# 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！

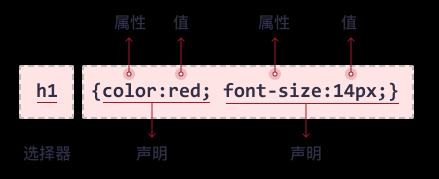
样式定义通常保存在外部 .css 文件中。

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。

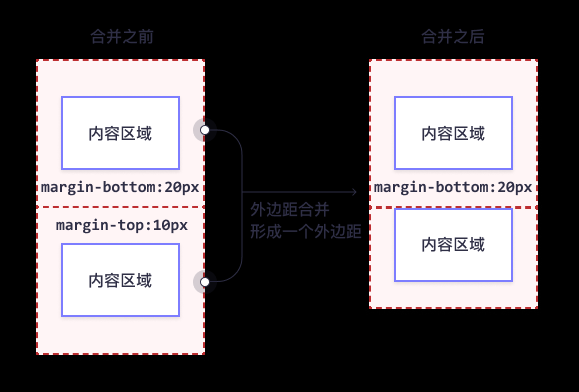
合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

## 23.1 外边距合并

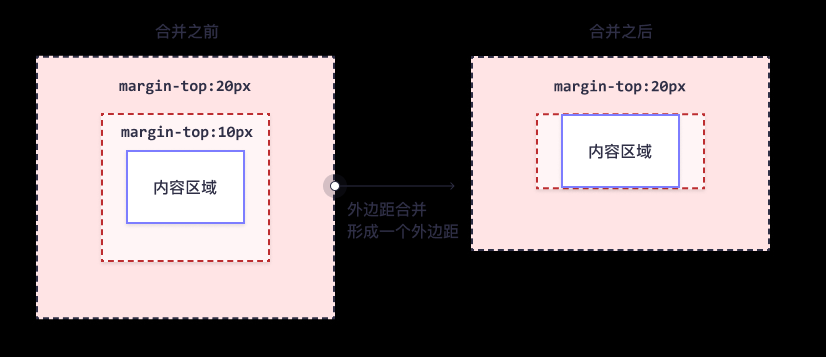
外边距合并（叠加）是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。

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：

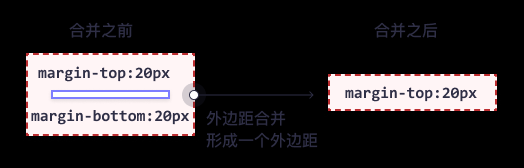


当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并。请看下图：

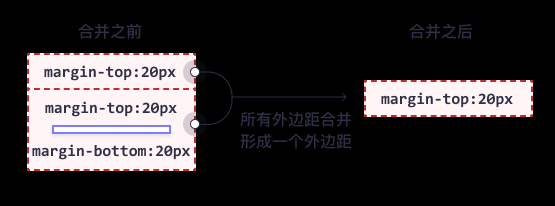


尽管看上去有些奇怪，但是外边距甚至可以与自身发生合并。

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：

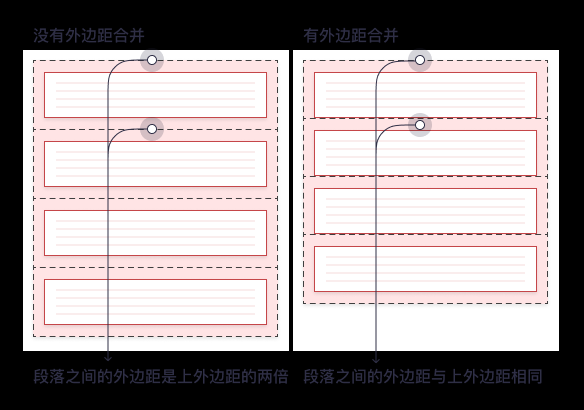


如果这个外边距遇到另一个元素的外边距，它还会发生合并：



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

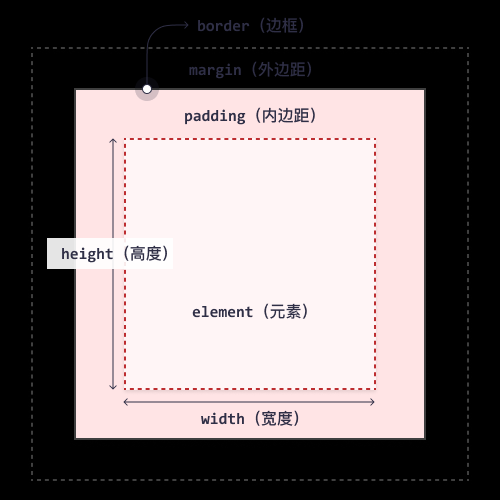
|  |  |
| --- | --- |
| 属性 | 描述 |
| height | 设置元素的高度。 |
| max-height | 设置元素的最大高度。 |
| max-width | 设置元素的最大宽度。 |
| min-height | 设置元素的最小高度。 |
| min-width | 设置元素的最小宽度。 |
| width | 设置元素的宽度。 |

# 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：

\* {

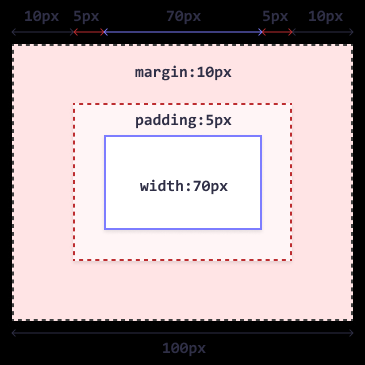
margin: 0;

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