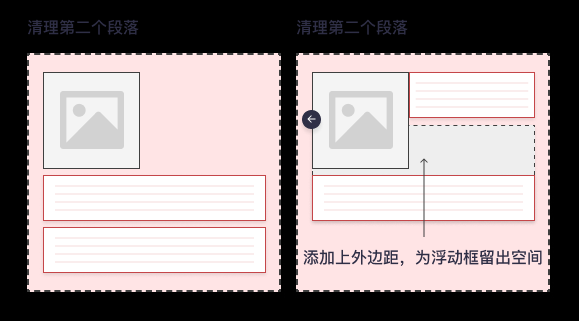
浮动框旁边的行框被缩短，从而给浮动框留出空间，行框围绕浮动框。

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">

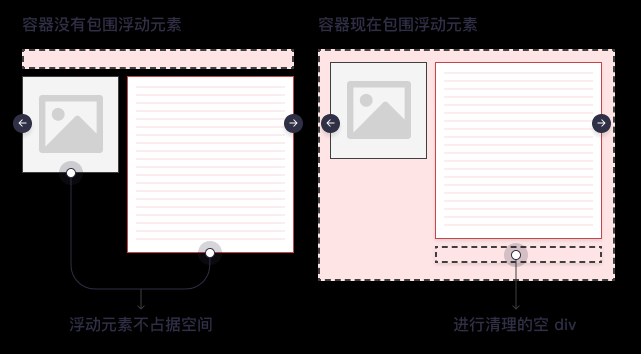
<img src="news-pic.jpg" />

<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。

# 第354章CSS 元素选择器

CSS 元素选择器

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

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

html {color:black;}

h1 {color:blue;}

h2 {color:silver;}

亲自试一试

可以将某个样式从一个元素切换到另一个元素。

假设您决定将上面的段落文本（而不是 h1 元素）设置为灰色。只需要把 h1 选择器改为 p：

html {color:black;}

p {color:gray;}

h2 {color:silver;}

亲自试一试

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;}

因此，我们也可以为 XML 文档中的元素设置样式：

XML 文档：

<?xml version="1.0" encoding="ISO-8859-1"?>

<?xml-stylesheet type="text/css" href="note.css"?>

<note>

<to>George</to>

<from>John</from>

<heading>Reminder</heading>

<body>Don't forget the meeting!</body>

</note>

CSS 文档：

note

{

font-family:Verdana, Arial;

margin-left:30px;

}

to

{

font-size:28px;

display: block;

}

from

{

font-size:28px;

display: block;

}

heading

{

color: red;

font-size:60px;

display: block;

}

body

{

color: blue;

font-size:35px;

display: block;

}

查看效果

通过上面的例子，您可以看到，CSS 元素选择器（类型选择器）可以设置 XML 文档中元素的样式。

如果您需要更多关于为 XML 文档添加样式的知识，请访问 w3school 的 XML 教程。

# 第355章 CSS 选择器分组

选择器分组

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

h2, p {color:gray;}

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

可以将任意多个选择器分组在一起，对此没有任何限制。

例如，如果您想把很多元素显示为灰色，可以使用类似如下的规则：

body, h2, p, table, th, td, pre, strong, em {color:gray;}

提示：通过分组，创作者可以将某些类型的样式“压缩”在一起，这样就可以得到更简洁的样式表。

以下的两组规则能得到同样的结果，不过可以很清楚地看出哪一个写起来更容易：

/\* no grouping \*/

h1 {color:blue;}

h2 {color:blue;}

h3 {color:blue;}

h4 {color:blue;}

h5 {color:blue;}

h6 {color:blue;}

/\* grouping \*/

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

亲自试一试

分组提供了一些有意思的选择。例如，下例中的所有规则分组都是等价的，每个组只是展示了对选择器和声明分组的不同方法：

/\* group 1 \*/

h1 {color:silver; background:white;}

h2 {color:silver; background:gray;}

h3 {color:white; background:gray;}

h4 {color:silver; background:white;}

b {color:gray; background:white;}

/\* group 2 \*/

h1, h2, h4 {color:silver;}

h2, h3 {background:gray;}

h1, h4, b {background:white;}

h3 {color:white;}

b {color:gray;}

/\* group 3 \*/

h1, h4 {color:silver; background:white;}

h2 {color:silver;}

h3 {color:white;}

h2, h3 {background:gray;}

b {color:gray; background:white;}

亲自试一试：

group 1

group 2

group 3

请注意，group 3 中使用了“声明分组”。稍后我们会为您介绍“声明分组”。

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;}

亲自试一试

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。

假设您希望所有 h1 元素都有红色背景，并使用 28 像素高的 Verdana 字体显示为蓝色文本，可以写以下样式：

h1 {font: 28px Verdana;}

h1 {color: blue;}

h1 {background: red;}

但是上面这种做法的效率并不高。尤其是当我们为一个有多个样式的元素创建这样一个列表时会很麻烦。相反，我们可以将声明分组在一起：

h1 {font: 28px Verdana; color: white; background: black;}

这与前面的 3 行样式表的效果完全相同。

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;

}

亲自试一试

怎么样，上面这种写法的可读性是不是更强。

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

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。

结合选择器和声明的分组

我们可以在一个规则中结合选择器分组和声明分组，就可以使用很少的语句定义相对复杂的样式。

下面的规则为所有标题指定了一种复杂的样式：

h1, h2, h3, h4, h5, h6 {

color:gray;

background: white;

padding: 10px;

border: 1px solid black;

font-family: Verdana;

}

亲自试一试

上面这条规则将所有标题的样式定义为带有白色背景的灰色文本，其内边距是 10 像素，并带有 1 像素的实心边框，文本字体是 Verdana。

# 第356章 CSS 类选择器详解

类选择器允许以一种独立于文档元素的方式来指定样式。

CSS 类选择器

类选择器允许以一种独立于文档元素的方式来指定样式。

该选择器可以单独使用，也可以与其他元素结合使用。

提示：只有适当地标记文档后，才能使用这些选择器，所以使用这两种选择器通常需要先做一些构想和计划。

要应用样式而不考虑具体设计的元素，最常用的方法就是使用类选择器。

修改 HTML 代码

在使用类选择器之前，需要修改具体的文档标记，以便类选择器正常工作。

为了将类选择器的样式与元素关联，必须将 class 指定为一个适当的值。请看下面的 HTML 代码：

<h1 class="important">

This heading is very important.

</h1>

<p class="important">

This paragraph is very important.

</p>

在上面的代码中，两个元素的 class 都指定为 important：第一个标题（ h1 元素），第二个段落（p 元素）。

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

结合元素选择器

类选择器可以结合元素选择器来使用。

例如，您可能希望只有段落显示为红色文本：

p.important {color:red;}

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

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

p.important {color:red;}

h1.important {color:blue;}

亲自试一试

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

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

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

.important.urgent {background:silver;}

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

<p class="important urgent warning">

This paragraph is a very important and urgent warning.

</p>

亲自试一试

重要事项：在 IE7 之前的版本中，不同平台的 Internet Explorer 都不能正确地处理多类选择器。

# 第357章 CSS ID 选择器详解

ID 选择器允许以一种独立于文档元素的方式来指定样式。

CSS ID 选择器

在某些方面，ID 选择器类似于类选择器，不过也有一些重要差别。

语法

首先，ID 选择器前面有一个 # 号 - 也称为棋盘号或井号。

请看下面的规则：

\*#intro {font-weight:bold;}

与类选择器一样，ID 选择器中可以忽略通配选择器。前面的例子也可以写作：

#intro {font-weight:bold;}

这个选择器的效果将是一样的。

第二个区别是 ID 选择器不引用 class 属性的值，毫无疑问，它要引用 id 属性中的值。

以下是一个实际 ID 选择器的例子：

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

亲自试一试

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>

亲自试一试：

为 id 为 mostImportant 的 h1 元素设定样式

为 id 为 mostImportant 的 em 元素设定样式

为 id 为 mostImportant 的 ul 元素设定样式

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。

# 第358章 CSS 属性选择器详解

CSS 2 引入了属性选择器。

属性选择器可以根据元素的属性及属性值来选择元素。

简单属性选择

如果希望选择有某个属性的元素，而不论属性值是什么，可以使用简单属性选择器。

例子 1

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

例子 2

与上面类似，可以只对有 href 属性的锚（a 元素）应用样式：

a[href] {color:red;}

亲自试一试

例子 3

还可以根据多个属性进行选择，只需将属性选择器链接在一起即可。

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

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

亲自试一试

例子 4

可以采用一些创造性的方法使用这个特性。

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

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

亲自试一试

提示：上面这个特例更适合用来诊断而不是设计，即用来确定图像是否确实有效。

例子 5：为 XML 文档使用属性选择器

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

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

planet[moons] {color:red;}

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

<planet>Venus</planet>

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

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

查看效果

根据具体属性值选择

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

例子 1

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

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

亲自试一试

例子 2

与简单属性选择器类似，可以把多个属性-值选择器链接在一起来选择一个文档。

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

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

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

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

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

亲自试一试

例子 3

同样地，XML 语言也可以利用这种方法来设置样式。

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

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

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

<planet>Venus</planet>

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

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

查看效果

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

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

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

子串匹配属性选择器

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

很多现代浏览器都支持这些选择器，包括 IE7。

下表是对这些选择器的简单总结：

类型 描述

[abc^="def"] 选择 abc 属性值以 "def" 开头的所有元素

[abc$="def"] 选择 abc 属性值以 "def" 结尾的所有元素

[abc\*="def"] 选择 abc 属性值中包含子串 "def" 的所有元素

可以想到，这些选择有很多用途。

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

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

亲自试一试

提示：任何属性都可以使用这些选择器。

特定属性选择类型

最后为您介绍特定属性选择器。请看下面的例子：

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

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

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

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

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

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

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

亲自试一试

一般来说，[att|="val"] 可以用于任何属性及其值。

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

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

亲自试一试

当然，这种属性选择器最常见的用途还是匹配语言值。

CSS 选择器参考手册

选择器 描述

[attribute] 用于选取带有指定属性的元素。

[attribute=value] 用于选取带有指定属性和值的元素。

[attribute~=value] 用于选取属性值中包含指定词汇的元素。

[attribute|=value] 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] 匹配属性值以指定值开头的每个元素。

[attribute$=value] 匹配属性值以指定值结尾的每个元素。

[attribute\*=value] 匹配属性值中包含指定值的每个元素。

# 第359章 CSS 后代选择器

后代选择器（descendant selector）又称为包含选择器。

后代选择器可以选择作为某元素后代的元素。

根据上下文选择元素

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

举例来说，如果您希望只对 h1 元素中的 em 元素应用样式，可以这样写：

h1 em {color:red;}

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

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

语法解释

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

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

具体应用

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

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

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

div.sidebar {background:blue;}

div.maincontent {background:white;}

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

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

有关后代选择器有一个易被忽视的方面，即两个元素之间的层次间隔可以是无限的。

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

因此，ul em 将会选择以下标记中的所有 em 元素：

<ul>

<li>List item 1

<ol>

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

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

<li>List item 1-3

<ol>

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

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

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

</ol>

</li>

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

</ol>

</li>

<li>List item 2</li>

<li>List item 3</li>

</ul>

# 第360章 CSS 子元素选择器

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

选择子元素

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

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

h1 > strong {color:red;}

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

<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

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。

# 第361章 CSS 相邻兄弟选择器

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

选择相邻兄弟

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

例如，如果要增加紧接在 h1 元素后出现的段落的上边距，可以这样写：

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

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

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

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。