CSS3 教程

CSS 用于控制网页的样式和布局。
CSS3 是最新的 CSS 标准。

本教程向您讲解 CSS3 中的新特性。

开始学习 CSS3!

CSS3 Transforms

CSS3 实例

```
div
{
transform:rotate(30deg);
}
```

尝试一下 »

点击 "尝试一下" 按钮查看在线实例。

CSS3 参考手册

在菜鸟教程中，我们提供完整的 CSS3 参考手册，包括所有属性和选择器的语法、实例、浏览器支持信息。

CSS 属性参考手册

CSS3 浏览器支持情况

CSS 选择器参考手册

CSS 颜色参考手册

首页 HTML CSS JS 本地书签

CSS3 简介

对CSS3已完全向后兼容，所以你就不必改变现有的设计。浏览器将永远支持CSS2。

CSS3 模块

CSS3被拆分为"模块"。旧规范已拆分成小块，还增加了新的。

一些最重要CSS3模块如下：

- 选择器
- 盒模型
- 背景和边框
- 文字特效
- 2D/3D转换
- 动画
- 多列布局
- 用户界面

CSS3 建议

W3C的CSS3规范仍在开发。

但是，许多新的CSS3属性已在现代浏览器使用。

CSS3 教程 CSS3 边框

点我分享笔记

反馈/建议

CSS3 简介 CSS3 背景

CSS3 边框

CSS3 边框

用 CSS3，你可以创建圆角边框，添加阴影框，并作为边界的形象而不使用设计程序，如 Photoshop。

在本章中，您将了解以下的边框属性：

- border-radius
- box-shadow
- border-image

CSS3 圆角

在 CSS2 中添加圆角棘手。我们不得不在每个角落使用不同的图像。

在 CSS3 中，很容易创建圆角。

在 CSS3 中 border-radius 属性被用于创建圆角：

这是圆角边框！

实例

在div中添加圆角元素：

```
div
{
border:2px solid;
border-radius:25px;
}
```

尝试一下 »

CSS3 盒阴影

CSS3 中的 box-shadow 属性被用来添加阴影:

实例

在div中添加box-shadow属性

```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

尝试一下 »

CSS3 边界图片

有了 CSS3 的 border-image 属性，你可以使用图像创建一个边框：

border-image 属性允许你指定一个图片作为边框！ 用于创建上文边框的原始图像：

在 div 中使用图片创建边框:

Border

实例

在 div 中使用图片创建边框

```
div
{
border-image:url(border.png) 30 30 round;
-webkit-border-image:url(border.png) 30 30 round; /* Safari 5 and older */
-o-border-image:url(border.png) 30 30 round; /* Opera */
}
```

尝试一下 »

新边框属性

属性	说明	CSS
border-image	设置所有边框图像的速记属性。	3
border-radius	一个用于设置所有四个边框- *-半径属性的速记属性	3

box-shadow	附加一个或多个下拉框的阴影	3

CSS3 简介 CSS3 背景

点我分享笔记

反馈/建议

CSS3 渐变 CSS 图片

CSS3 圆角

CSS3 圆角

使用 CSS3 border-radius 属性，你可以给任何元素制作 "圆角"。

CSS3 圆角制作器

浏览器支持

表格中的数字表示支持该属性的第一个浏览器的版本号。

-webkit- 或 -moz- 前面的数字表示支持该前缀的第一个版本。

属性					
border-radius	9.0	5.0 4.0 -webkit-	4.0 3.0 -moz-	5.0 3.1 -webkit-	10.5

CSS3 border-radius 属性

使用 CSS3 border-radius 属性，你可以给任何元素制作 "圆角"。

以下为三个实例：

1. 指定背景颜色的元素圆角：

圆角！

2. 指定边框的元素圆角:

圆角！

3. 指定背景图片的元素圆角:

圆角！

代码如下:

实例

```
#rcorners1 {
    border-radius: 25px;
    background: #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}

#rcorners2 {
    border-radius: 25px;
    border: 2px solid #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}

#rcorners3 {
    border-radius: 25px;
    background: url(paper.gif);
    background-position: left top;
    background-repeat: repeat;
    padding: 20px;
    width: 200px;
    height: 150px;
}
```

尝试一下 »

CSS3 border-radius - 指定每个圆角

如果你在 border-radius 属性中只指定一个值，那么将生成 4 个 圆角。

但是，如果你要在四个角上一一指定，可以使用以下规则：

四个值: 第一个值为左上角，第二个值为右上角，第三个值为右下角，第四个值为左下角。

三个值: 第一个值为左上角, 第二个值为右上角和左下角，第三个值为右下角

两个值: 第一个值为左上角与右下角，第二个值为右上角与左下角

一个值： 四个圆角值相同

以下为三个实例:

1. 四个值 - border-radius: 15px 50px 30px 5px:

2. 三个值 - border-radius: 15px 50px 30px:

3. 两个值 - border-radius: 15px 50px:

以下为源代码:

实例

```
#rcorners4 {
    border-radius: 15px 50px 30px 5px;
    background: #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}

#rcorners5 {
    border-radius: 15px 50px 30px;
    background: #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}

#rcorners6 {
    border-radius: 15px 50px;
    background: #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}
```

尝试一下 »

您还可以创建椭圆边角:

实例

```
#rcorners7 {
    border-radius: 50px/15px;
    background: #8AC007;
    padding: 20px;
    width: 200px;
```

```
    height: 150px;
}

#rcorners8 {
    border-radius: 15px/50px;
    background: #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}

#rcorners9 {
    border-radius: 50%;
    background: #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}
```

尝试一下 »

CSS3 圆角属性

属性	描述
border-radius	所有四个边角 border-*-*-radius 属性的缩写
border-top-left-radius	定义了左上角的弧度
border-top-right-radius	定义了右上角的弧度
border-bottom-right-radius	定义了右下角的弧度
border-bottom-left-radius	定义了左下角的弧度

CSS3 渐变 CSS 图片

点我分享笔记

反馈/建议

CSS3 边框 CSS3 文本效果

CSS3 背景

CSS3 背景

CSS3中包含几个新的背景属性，提供更大背景元素控制。

在本章您将了解以下背景属性：

- background-image
- background-size
- background-origin
- background-clip

您还将学习如何使用多重背景图像。

浏览器支持

表格中的数字表示支持该属性的第一个浏览器版本号。

紧跟在 -webkit-, -ms- 或 -moz- 前的数字为支持该前缀属性的第一个浏览器版本号。

属性					
background-image (with multiple backgrounds)	4.0	9.0	3.6	3.1	11.5
background-size	4.0 1.0 -webkit-	9.0	4.0 3.6 -moz-	4.1 3.0 -webkit-	10.5 10.0 -o-
background-origin	1.0	9.0	4.0	3.0	10.5
background-clip	4.0	9.0	4.0	3.0	10.5

CSS3 background-image属性

CSS3中可以通过background-image属性添加背景图片。

不同的背景图像和图像用逗号隔开，所有的图片中显示在最顶端的为第一张。

实例

```
#example1 {
background-image: url(img_flwr.gif), url(paper.gif);
background-position: right bottom, left top;
background-repeat: no-repeat, repeat;
}
```

尝试一下 »

可以给不同的图片设置多个不同的属性

实例

```
#example1 {
background: url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;
}
```

尝试一下 »

CSS3 background-size 属性

background-size指定背景图像的大小。CSS3以前，背景图像大小由图像的实际大小决定。

CSS3中可以指定背景图片，让我们重新在不同的环境中指定背景图片的大小。您可以指定像素或百分比大小。

你指定的大小是相对于父元素的宽度和高度的百分比的大小。

实例 1

重置背景图像：

```
div
{
background:url(img_flwr.gif);
background-size:80px 60px;
background-repeat:no-repeat;
}
```

尝试一下 »

实例 2

伸展背景图像完全填充内容区域：

```
div
{
background:url(img_flwr.gif);
background-size:100% 100%;
background-repeat:no-repeat;
}
```

尝试一下 »

CSS3的background-Origin属性

background-Origin属性指定了背景图像的位置区域。

content-box, padding-box,和 border-box区域内可以放置背景图像。

实例

在 content-box 中定位背景图片：

```
div
{
background:url(img_flwr.gif);
background-repeat:no-repeat;
background-size:100% 100%;
background-origin:content-box;
}
```

尝试一下 »

CSS3 多个背景图像

CSS3 允许你在元素上添加多个背景图像。

实例

在 body 元素中设置两个背景图像：

```
body
{
background-image:url(img_flwr.gif),url(img_tree.gif);
}
```

尝试一下 »

CSS3 background-clip属性

CSS3中background-clip背景剪裁属性是从指定位置开始绘制。

实例

```
#example1 {
border: 10px dotted black;
padding: 35px;
background: yellow;
background-clip: content-box;
}
```

尝试一下 »

新的背景属性

顺序	描述	CSS
background-clip	规定背景的绘制区域。	3
background-origin	规定背景图片的定位区域。	3
background-size	规定背景图片的尺寸。	3

CSS3 边框 CSS3 文本效果

1 篇笔记
#1

写笔记

补充：CSS3 多重背景(multiple backgrounds)
多重背景，也就是CSS2里background的属性外加origin、clip和size组成的新background的多次叠加，缩写时为用逗号隔开的每组值；用分解写法时，如果有多个背景图片，而其他属性只有一个（例如background-repeat只有一个），表明所有背景图片应用该属性值。
语法缩写如下：

```
background :  [background-color] | [background-image] | [background-position][/background-size] | [background-repeat] | [background-attachment] | [background-clip] | [background-origin],...
```

可以把上面的缩写拆解成以下形式：

```
background-image:url1,url2,...,urlN;

background-repeat : repeat1,repeat2,...,repeatN;

backround-position : position1,position2,...,positionN;

background-size : size1,size2,...,sizeN;

background-attachment : attachment1,attachment2,...,attachmentN;

background-clip : clip1,clip2,...,clipN;

background-origin : origin1,origin2,...,originN;

background-color : color;
```

注意：
用逗号隔开每组 background 的缩写值；
如果有 size 值，需要紧跟 position 并且用 "/" 隔开；
如果有多个背景图片，而其他属性只有一个（例如 background-repeat 只有一个），表明所有背景图片应用该属性值。
background-color 只能设置一个。

尝试一下 »

kalevi1年前 (2017-09-20)

反馈/建议

CSS3 渐变（Gradients）

线性渐变

CSS3 渐变（gradients）可以让你在两个或多个指定的颜色之间显示平稳的过渡。

以前，你必须使用图像来实现这些效果。但是，通过使用 CSS3 渐变（gradients），你可以减少下载的时间和宽带的使用。此外，渐变效果的元素在放大时看起来效果更好，因为渐变（gradient）是由浏览器生成的。

CSS3 定义了两种类型的渐变（gradients）：

线性渐变（Linear Gradients）- 向下/向上/向左/向右/对角方向

径向渐变（Radial Gradients）- 由它们的中心定义

浏览器支持

表中的数字指定了完全支持该属性的第一个浏览器版本。

后边跟 -webkit-、-moz- 或 -o- 的数字指定了需加上前缀才能支持属性的第一个版本。

属性					
linear-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.1 -o-
radial-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.6 -o-
repeating-linear-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.1 -o-
repeating-radial-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.6 -o-

CSS3 线性渐变

为了创建一个线性渐变，你必须至少定义两种颜色结点。颜色结点即你想要呈现平稳过渡的颜色。同时，你也可以设置一个起点和一个方向（或一个角度）。

线性渐变的实例：

线性渐变

语法

```
background: linear-gradient(direction, color-stop1, color-stop2, ...);
```

线性渐变 - 从上到下（默认情况下）

下面的实例演示了从顶部开始的线性渐变。起点是红色，慢慢过渡到蓝色：

实例

从上到下的线性渐变：

```
#grad {
background: -webkit-linear-gradient(red, blue); /* Safari 5.1 - 6.0 */
background: -o-linear-gradient(red, blue); /* Opera 11.1 - 12.0 */
background: -moz-linear-gradient(red, blue); /* Firefox 3.6 - 15 */
background: linear-gradient(red, blue); /* 标准的语法 */
}
```

尝试一下 »

线性渐变 - 从左到右

下面的实例演示了从左边开始的线性渐变。起点是红色，慢慢过渡到蓝色：

实例

从左到右的线性渐变：

```
#grad {
background: -webkit-linear-gradient(left, red , blue); /* Safari 5.1 - 6.0 */
background: -o-linear-gradient(right, red, blue); /* Opera 11.1 - 12.0 */
background: -moz-linear-gradient(right, red, blue); /* Firefox 3.6 - 15 */
background: linear-gradient(to right, red , blue); /* 标准的语法 */
}
```

尝试一下 »

线性渐变 - 对角

你可以通过指定水平和垂直的起始位置来制作一个对角渐变。

下面的实例演示了从左上角开始（到右下角）的线性渐变。起点是红色，慢慢过渡到蓝色：

实例

从左上角到右下角的线性渐变：

```
#grad {
background: -webkit-linear-gradient(left top, red , blue); /* Safari 5.1 - 6.0 */
background: -o-linear-gradient(bottom right, red, blue); /* Opera 11.1 - 12.0 */
background: -moz-linear-gradient(bottom right, red, blue); /* Firefox 3.6 - 15 */
background: linear-gradient(to bottom right, red , blue); /* 标准的语法 */
}
```

尝试一下 »

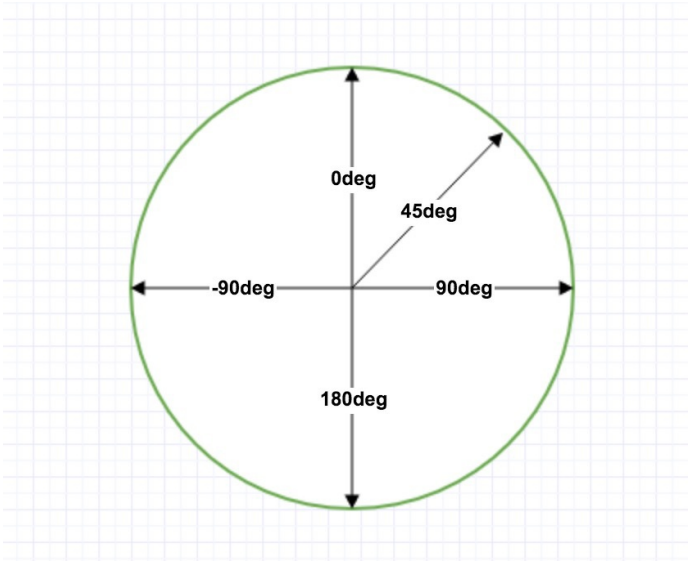
使用角度

如果你想要在渐变的方向上做更多的控制，你可以定义一个角度，而不用预定义方向（to bottom、to top、to right、to left、to bottom right，等等）。

语法

```
background: linear-gradient(angle, color-stop1, color-stop2);
```

角度是指水平线和渐变线之间的角度，逆时针方向计算。换句话说，0deg 将创建一个从下到上的渐变，90deg 将创建一个从左到右的渐变。



但是，请注意很多浏览器(Chrome,Safari,fiefox等)的使用了旧的标准，即 0deg 将创建一个从左到右的渐变，90deg 将创建一个从下到上的渐变。换算公式 **90 - x = y** 其中 x 为标准角度，y为非标准角度。

下面的实例演示了如何在线性渐变上使用角度：

实例

带有指定的角度的线性渐变：

```
#grad {
background: -webkit-linear-gradient(180deg, red, blue); /* Safari 5.1 - 6.0 */
background: -o-linear-gradient(180deg, red, blue); /* Opera 11.1 - 12.0 */
background: -moz-linear-gradient(180deg, red, blue); /* Firefox 3.6 - 15 */
background: linear-gradient(180deg, red, blue); /* 标准的语法 */
}
```

尝试一下 »

使用多个颜色结点

下面的实例演示了如何设置多个颜色结点：

实例

带有多个颜色结点的从上到下的线性渐变：

```
#grad {
background: -webkit-linear-gradient(red, green, blue); /* Safari 5.1 - 6.0 */
background: -o-linear-gradient(red, green, blue); /* Opera 11.1 - 12.0 */
background: -moz-linear-gradient(red, green, blue); /* Firefox 3.6 - 15 */
background: linear-gradient(red, green, blue); /* 标准的语法 */
}
```

尝试一下 »

下面的实例演示了如何创建一个带有彩虹颜色和文本的线性渐变：

实例

```
#grad {
/* Safari 5.1 - 6.0 */
background: -webkit-linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
/* Opera 11.1 - 12.0 */
background: -o-linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
/* Firefox 3.6 - 15 */
background: -moz-linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
/* 标准的语法 */
background: linear-gradient(to right, red,orange,yellow,green,blue,indigo,violet);
}
```

尝试一下 »

使用透明度（transparent）

CSS3 渐变也支持透明度（transparent），可用于创建减弱变淡的效果。

为了添加透明度，我们使用 rgba() 函数来定义颜色结点。rgba() 函数中的最后一个参数可以是从 0 到 1 的值，它定义了颜色的透明度：0 表示完全透明，1 表示完全不透明。

下面的实例演示了从左边开始的线性渐变。起点是完全透明，慢慢过渡到完全不透明的红色：

实例

从左到右的线性渐变，带有透明度：

```
#grad {
background: -webkit-linear-gradient(left,rgba(255,0,0,0),rgba(255,0,0,1)); /* Safari 5.1 - 6 */
background: -o-linear-gradient(right,rgba(255,0,0,0),rgba(255,0,0,1)); /* Opera 11.1 - 12*/
background: -moz-linear-gradient(right,rgba(255,0,0,0),rgba(255,0,0,1)); /* Firefox 3.6 - 15*/
background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1)); /* 标准的语法 */
}
```

尝试一下 »

重复的线性渐变

repeating-linear-gradient() 函数用于重复线性渐变：

实例

一个重复的线性渐变：

```
#grad {
/* Safari 5.1 - 6.0 */
background: -webkit-repeating-linear-gradient(red, yellow 10%, green 20%);
/* Opera 11.1 - 12.0 */
background: -o-repeating-linear-gradient(red, yellow 10%, green 20%);
/* Firefox 3.6 - 15 */
background: -moz-repeating-linear-gradient(red, yellow 10%, green 20%);
/* 标准的语法 */
background: repeating-linear-gradient(red, yellow 10%, green 20%);
}
```

尝试一下 »

CSS3 径向渐变

径向渐变由它的中心定义。

为了创建一个径向渐变，你也必须至少定义两种颜色结点。颜色结点即你想要呈现平稳过渡的颜色。同时，你也可以指定渐变的中心、形状（圆形或椭圆形）、大小。默认情况下，渐变的中心是 center（表示在中心点），渐变的形状是 ellipse（表示椭圆形），渐变的大小是 farthest-corner（表示到最远的角落）。

径向渐变的实例：

Radial gradient

语法

```
background: radial-gradient(center, shape size, start-color, ..., last-color);
```

径向渐变 - 颜色结点均匀分布（默认情况下）

实例

颜色结点均匀分布的径向渐变：

```
#grad {
background: -webkit-radial-gradient(red, green, blue); /* Safari 5.1 - 6.0 */
background: -o-radial-gradient(red, green, blue); /* Opera 11.6 - 12.0 */
background: -moz-radial-gradient(red, green, blue); /* Firefox 3.6 - 15 */
background: radial-gradient(red, green, blue); /* 标准的语法 */
}
```

尝试一下 »

径向渐变 - 颜色结点不均匀分布

实例

颜色结点不均匀分布的径向渐变：

```
#grad {
background: -webkit-radial-gradient(red 5%, green 15%, blue 60%); /* Safari 5.1 - 6.0 */
background: -o-radial-gradient(red 5%, green 15%, blue 60%); /* Opera 11.6 - 12.0 */
background: -moz-radial-gradient(red 5%, green 15%, blue 60%); /* Firefox 3.6 - 15 */
background: radial-gradient(red 5%, green 15%, blue 60%); /* 标准的语法 */
}
```

尝试一下 »

设置形状

shape 参数定义了形状。它可以是值 circle 或 ellipse。其中，circle 表示圆形，ellipse 表示椭圆形。默认值是 ellipse。

实例

形状为圆形的径向渐变：

```
#grad {
background: -webkit-radial-gradient(circle, red, yellow, green); /* Safari 5.1 - 6.0 */
background: -o-radial-gradient(circle, red, yellow, green); /* Opera 11.6 - 12.0 */
background: -moz-radial-gradient(circle, red, yellow, green); /* Firefox 3.6 - 15 */
background: radial-gradient(circle, red, yellow, green); /* 标准的语法 */
}
```

尝试一下 »

不同尺寸大小关键字的使用

size 参数定义了渐变的大小。它可以是以下四个值：

closest-side

farthest-side

closest-corner

farthest-corner

实例

带有不同尺寸大小关键字的径向渐变：

```
#grad1 {
/* Safari 5.1 - 6.0 */
background: -webkit-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
/* Opera 11.6 - 12.0 */
background: -o-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
/* Firefox 3.6 - 15 */
background: -moz-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
/* 标准的语法 */
```

```
background: radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
}
#grad2 {
/* Safari 5.1 - 6.0 */
background: -webkit-radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
/* Opera 11.6 - 12.0 */
background: -o-radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
/* Firefox 3.6 - 15 */
background: -moz-radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
/* 标准的语法 */
background: radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
}
```

尝试一下 »

重复的径向渐变

repeating-radial-gradient() 函数用于重复径向渐变：

实例

一个重复的径向渐变：

```
#grad {
/* Safari 5.1 - 6.0 */
background: -webkit-repeating-radial-gradient(red, yellow 10%, green 15%);
/* Opera 11.6 - 12.0 */
background: -o-repeating-radial-gradient(red, yellow 10%, green 15%);
/* Firefox 3.6 - 15 */
background: -moz-repeating-radial-gradient(red, yellow 10%, green 15%);
/* 标准的语法 */
background: repeating-radial-gradient(red, yellow 10%, green 15%);
}
```

尝试一下 »

□ CSS3 用户界面　　CSS3 圆角 □

□

2 篇笔记

#2

□写笔记

□

CSS 中 border: 1px solid rgba(0, 0, 0, 0.1); 是什么意思？

意思是: 设定元素的边框为 1 像素宽、实线、颜色使用 rgba 来表达。

其中，rgba 是 CSS3 中的属性。rgba 括号中前 3 个数字代表着 red green blue 三种颜色的 rgb 值（0-255），最后一个是设定这个颜色的透明度即 alpha 值。范围从 0 到 1，越接近 1，代表透明度越低。

薛定谔的bug5个月前 (04-30)

#1

□

径向渐变： radial-gradient（设置渐变的中心，渐变形状 渐变大小，起始颜色值，中间颜色值 中间颜色位置，终点颜色）

渐变中心，可选参数，如30px 20px指距离左侧30px距离上侧20px，可以是像素，可以是百分比，也可以是关键字，默认为中心位置。

渐变形状，可选参数，可以取值circle或ellipse【默认】

渐变大小，可循环参数，可以取值

closest-side：

指定径向渐变的半径长度为从圆心到离圆心最近的边

closest-corner：

指定径向渐变的半径长度为从圆心到离圆心最近的角

farthest-side：

指定径向渐变的半径长度为从圆心到离圆心最远的边

farthest-corner：

指定径向渐变的半径长度为从圆心到离圆心最远的角

contain：

包含，指定径向渐变的半径长度为从圆心到离圆心最近的点。类同于closest-side

cover：

覆盖，指定径向渐变的半径长度为从圆心到离圆心最远的点。类同于farthest-corner

CSS3 文本效果

CSS3 文本效果

CSS3中包含几个新的文本特征。

在本章中您将了解以下文本属性：

- text-shadow
- box-shadow
- text-overflow
- word-wrap
- word-break

浏览器支持

属性					
text-shadow	4.0	10.0	3.5	4.0	9.5
box-shadow	10.0 4.0 -webkit-	9.0	4.0 3.5 -moz-	5.1 3.1 -webkit-	10.5
text-overflow	4.0	6.0	7.0	3.1	11.0 9.0 -o-
word-wrap	23.0	5.5	3.5	6.1	12.1
word-break	4.0	5.5	15.0	3.1	15.0

CSS3 的文本阴影

CSS3 中，text-shadow属性适用于文本阴影。

您指定了水平阴影，垂直阴影，模糊的距离，以及阴影的颜色：

实例

给标题添加阴影:

```
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
```

尝试一下 »

CSS3 box-shadow属性

CSS3 中 CSS3 box-shadow 属性适用于盒子阴影

实例

```
div {
box-shadow: 10px 10px 5px #888888;
}
```

尝试一下 »

接下来给阴影添加颜色

实例

```
div {
box-shadow: 10px 10px grey;
}
```

尝试一下 »

接下来给阴影添加一个模糊效果

实例

```
div {
box-shadow: 10px 10px 5px grey;
}
```

尝试一下 »

你也可以在 **::before** 和 **::after** 两个伪元素中添加阴影效果

实例

```
#boxshadow {
position: relative;
b ox-shadow: 1px 2px 4px rgba(0, 0, 0, .5);
pa dding: 10px;
bac kground: white;
}
#boxshadow img {
width: 100%;
border: 1px solid #8a4419;
border-style: inset;
}
#boxshadow::after {
content: '';
position: absolute;
z-index: -1; /* hide shadow behind image */
box-shadow: 0 15px 20px rgba(0, 0, 0, 0.3);
width: 70%;
left: 15%; /* one half of the remaining 30% */
height: 100px;
bottom: 0;
}
```

尝试一下 »

阴影的一个使用特例是卡片效果

实例

```
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;
}
```

文字卡片 » 图片卡片 »

CSS3 Text Overflow属性

CSS3文本溢出属性指定应向用户如何显示溢出内容

实例

```
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;
}
```

尝试一下 »

CSS3的换行

如果某个单词太长，不适合在一个区域内，它扩展到外面：

CSS3中，自动换行属性允许您强制文本换行 - 即使这意味着分裂它中间的一个字：

CSS代码如下：

实例

允许长文本换行:

```
p {word-wrap:break-word;}
```

尝试一下 »

CSS3 单词拆分换行

CSS3 单词拆分换行属性指定换行规则：

CSS代码如下：

实例

```
p.test1 {
word-break: keep-all;
}
p.test2 {
word-break: break-all;
}
```

尝试一下 »

新文本属性

属性	描述	CSS
hanging-punctuation	规定标点字符是否位于线框之外。	3
punctuation-trim	规定是否对标点字符进行修剪。	3
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行。	3
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色。	3
text-justify	规定当 text-align 设置为 "justify" 时所使用的对齐方法。	3
text-outline	规定文本的轮廓。	3
text-overflow	规定当文本溢出包含元素时发生的事情。	3
text-shadow	向文本添加阴影。	3
text-wrap	规定文本的换行规则。	3
word-break	规定非中日韩文本的换行规则。	3
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行。	3

CSS3 背景 CSS3 字体

点我分享笔记

反馈/建议

CSS3 文本效果 CSS3 2D 转换

CSS3 字体

CSS3 @font-face 规则

以前 **CSS3** 的版本，网页设计师不得不使用用户计算机上已经安装的字体。

使用 **CSS3**，网页设计师可以使用他/她喜欢的任何字体。

当你发现您要使用的字体文件时，只需简单的将字体文件包含在网站中，它会自动下载给需要的用户。

您所选择的字体在新的 `CSS3` 版本有关于 `@font-face` 规则描述。

您"自己的"的字体是在 `CSS3 @font-face` 规则中定义的。

浏览器支持

表格中的数字表示支持该属性的第一个浏览器版本号。

属性					
@font-face	4.0	9.0	3.5	3.2	10.0

Internet Explorer 9+, Firefox, Chrome, Safari, 和 Opera 支持 WOFF (Web Open Font Format) 字体.

Firefox, Chrome, Safari, 和 Opera 支持 .ttf(True Type字体)和.otf(OpenType)字体字体类型）。

Chrome, Safari 和 Opera 也支持 SVG 字体/折叠.

Internet Explorer 同样支持 EOT (Embedded OpenType) 字体.

注意： Internet Explorer 8 以及更早的版本不支持新的 @font-face 规则。

使用您需要的字体

在新的 @font-face 规则中，您必须首先定义字体的名称（比如 myFirstFont），然后指向该字体文件。

提示：URL请使用小写字母的字体，大写字母在IE中会产生意外的结果

如需为 HTML 元素使用字体，请通过 font-family 属性来引用字体的名称 (myFirstFont)：

实例

```
<style>
@font-face
{
font-family: myFirstFont;
src: url(sansation_light.woff);
}
div
{
font-family:myFirstFont;
}
</style>
```

尝试一下 »

使用粗体文本

您必须添加另一个包含粗体文字的@font-face规则：

实例

```
@font-face
{
font-family: myFirstFont;
src: url(sansation_bold.woff);
font-weight:bold;
}
```

尝试一下 »

该文件"Sansation_Bold.ttf"是另一种字体文件，包含Sansation字体的粗体字。

浏览器使用这一文本的字体系列"myFirstFont"时应该呈现为粗体。

这样你就可以有许多相同的字体@font-face的规则。

CSS3 字体描述

下表列出了所有的字体描述和里面的@font-face规则定义：

描述符	值	描述

font-family	*name*	必需。规定字体的名称。
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。定义如何拉伸字体。默认是 "normal"。
font-style	normal italic oblique	可选。定义字体的样式。默认是 "normal"。
font-weight	normal bold 100 200 300 400 500 600 700 800 900	可选。定义字体的粗细。默认是 "normal"。
unicode-range	*unicode-range*	可选。定义字体支持的 UNICODE 字符范围。默认是 "U+0-10FFFF"。

□ CSS3 文本效果 CSS3 2D 转换 □

CSS3 2D 转换

CSS3 转换

CSS3 转换可以可以对元素进行移动、缩放、转动、拉长或拉伸。

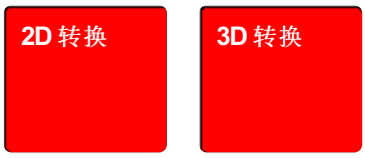
CSS3 Transforms

它是如何工作？

转换的效果是让某个元素改变形状，大小和位置。

您可以使用 2D 或 3D 转换来转换您的元素。

鼠标移动到以下元素上，查看 2D 和 3D 的转换效果：



浏览器支持

表格中的数字表示支持该属性的第一个浏览器版本号。

紧跟在 -webkit-, -ms- 或 -moz- 前的数字为支持该前缀属性的第一个浏览器版本号。

属性					
transform	36.0 4.0 -webkit-	10.0 9.0 -ms-	16.0 3.5 -moz-	3.2 -webkit-	23.0 15.0 -webkit- 12.1 10.5 -o-
transform-origin (two-value syntax)	36.0 4.0 -webkit-	10.0 9.0 -ms-	16.0 3.5 -moz-	3.2 -webkit-	23.0 15.0 -webkit- 12.1 10.5 -o-

Internet Explorer 10, Firefox, 和 Opera支持transform 属性.

Chrome 和 Safari 要求前缀 -webkit- 版本.

注意： Internet Explorer 9 要求前缀 -ms- 版本.

2D 转换

在本章您将了解2D变换方法：

- translate()
- rotate()
- scale()
- skew()
- matrix()

在下一章中您将了解3D转换。

实例

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg); /* IE 9 */
-webkit-transform: rotate(30deg); /* Safari and Chrome */
```

```
}
```

尝试一下 »

translate() 方法

translate()方法，根据左(X轴)和顶部(Y轴)位置给定的参数，从当前元素位置移动。

实例

```
div
{
transform: translate(50px,100px);
-ms-transform: translate(50px,100px); /* IE 9 */
-webkit-transform: translate(50px,100px); /* Safari and Chrome */
}
```

尝试一下 »

translate值（50px，100px）是从左边元素移动50个像素，并从顶部移动100像素。

rotate() 方法

rotate()方法，在一个给定度数顺时针旋转的元素。负值是允许的，这样是元素逆时针旋转。

实例

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg); /* IE 9 */
-webkit-transform: rotate(30deg); /* Safari and Chrome */
}
```

尝试一下 »

rotate值（30deg）元素顺时针旋转30度。

scale() 方法

scale()方法，该元素增加或减少的大小，取决于宽度（X轴）和高度（Y轴）的参数：

实例

```
-ms-transform:scale(2,3); /* IE 9 */
-webkit-transform: scale(2,3); /* Safari */
transform: scale(2,3); /* 标准语法 */
```

尝试一下 »

scale（2,3）转变宽度为原来的大小的2倍，和其原始大小3倍的高度。

skew() 方法

语法

```
transform:skew(<angle> [,<angle>]);
```

包含两个参数值，分别表示X轴和Y轴倾斜的角度，如果第二个参数为空，则默认为0，参数为负表示向相反方向倾斜。

skewX(<angle>);表示只在X轴(水平方向)倾斜。

skewY(<angle>);表示只在Y轴(垂直方向)倾斜。

实例

```
div
{
transform: skew(30deg,20deg);
-ms-transform: skew(30deg,20deg); /* IE 9 */
-webkit-transform: skew(30deg,20deg); /* Safari and Chrome */
}
```

尝试一下 »

skew(30deg,20deg) 元素在X轴和Y轴上倾斜20度30度。

matrix() 方法

matrix()方法和2D变换方法合并成一个。

matrix 方法有六个参数，包含旋转，缩放，移动（平移）和倾斜功能。

实例

利用matrix()方法旋转div元素30°

```
div
{
transform:matrix(0.866,0.5,-0.5,0.866,0,0);
-ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* IE 9 */
-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Safari and Chrome */
}
```

尝试一下 »

新转换属性

以下列出了所有的转换属性:

Property	描述	CSS
transform	适用于2D或3D转换的元素	3
transform-origin	允许您更改转化元素位置	3

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。

□ CSS3 字体 CSS3 3D 转换 □

CSS3 3D 转换

3D 转换

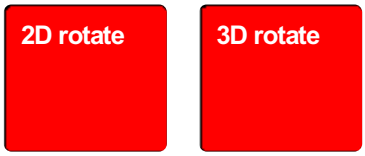
CSS3 允许您使用 3D 转换来对元素进行格式化。

在本章中，您将学到其中的一些 3D 转换方法：

rotateX()

rotateY()

点击下面的元素，来查看 2D 转换与 3D 转换之间的不同之处：



浏览器支持

表格中的数字表示支持该属性的第一个浏览器版本号。

紧跟在 -webkit-, -ms- 或 -moz- 前的数字为支持该前缀属性的第一个浏览器版本号。

属性					
transform	36.0 12.0 -webkit-	10.0	16.0 10.0 -moz-	4.0 -webkit-	23.0 15.0 -webkit-
transform-origin (three-value syntax)	36.0 12.0 -webkit-	10.0	16.0 10.0 -moz-	4.0 -webkit-	23.0 15.0 -webkit-
transform-style	36.0 12.0 -webkit-	11.0	16.0 10.0 -moz-	4.0 -webkit-	23.0 15.0 -webkit-
perspective	36.0 12.0 -webkit-	10.0	16.0 10.0 -moz-	4.0 -webkit-	23.0 15.0 -webkit-
perspective-origin	36.0 12.0 -webkit-	10.0	16.0 10.0 -moz-	4.0 -webkit-	23.0 15.0 -webkit-

backface-visibility	36.0 12.0 -webkit-	10.0	16.0 10.0 -moz-	4.0 -webkit-	23.0 15.0 -webkit-

rotateX() 方法

rotateX()方法，围绕其在一个给定度数X轴旋转的元素。

实例

```
div
{
transform: rotateX(120deg);
-webkit-transform: rotateX(120deg); /* Safari 与 Chrome */
}
```

尝试一下 »

rotateY() 方法

rotateY()方法，围绕其在一个给定度数Y轴旋转的元素。

实例

```
div
{
transform: rotateY(130deg);
-webkit-transform: rotateY(130deg); /* Safari 与 Chrome */
}
```

尝试一下 »

转换属性

下表列出了所有的转换属性：

属性	描述	CSS
transform	向元素应用 2D 或 3D 转换。	3
transform-origin	允许你改变被转换元素的位置。	3
transform-style	规定被嵌套元素如何在 3D 空间中显示。	3
perspective	规定 3D 元素的透视效果。	3
perspective-origin	规定 3D 元素的底部位置。	3
backface-visibility	定义元素在不面对屏幕时是否可见。	3

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。

CSS3 2D 转换 CSS3 过渡

点我分享笔记

反馈/建议

CSS3 3D 转换 CSS3 动画

CSS3 过渡

CSS3 过渡

CSS3中，我们为了添加某种效果可以从一种样式转变到另一个的时候，无需使用Flash动画或JavaScript。用鼠标移过下面的元素：

用鼠标移过下面的元素：

浏览器支持

表格中的数字表示支持该属性的第一个浏览器版本号。

紧跟在 -webkit-, -ms- 或 -moz- 前的数字为支持该前缀属性的第一个浏览器版本号。

属性					
transition	26.0 4.0 -webkit-	10.0	16.0 4.0 -moz-	6.1 3.1 -webkit-	12.1 10.5 -o-

transition-delay	26.0 4.0 -webkit-	10.0	16.0 4.0 -moz-	6.1 3.1 -webkit-	12.1 10.5 -o-
transition-duration	26.0 4.0 -webkit-	10.0	16.0 4.0 -moz-	6.1 3.1 -webkit-	12.1 10.5 -o-
transition-property	26.0 4.0 -webkit-	10.0	16.0 4.0 -moz-	6.1 3.1 -webkit-	12.1 10.5 -o-
transition-timing-function	26.0 4.0 -webkit-	10.0	16.0 4.0 -moz-	6.1 3.1 -webkit-	12.1 10.5 -o-

它是如何工作？

CSS3 过渡是元素从一种样式逐渐改变为另一种的效果。

要实现这一点，必须规定两项内容：

指定要添加效果的CSS属性

指定效果的持续时间。

实例

应用于宽度属性的过渡效果，时长为 2 秒：

```
div
{
transition: width 2s;
-webkit-transition: width 2s; /* Safari */
}
```

注意： 如果未指定的期限，transition将没有任何效果，因为默认值是0。

指定的CSS属性的值更改时效果会发生变化。一个典型CSS属性的变化是用户鼠标放在一个元素上时：

实例

规定当鼠标指针悬浮(:hover)于 元素上时：

```
div:hover
{
width:300px;
}
```

尝试一下 »

注意： 当鼠标光标移动到该元素时，它逐渐改变它原有样式

多项改变

要添加多个样式的变换效果，添加的属性由逗号分隔：

实例

添加了宽度，高度和转换效果：

```
div
{
transition: width 2s, height 2s, transform 2s;
-webkit-transition: width 2s, height 2s, -webkit-transform 2s;
}
```

尝试一下 »

过渡属性

下表列出了所有的过渡属性:

属性	描述	CSS

transition	简写属性，用于在一个属性中设置四个过渡属性。	3
transition-property	规定应用过渡的 CSS 属性的名称。	3
transition-duration	定义过渡效果花费的时间。默认是 0。	3
transition-timing-function	规定过渡效果的时间曲线。默认是 "ease"。	3
transition-delay	规定过渡效果何时开始。默认是 0。	3

下面的两个例子设置所有过渡属性：

实例

在一个例子中使用所有过渡属性：

```
div
{
transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
/* Safari */
-webkit-transition-property:width;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:linear;
-webkit-transition-delay:2s;
}
```

尝试一下 »

实例

与上面的例子相同的过渡效果，但是使用了简写的 transition 属性：

```
div
{
transition: width 1s linear 2s;
/* Safari */
-webkit-transition:width 1s linear 2s;
}
```

尝试一下 »

CSS3 3D 转换　　CSS3 动画

CSS3 动画

CSS3 动画

CSS3，我们可以创建动画，它可以取代许多网页动画图像，Flash动画，和JAVAScripts。

CSS3
动画

CSS3 @keyframes 规则

要创建CSS3动画，你将不得不了解@keyframes规则。

@keyframes规则是创建动画。 @keyframes规则内指定一个CSS样式和动画将逐步从目前的样式更改为新的样式。

浏览器支持

表格中的数字表示支持该属性的第一个浏览器版本号。

紧跟在 -webkit-, -ms- 或 -moz- 前的数字为支持该前缀属性的第一个浏览器版本号。

属性					
@keyframes	43.0 4.0 -webkit-	10.0	16.0 5.0 -moz-	9.0 4.0 -webkit-	30.0 15.0 -webkit- 12.0 -o-
animation	43.0 4.0 -webkit-	10.0	16.0 5.0 -moz-	9.0 4.0 -webkit-	30.0 15.0 -webkit- 12.0 -o-

实例

```
@keyframes myfirst
{
from {background: red;}
to {background: yellow;}
}
@-webkit-keyframes myfirst /* Safari 与 Chrome */
{
from {background: red;}
to {background: yellow;}
}
```

CSS3 动画

当在 **@keyframes** 创建动画，把它绑定到一个选择器，否则动画不会有任何效果。

指定至少这两个CSS3的动画属性绑定向一个选择器：

规定动画的名称

规定动画的时长

实例

把 "myfirst" 动画捆绑到 div 元素，时长：5 秒：

```
div
{
animation: myfirst 5s;
-webkit-animation: myfirst 5s; /* Safari 与 Chrome */
}
```

尝试一下 »

注意：您必须定义动画的名称和动画的持续时间。如果省略的持续时间，动画将无法运行，因为默认值是0。

CSS3动画是什么？

动画是使元素从一种样式逐渐变化为另一种样式的效果。

您可以改变任意多的样式任意多的次数。

请用百分比来规定变化发生的时间，或用关键词 "from" 和 "to"，等同于 0% 和 100%。

0% 是动画的开始，100% 是动画的完成。

为了得到最佳的浏览器支持，您应该始终定义 0% 和 100% 选择器。

实例

当动画为 25% 及 50% 时改变背景色，然后当动画 100% 完成时再次改变：

```
@keyframes myfirst
{
0% {background: red;}
25% {background: yellow;}
50% {background: blue;}
100% {background: green;}
}
@-webkit-keyframes myfirst /* Safari 与 Chrome */
{
0% {background: red;}
25% {background: yellow;}
50% {background: blue;}
100% {background: green;}
}
```

尝试一下 »

实例

改变背景色和位置：

```
@keyframes myfirst
{
0% {background: red; left:0px; top:0px;}
25% {background: yellow; left:200px; top:0px;}
50% {background: blue; left:200px; top:200px;}
75% {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
@-webkit-keyframes myfirst /* Safari 与 Chrome */
{
0% {background: red; left:0px; top:0px;}
25% {background: yellow; left:200px; top:0px;}
50% {background: blue; left:200px; top:200px;}
75% {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
```

尝试一下 »

CSS3的动画属性

下面的表格列出了 @keyframes 规则和所有动画属性：

属性	描述	CSS
@keyframes	规定动画。	3
animation	所有动画属性的简写属性，除了 animation-play-state 属性。	3
animation-name	规定 @keyframes 动画的名称。	3

animation-duration	规定动画完成一个周期所花费的秒或毫秒。默认是 0。	3
animation-timing-function	规定动画的速度曲线。默认是 "ease"。	3
animation-fill-mode	规定当动画不播放时（当动画完成时，或当动画有一个延迟未开始播放时），要应用到元素的样式。	3
animation-delay	规定动画何时开始。默认是 0。	3
animation-iteration-count	规定动画被播放的次数。默认是 1。	3
animation-direction	规定动画是否在下一周期逆向地播放。默认是 "normal"。	3
animation-play-state	规定动画是否正在运行或暂停。默认是 "running"。	3

下面两个例子设置所有动画属性：

实例

运行myfirst动画，设置所有的属性：

```
div
{
animation-name: myfirst;
animation-duration: 5s;
animation-timing-function: linear;
animation-delay: 2s;
animation-iteration-count: infinite;
animation-direction: alternate;
animation-play-state: running;
/* Safari 与 Chrome: */
-webkit-animation-name: myfirst;
-webkit-animation-duration: 5s;
-webkit-animation-timing-function: linear;
-webkit-animation-delay: 2s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-webkit-animation-play-state: running;
}
```

尝试一下 »

实例

与上面的动画相同，但是使用了简写的动画 animation 属性：

```
div
{
animation: myfirst 5s linear 2s infinite alternate;
/* Safari 与 Chrome: */
-webkit-animation: myfirst 5s linear 2s infinite alternate;
}
```

尝试一下 »

CSS3 过渡 CSS3 多列

CSS3 多列

CSS3 可以将文本内容设计成像报纸一样的多列布局，如下实例:

菜鸟教程 - 学的不仅是技术，更是梦想！菜鸟教程(www.runoob.com)提供了最全的编程技术基础教程, 介绍了HTML、CSS、Javascript、Python，Java，Ruby，C，PHP , MySQL等各种编程语言的基础知识。 同时本站中也提供了大量的在线实例，通过实例，您可以更好的学习编程。

浏览器支持

表格中的数字表示支持该方法的第一个浏览器的版本号。

紧跟在数字后面的 -webkit- 或 -moz- 为指定浏览器的前缀。

属性					
column-count	4.0 -webkit-	10.0	2.0 -moz-	3.1 -webkit-	15.0 -webkit- 11.1
column-gap	4.0 -webkit-	10.0	2.0 -moz-	3.1 -webkit-	15.0 -webkit- 11.1
column-rule	4.0 -webkit-	10.0	2.0 -moz-	3.1 -webkit-	15.0 -webkit- 11.1
column-rule-color	4.0 -webkit-	10.0	2.0 -moz-	3.1 -webkit-	15.0 -webkit 11.1
column-rule-style	4.0 -webkit-	10.0	2.0 -moz-	3.1 -webkit-	15.0 -webkit 11.1
column-rule-width	4.0 -webkit-	10.0	2.0 -moz-	3.1 -webkit-	15.0 -webkit 11.1
column-width	4.0 -webkit-	10.0	2.0 -moz-	3.1 -webkit-	15.0 -webkit 11.1

CSS3 多列属性

本章节我们将学习以下几个 CSS3 的多列属性:

```
column-count
column-gap
column-rule-style
column-rule-width
column-rule-color
column-rule
column-span
column-width
```

CSS3 创建多列

column-count 属性指定了需要分割的列数。

以下实例将 元素中的文本分为 3 列：

实例

```
div {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;
}
```

尝试一下 »

CSS3 多列中列与列间的间隙

column-gap 属性指定了列与列间的间隙。

以下实例指定了列与列间的间隙为 40 像素：

实例

```
div {
    -webkit-column-gap: 40px; /* Chrome, Safari, Opera */
    -moz-column-gap: 40px; /* Firefox */
    column-gap: 40px;
}
```

尝试一下 »

CSS3 列边框

column-rule-style 属性指定了列与列间的边框样式：

实例

```
div {
    -webkit-column-rule-style: solid; /* Chrome, Safari, Opera */
    -moz-column-rule-style: solid; /* Firefox */
    column-rule-style: solid;
}
```

尝试一下 »

column-rule-width 属性指定了两列的边框厚度:

实例

```
div {
    -webkit-column-rule-width: 1px; /* Chrome, Safari, Opera */
    -moz-column-rule-width: 1px; /* Firefox */
    column-rule-width: 1px;
}
```

尝试一下 »

column-rule-color 属性指定了两列的边框颜色：

实例

```
div {
    -webkit-column-rule-color: lightblue; /* Chrome, Safari, Opera */
    -moz-column-rule-color: lightblue; /* Firefox */
    column-rule-color: lightblue;
}
```

尝试一下 »

column-rule 属性是 column-rule-* 所有属性的简写。

以下实例设置了列直接的边框的厚度，样式及颜色：

实例

```
div {
    -webkit-column-rule: 1px solid lightblue; /* Chrome, Safari, Opera */
    -moz-column-rule: 1px solid lightblue; /* Firefox */
    column-rule: 1px solid lightblue;
}
```

尝试一下 »

指定元素跨越多少列

以下实例指定 <h2> 元素跨越所有列：

实例

```
h2 {
    -webkit-column-span: all; /* Chrome, Safari, Opera */
    column-span: all;
}
```

尝试一下 »

指定列的宽度

column-width 属性指定了列的宽度。

实例

```
div {
    -webkit-column-width: 100px; /* Chrome, Safari, Opera */
    column-width: 100px;
}
```

尝试一下 »

CSS3 多列属性

下表列出了所有 CSS3 的多列属性：

属性	描述
column-count	指定元素应该被分割的列数。
column-fill	指定如何填充列
column-gap	指定列与列之间的间隙
column-rule	所有 column-rule-* 属性的简写
column-rule-color	指定两列间边框的颜色
column-rule-style	指定两列间边框的样式
column-rule-width	指定两列间边框的厚度
column-span	指定元素要跨越多少列
column-width	指定列的宽度
columns	设置 column-width 和 column-count 的简写

CSS3 用户界面

CSS3 用户界面

在 CSS3 中, 增加了一些新的用户界面特性来调整元素尺寸，框尺寸和外边框。

在本章中，您将了解以下的用户界面属性：

- resize
- box-sizing
- outline-offset

浏览器支持

表格中的数字表示支持该属性的第一个浏览器版本号。

紧跟在 -webkit-, -ms- 或 -moz- 前的数字为支持该前缀属性的第一个浏览器版本号。

属性					
resize	4.0	不兼容	5.0 4.0 -moz-	4.0	15.0
box-sizing	10.0 4.0 -webkit-	8.0	29.0 2.0 -moz-	5.1 3.1 -webkit-	9.5
outline-offset	4.0	不兼容	5.0 4.0 -moz-	4.0	9.5

CSS3 调整尺寸(Resizing)

CSS3中，resize属性指定一个元素是否应该由用户去调整大小。
这个 div 元素由用户调整大小。 (在 Firefox 4+, Chrome, 和 Safari中)
CSS代码如下：

实例

由用户指定一个div元素尺寸大小：

```
div
{
resize:both;
overflow:auto;
}
```

尝试一下 »

CSS3 方框大小调整(Box Sizing)

box-sizing 属性允许您以确切的方式定义适应某个区域的具体内容。

实例

规定两个并排的带边框方框：

```
div
{
box-sizing:border-box;
-moz-box-sizing:border-box; /* Firefox */
width:50%;
float:left;
}
```

尝试一下 »

CSS3 外形修饰（outline-offset ）

outline-offset 属性对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。

轮廓与边框有两点不同：

轮廓不占用空间

轮廓可能是非矩形

这个 div 在边框之外 15 像素处有一个轮廓。
CSS 代码如下:

实例

规定边框边缘之外 15 像素处的轮廓：

```
div
{
border:2px solid black;
outline:2px solid red;
outline-offset:15px;
}
```

尝试一下 »

新的用户界面特性

属性	说明	CSS
appearance	允许您使一个元素的外观像一个标准的用户界面元素	3
box-sizing	允许你以适应区域而用某种方式定义某些元素	3
icon	为创作者提供了将元素设置为图标等价物的能力。	3
nav-down	指定在何处使用箭头向下导航键时进行导航	3
nav-index	指定一个元素的Tab的顺序	3
nav-left	指定在何处使用左侧的箭头导航键进行导航	3
nav-right	指定在何处使用右侧的箭头导航键进行导航	3
nav-up	指定在何处使用箭头向上导航键时进行导航	3
outline-offset	外轮廓修饰并绘制超出边框的边缘	3

resize	指定一个元素是否是由用户调整大小	3

CSS3 多列 CSS3 渐变

CSS3 圆角 CSS 按钮

CSS 图片

本章节将为大家介绍如何使用 CSS 来布局图片。

圆角图片

实例

圆角图片:

```
img {
    border-radius: 8px;
}
```

尝试一下 »

实例

椭圆形图片:

```
img {
    border-radius: 50%;
}
```

尝试一下 »

缩略图

我们使用 border 属性来创建缩略图。

实例

```
img {
    border: 1px solid #ddd;
    border-radius: 4px;
    padding: 5px;
}

<img src="paris.jpg" alt="Paris">
```

尝试一下 »

实例

```
a {
    display: inline-block;
    border: 1px solid #ddd;
    border-radius: 4px;
    padding: 5px;
    transition: 0.3s;
}

a:hover {
    box-shadow: 0 0 2px 1px rgba
    (0, 140, 186, 0.5);
}

<a href="paris.jpg">
  <img src="paris.jpg" alt="Paris">
</a>
```

尝试一下 »

响应式图片

响应式图片会自动适配各种尺寸的屏幕。

实例中，你可以通过重置浏览器大小查看效果:

如果你需要自由缩放图片，且图片放大的尺寸不大于其原始的最大值，则可使用以下代码：

实例

```
img {
    max-width: 100%;
    height: auto;
}
```

尝试一下 »

提示: Web 响应式设计更多内容可以参考 CSS 响应式设计教程。

图片文本

如何定位图片文本:

实例

尝试一下:

左上角 » 右上角 » 左下角 » 右下角 » 居中 »

卡片式图片

实例

```
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);
}

img {width: 100%}

div.container {
    text-align: center;
    padding: 10px 20px;
}
```

尝试一下 »

图片滤镜

CSS `filter` 属性用为元素添加可视效果 (例如：模糊与饱和度) 。

注意: Internet Explorer 或 Safari 5.1 (及更早版本) 不支持该属性。

实例

修改所有图片的颜色为黑白 (100% 灰度):

```
img {
    -webkit-filter: grayscale(100%); /* Chrome, Safari, Opera */
    filter: grayscale(100%);
}
```

Error response

Error code 404.

Message: File not found.

Error code explanation: 404 = Nothing matches the given URI.

尝试一下 »

提示: 访问 CSS 滤镜参考手册 查看更多内容。

响应式图片相册

实例

```
.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%;
    }
}
```

尝试一下 »

图片 Modal(模态)

本实例演示了如何结合 CSS 和 JavaScript 来一起渲染图片。

首先，我们使用 CSS 来创建 modal 窗口 (对话框), 默认是隐藏的。

然后，我们使用 JavaScript 来显示模态窗口，当我们点击图片时，图片会在弹出的窗口中显示：

实例

```
// 获取模态窗口
var modal = document.getElementById('myModal');

// 获取图片模态框，alt 属性作为图片弹出中文本描述
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;
    modalImg.alt = this.alt;
    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";
}
```

尝试一下 »

□ CSS3 圆角 CSS 按钮 □

□ CSS 图片 CSS 分页实例 □

CSS 按钮

本章节我们为大家介绍使用 CSS 来制作按钮。

基本按钮样式

默认按钮

CSS 按钮

CSS 实例

```
.button {
    background-color: #4CAF50; /* Green */
    border: none;
    color: white;
```

```
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
}
```

尝试一下 »

按钮颜色

我们可以使用 background-color 属性来设置按钮颜色:

CSS 实例

```
.button1 {background-color: #4CAF50;} /* Green */
.button2 {background-color: #008CBA;} /* Blue */
.button3 {background-color: #f44336;} /* Red */
.button4 {background-color: #e7e7e7; color: black;} /* Gray */
.button5 {background-color: #555555;} /* Black */
```

尝试一下 »

按钮大小

我们可以使用 font-size 属性来设置按钮大小:

CSS 实例

```
.button1 {font-size: 10px;}
.button2 {font-size: 12px;}
.button3 {font-size: 16px;}
.button4 {font-size: 20px;}
.button5 {font-size: 24px;}
```

尝试一下 »

圆角按钮

我们可以使用 border-radius 属性来设置圆角按钮:

CSS 实例

```
.button1 {border-radius: 2px;}
.button2 {border-radius: 4px;}
.button3 {border-radius: 8px;}
.button4 {border-radius: 12px;}
.button5 {border-radius: 50%;}
```

尝试一下 »

按钮边框颜色

绿 蓝 红 灰 黑

我们可以使用 border 属性设置按钮边框颜色:

CSS 实例

```
.button1 {
    background-color: white;
    color: black;
    border: 2px solid #4CAF50; /* Green */
}
...
```

尝试一下 »

鼠标悬停按钮

绿 蓝 红 灰 黑

绿 蓝 红 灰 黑

我们可以使用 :hover 选择器来修改鼠标悬停在按钮上的样式。

提示: 我们可以使用 transition-duration 属性来设置 "hover" 效果的速度:

CSS 实例

```
.button {
    -webkit-transition-duration: 0.4s; /* Safari */
    transition-duration: 0.4s;
}

.button:hover {
    background-color: #4CAF50; /* Green */
    color: white;
}
...
```

尝试一下 »

按钮阴影

阴影按钮 鼠标悬停后显示阴影

我们可以使用 box-shadow 属性来为按钮添加阴影:

CSS 实例

```
.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);
}
```

尝试一下 »

禁用按钮

正常按钮 禁用按钮

我们可以使用 opacity 属性为按钮添加透明度 (看起来类似 "disabled" 属性效果)。

提示: 我们可以添加 cursor 属性并设置为 "not-allowed" 来设置一个禁用的图片:

CSS 实例

```
.disabled {
    opacity: 0.6;
    cursor: not-allowed;
}
```

尝试一下 »

按钮宽度

250px

50%

100%

默认情况下，按钮的大小有按钮上的文本内容决定(根据文本内容匹配长度)。 我们可以使用 width 属性来设置按钮的宽度:

提示: 如果要设置固定宽度可以使用像素 (px) 为单位，如果要设置响应式的按钮可以设置为百分比。

CSS 实例

```
.button1 {width: 250px;}
.button2 {width: 50%;}
.button3 {width: 100%;}
```

尝试一下 »

按钮组

Button Button Button Button

移除外边距并添加 float:left 来设置按钮组:

CSS 实例

```
.button {
    float: left;
}
```

尝试一下 »

带边框按钮组

Button Button Button Button

我们可以使用 border 属性来设置带边框的按钮组:

CSS 实例

```
.button {
    float: left;
    border: 1px solid green
}
```

尝试一下 »

按钮动画

CSS 实例

鼠标移动到按钮上后添加箭头标记:

Hover

尝试一下 »

CSS 实例

点击时添加 "波纹" 效果:

Click

尝试一下 »

CSS 实例

点击时添加 "压下" 效果:

Click

尝试一下 »

□ CSS 图片

CSS 分页实例 □

□点我分享笔记

反馈/建议

□

CSS 分页实例

本章节我们将为大家介绍如何通过使用 CSS 来创建分页的实例。

简单分页

如果你的网站有很多个页面，你就需要使用分页来为每个页面做导航。

以下实例演示了如何使用 HTML 和 CSS 来创建分页：

CSS 实例

```
ul.pagination {
    display: inline-block;
    padding: 0;
    margin: 0;
}

ul.pagination li {display: inline;}

ul.pagination li a {
    color: black;
    float: left;
    padding: 8px 16px;
    text-decoration: none;
}
```

尝试一下 »

点击及鼠标悬停分页样式

« 1 2 3 4 5 6 7 »

如果点击当前页，可以使用 .active 来设置当期页样式，鼠标悬停可以使用 :hover 选择器来修改样式：

CSS 实例

```
ul.pagination li a.active {
    background-color: #4CAF50;
    color: white;
}

ul.pagination li a:hover:not(.active) {background-color: #ddd;}
```

尝试一下 »

CSS 实例

```
ul.pagination li a.active {
    background-color: #4CAF50;
    color: white;
}

ul.pagination li a:hover:not(.active) {background-color: #ddd;}
```

尝试一下 »

圆角样式

« 1 2 3 4 5 6 7 »

可以使用 border-radius 属性为选中的页码来添加圆角样式:

CSS 实例

```
ul.pagination li a {
    border-radius: 5px;
}

ul.pagination li a.active {
    border-radius: 5px;
}
```

尝试一下 »

鼠标悬停过渡效果

« 1 2 3 4 5 6 7 »

我们可以通过添加 transition 属性来为鼠标移动到页码上时添加过渡效果:

CSS 实例

```
ul.pagination li a {
    transition: background-color .3s;
}
```

尝试一下 »

带边框分页

« 1 2 3 4 5 6 7 »

我们可以使用 border 属性来添加带边框分页:

CSS 实例

```
ul.pagination li a {
    border: 1px solid #ddd; /* Gray */
}
```

尝试一下 »

圆角边框

提示: 在第一个分页链接和最后一个分页链接添加圆角:

« 1 2 3 4 5 6 7 »

CSS 实例

```
.pagination li:first-child a {
    border-top-left-radius: 5px;
    border-bottom-left-radius: 5px;
}

.pagination li:last-child a {
    border-top-right-radius: 5px;
    border-bottom-right-radius: 5px;
}
```

尝试一下 »

分页间隔

提示: 你可以使用 `margin` 属性来为每个页码直接添加空格:

« 1 2 3 4 5 6 7 »

CSS 实例

```
ul.pagination li a {
    margin: 0 4px; /* 0 is for top and bottom. Feel free to change it */
}
```

尝试一下 »

分页字体大小

« 1 2 3 4 5 6 7 »

我们可以使用 `font-size` 属性来设置分页的字体大小:

CSS 实例

```
ul.pagination li a {
    font-size: 22px;
}
```

尝试一下 »

居中分页

« 1 2 3 4 5 6 7 »

如果要让分页居中，可以在容器元素上 (如) 添加 **text-align:center** 样式:

CSS 实例

```
div.center {
    text-align: center;
}
```

尝试一下 »

更多实例

CSS 实例

Error response

Error code 404.

Message: File not found.

Error code explanation: 404 = Nothing matches the given URI.

尝试一下 »

面包屑导航

首页 / 前端 / HTML 教程 / HTML 段落

另外一种导航为面包屑导航，实例如下：

CSS 实例

```
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";
}
```

尝试一下 »

CSS 按钮 CSS3 框大小

点我分享笔记

反馈/建议

CSS 分页实例 CSS3 弹性盒子

CSS3 框大小

CSS3 box-sizing 属性可以设置 width 和 height 属性中包含了 padding(内边距) 和 border(边框)。

浏览器支持

表格中的数字表示支持该属性的第一个浏览器的版本号。

紧跟在数字后面的 -webkit- 或 -moz- 为指定浏览器的前缀。

属性					
box-sizing	10.0 4.0 -webkit-	8.0	29.0 2.0 -moz-	5.1 3.1 -webkit-	9.5

不使用 CSS3 box-sizing 属性

默认情况下，元素的宽度与高度计算方式如下：

width(宽) + padding(内边距) + border(边框) = 元素实际宽度

height(高) + padding(内边距) + border(边框) = 元素实际高度

这就意味着我们在设置元素的 width/height 时，元素真实展示的高度与宽度会更大(因为元素的边框与内边距也会计算在 width/height 中)。

这个是个较小的框 (width 为 300px，height 为 100px)。

这个是个较大的框 (width 为 300px，height 为 100px)。

以上两个 元素虽然宽度与高度设置一样，但真实展示的大小不一致，因为 div2 指定了内边距:

实例

```
.div1 {
    width: 300px;
    height: 100px;
    border: 1px solid blue;
}

.div2 {
    width: 300px;
    height: 100px;
    padding: 50px;
    border: 1px solid red;
}
```

尝试一下 »

使用这种方式如果想要获得较小的那个框且包含内边距，就不得不考虑到边框和内边距的宽度。

CSS3 的 box-sizing 属性很好的解决了这个问题。

使用 CSS3 box-sizing 属性

CSS3 box-sizing 属性在一个元素的 width 和 height 中包含 padding(内边距) 和 border(边框)。

如果在元素上设置了 box-sizing: border-box; 则 padding(内边距) 和 border(边框) 也包含在 width 和 height 中:

两个 div 现在是一样大小的!

菜鸟教程!

以下是两个 元素添加 box-sizing: border-box; 属性的简单实例。

实例

```
.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;
}
```

尝试一下 »

从结果上看 box-sizing: border-box; 效果更好，也正是很多开发人员需要的效果。

以下代码可以让所有元素以更直观的方式展示大小。很多浏览器已经支持 box-sizing: border-box; (但是并非所有 - 这就是为什么 input 和 text 元素设置了 width: 100%; 后的宽度却不一样)。

所有元素使用 box-sizing 是比较推荐的：

实例

```
* {
    box-sizing: border-box;
}
```

尝试一下 »

CSS 分页实例 CSS3 弹性盒子

点我分享笔记

反馈/建议

CSS3 框大小 CSS3 多媒体查询

CSS3 弹性盒子(Flex Box)

弹性盒子是 CSS3 的一种新的布局模式。

CSS3 弹性盒（ Flexible Box 或 flexbox），是一种当页面需要适应不同的屏幕大小以及设备类型时确保元素拥有恰当的行为的布局方式。

引入弹性盒布局模型的目的是提供一种更加有效的方式来对一个容器中的子元素进行排列、对齐和分配空白空间。

浏览器支持

表格中的数字表示支持该属性的第一个浏览器的版本号。

紧跟在数字后面的 -webkit- 或 -moz- 为指定浏览器的前缀。

属性					
Basic support (single-line flexbox)	29.0 21.0 -webkit-	11.0	22.0 18.0 -moz-	6.1 -webkit-	12.1 -webkit-
Multi-line flexbox	29.0 21.0 -webkit-	11.0	28.0	6.1 -webkit-	17.0 15.0 -webkit- 12.1

CSS3 弹性盒子内容

弹性盒子由弹性容器(Flex container)和弹性子元素(Flex item)组成。

弹性容器通过设置 display 属性的值为 flex 或 inline-flex将其定义为弹性容器。

弹性容器内包含了一个或多个弹性子元素。

注意： 弹性容器外及弹性子元素内是正常渲染的。弹性盒子只定义了弹性子元素如何在弹性容器内布局。

弹性子元素通常在弹性盒子内一行显示。默认情况每个容器只有一行。

以下元素展示了弹性子元素在一行内显示，从左到右:

实例

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
display: -webkit-flex;
display: flex;
width: 400px;
height: 250px;
background-color: lightgrey;
}
.flex-item {
background-color: cornflowerblue;
width: 100px;
height: 100px;
margin: 10px;
}
</style>
</head>
<body>
<div class="flex-container">
<div class="flex-item">flex item 1</div>
<div class="flex-item">flex item 2</div>
<div class="flex-item">flex item 3</div>
</div>
</body>
</html>
```

尝试一下 »

当然我们可以修改排列方式。

如果我们设置 `direction` 属性为 `rtl` (right-to-left),弹性子元素的排列方式也会改变，页面布局也跟着改变:

实例

```
body {
direction: rtl;
}
.flex-container {
display: -webkit-flex;
display: flex;
width: 400px;
height: 250px;
background-color: lightgrey;
}
.flex-item {
background-color: cornflowerblue;
width: 100px;
```

```
height: 100px;
margin: 10px;
}
```

尝试一下 »

flex-direction

flex-direction 属性指定了弹性子元素在父容器中的位置。

语法

```
flex-direction: row | row-reverse | column | column-reverse
```

flex-direction的值有:

- row：横向从左到右排列（左对齐），默认的排列方式。
- row-reverse：反转横向排列（右对齐，从后往前排，最后一项排在最前面。
- column：纵向排列。
- column-reverse：反转纵向排列，从后往前排，最后一项排在最上面。

以下实例演示了 row-reverse 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-flex-direction: row-reverse;
flex-direction: row-reverse;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 column 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-flex-direction: column;
flex-direction: column;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 column-reverse 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-flex-direction: column-reverse;
flex-direction: column-reverse;
width: 400px;
height: 250px;
```

```
background-color: lightgrey;
}
```

尝试一下 »

justify-content 属性

内容对齐（justify-content）属性应用在弹性容器上，把弹性项沿着弹性容器的主轴线（main axis）对齐。

justify-content 语法如下：

```
justify-content: flex-start | flex-end | center | space-between | space-around
```

各个值解析:

flex-start:

弹性项目向行头紧挨着填充。这个是默认值。第一个弹性项的main-start外边距边线被放置在该行的main-start边线，而后续弹性项依次平齐摆放。

flex-end:

弹性项目向行尾紧挨着填充。第一个弹性项的main-end外边距边线被放置在该行的main-end边线，而后续弹性项依次平齐摆放。

center:

弹性项目居中紧挨着填充。（如果剩余的自由空间是负的，则弹性项目将在两个方向上同时溢出）。

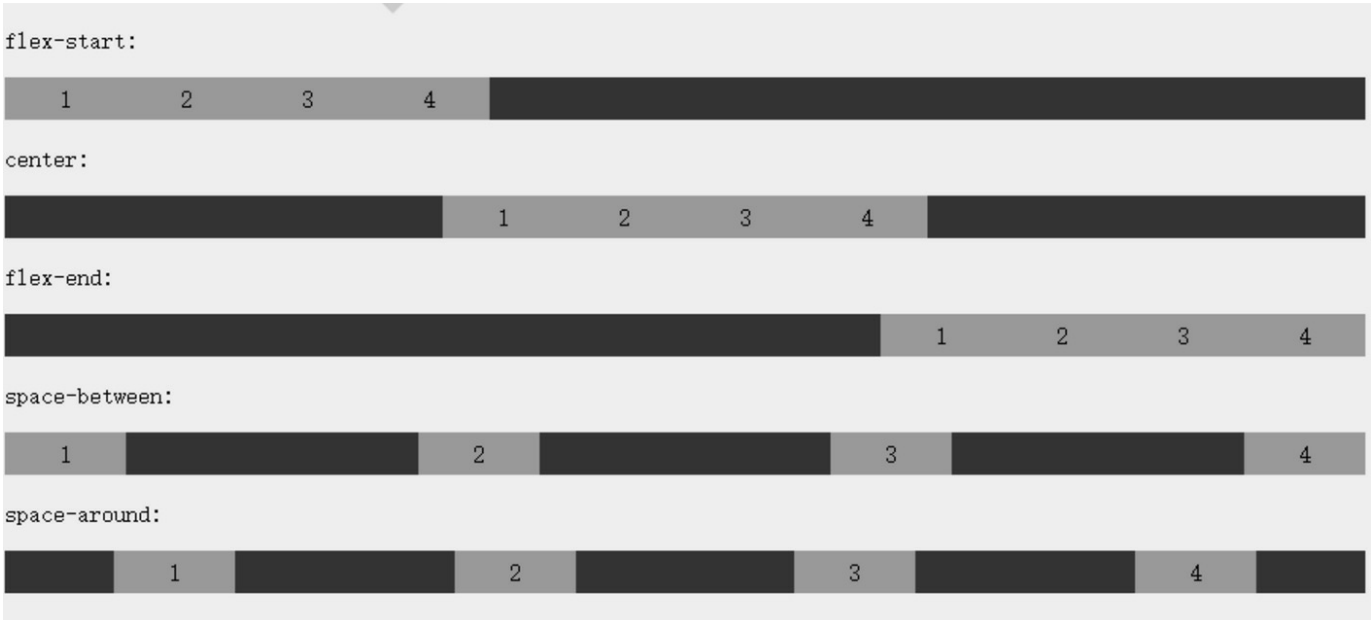
space-between:

弹性项目平均分布在该行上。如果剩余空间为负或者只有一个弹性项，则该值等同于flex-start。否则，第1个弹性项的外边距和行的main-start边线对齐，而最后1个弹性项的外边距和行的main-end边线对齐，然后剩余的弹性项分布在该行上，相邻项目的间隔相等。

space-around:

弹性项目平均分布在该行上，两边留有一半的间隔空间。如果剩余空间为负或者只有一个弹性项，则该值等同于center。否则，弹性项目沿该行分布，且彼此间隔相等（比如是20px），同时首尾两边和弹性容器之间留有一半的间隔（1/2*20px=10px）。

效果图展示:



以下实例演示了 flex-end 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-justify-content: flex-end;
justify-content: flex-end;
width: 400px;
height: 250px;
```

```
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 center 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-justify-content: center;
justify-content: center;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 space-between 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-justify-content: space-between;
justify-content: space-between;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 space-around 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-justify-content: space-around;
justify-content: space-around;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

align-items 属性

align-items 设置或检索弹性盒子元素在侧轴（纵轴）方向上的对齐方式。

语法

```
align-items: flex-start | flex-end | center | baseline | stretch
```

各个值解析:

- flex-start：弹性盒子元素的侧轴（纵轴）起始位置的边界紧靠住该行的侧轴起始边界。
- flex-end：弹性盒子元素的侧轴（纵轴）起始位置的边界紧靠住该行的侧轴结束边界。

center：弹性盒子元素在该行的侧轴（纵轴）上居中放置。（如果该行的尺寸小于弹性盒子元素的尺寸，则会向两个方向溢出相同的长度）。

baseline：如弹性盒子元素的行内轴与侧轴为同一条，则该值与'flex-start'等效。其它情况下，该值将参与基线对齐。

stretch：如果指定侧轴大小的属性值为'auto'，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照'min/max-width/height'属性的限制。

以下实例演示了 stretch(默认值) 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-align-items: stretch;
align-items: stretch;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 flex-start 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-align-items: flex-start;
align-items: flex-start;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 flex-end 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-align-items: flex-end;
align-items: flex-end;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 center 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-align-items: center;
align-items: center;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 baseline 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-align-items: baseline;
align-items: baseline;
width: 400px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

flex-wrap 属性

flex-wrap 属性用于指定弹性盒子的子元素换行方式。

语法

```
flex-wrap: nowrap|wrap|wrap-reverse|initial|inherit;
```

各个值解析:

nowrap - 默认， 弹性容器为单行。该情况下弹性子项可能会溢出容器。

wrap - 弹性容器为多行。该情况下弹性子项溢出的部分会被放置到新行，子项内部会发生断行

wrap-reverse -反转 wrap 排列。

以下实例演示了 nowrap 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-flex-wrap: nowrap;
flex-wrap: nowrap;
width: 300px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 wrap 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-flex-wrap: wrap;
flex-wrap: wrap;
width: 300px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

以下实例演示了 wrap-reverse 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-flex-wrap: wrap-reverse;
flex-wrap: wrap-reverse;
width: 300px;
height: 250px;
background-color: lightgrey;
}
```

尝试一下 »

align-content 属性

align-content 属性用于修改 flex-wrap 属性的行为。类似于 align-items, 但它不是设置弹性子元素的对齐，而是设置各个行的对齐。

语法

```
align-content: flex-start | flex-end | center | space-between | space-around | stretch
```

各个值解析:

stretch - 默认。各行将会伸展以占用剩余的空间。

flex-start - 各行向弹性盒容器的起始位置堆叠。

flex-end - 各行向弹性盒容器的结束位置堆叠。

center -各行向弹性盒容器的中间位置堆叠。

space-between -各行在弹性盒容器中平均分布。

space-around - 各行在弹性盒容器中平均分布，两端保留子元素与子元素之间间距大小的一半。

以下实例演示了 center 的使用:

实例

```
.flex-container {
display: -webkit-flex;
display: flex;
-webkit-flex-wrap: wrap;
flex-wrap: wrap;
-webkit-align-content: center;
align-content: center;
width: 300px;
height: 300px;
background-color: lightgrey;
}
```

尝试一下 »

弹性子元素属性

排序

语法

```
order:
```

各个值解析:

<integer>：用整数值来定义排列顺序，数值小的排在前面。可以为负值。

order 属性设置弹性容器内弹性子元素的属性:

实例

```
.flex-item {
background-color: cornflowerblue;
width: 100px;
height: 100px;
margin: 10px;
}
.first {
-webkit-order: -1;
order: -1;
}
```

尝试一下 »

对齐

设置"margin"值为"auto"值，自动获取弹性容器中剩余的空间。所以设置垂直方向margin值为"auto"，可以使弹性子元素在弹性容器的两上轴方向都完全居中。

以下实例在第一个弹性子元素上设置了 margin-right: auto; 。 它将剩余的空间放置在元素的右侧：

实例

```
.flex-item {
background-color: cornflowerblue;
width: 75px;
height: 75px;
margin: 10px;
}
.flex-item:first-child {
margin-right: auto;
}
```

尝试一下 »

完美的居中

以下实例将完美解决我们平时碰到的居中问题。

使用弹性盒子，居中变的很简单，只想要设置 margin: auto; 可以使得弹性子元素在两上轴方向上完全居中:

实例

```
.flex-item {
background-color: cornflowerblue;
width: 75px;
height: 75px;
margin: auto;
}
```

尝试一下 »

align-self

align-self 属性用于设置弹性元素自身在侧轴（纵轴）方向上的对齐方式。

语法

```
align-self: auto | flex-start | flex-end | center | baseline | stretch
```

各个值解析:

auto：如果'align-self'的值为'auto'，则其计算值为元素的父元素的'align-items'值，如果其没有父元素，则计算值为'stretch'。

flex-start：弹性盒子元素的侧轴（纵轴）起始位置的边界紧靠住该行的侧轴起始边界。

flex-end：弹性盒子元素的侧轴（纵轴）起始位置的边界紧靠住该行的侧轴结束边界。

center：弹性盒子元素在该行的侧轴（纵轴）上居中放置。（如果该行的尺寸小于弹性盒子元素的尺寸，则会向两个方向溢出相同的长度）。

baseline：如弹性盒子元素的行内轴与侧轴为同一条，则该值与'flex-start'等效。其它情况下，该值将参与基线对齐。

stretch：如果指定侧轴大小的属性值为'auto'，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照'min/max-width/height'属性的限制。

以下实例演示了弹性子元素上 align-self 不同值的应用效果：

实例

```
.flex-item {
background-color: cornflowerblue;
width: 60px;
min-height: 100px;
margin: 10px;
}
.item1 {
-webkit-align-self: flex-start;
align-self: flex-start;
}
.item2 {
-webkit-align-self: flex-end;
align-self: flex-end;
}
.item3 {
-webkit-align-self: center;
align-self: center;
}
.item4 {
-webkit-align-self: baseline;
align-self: baseline;
}
.item5 {
-webkit-align-self: stretch;
align-self: stretch;
}
```

尝试一下 »

flex

flex 属性用于指定弹性子元素如何分配空间。

语法

```
flex: auto | initial | none | inherit |  [ flex-grow ] || [ flex-shrink ] || [ flex-basis ]
```

各个值解析:

auto: 计算值为 1 1 auto

initial: 计算值为 0 1 auto

none：计算值为 0 0 auto

inherit：从父元素继承

[flex-grow]：定义弹性盒子元素的扩展比率。

[flex-shrink]：定义弹性盒子元素的收缩比率。

[flex-basis]：定义弹性盒子元素的默认基准值。

以下实例中，第一个弹性子元素占用了 2/4 的空间，其他两个各占 1/4 的空间:

实例

```
.flex-item {
background-color: cornflowerblue;
margin: 10px;
}
.item1 {
-webkit-flex: 2;
flex: 2;
}
.item2 {
-webkit-flex: 1;
flex: 1;
}
.item3 {
-webkit-flex: 1;
flex: 1;
}
```

尝试一下 »

实例

更多实例

使用弹性盒子创建响应式页面

CSS3 弹性盒子属性

下表列出了在弹性盒子中常用到的属性:

属性	描述
display	指定 HTML 元素盒子类型。
flex-direction	指定了弹性容器中子元素的排列方式
justify-content	设置弹性盒子元素在主轴（横轴）方向上的对齐方式。
align-items	设置弹性盒子元素在侧轴（纵轴）方向上的对齐方式。
flex-wrap	设置弹性盒子的子元素超出父容器时是否换行。
align-content	修改 flex-wrap 属性的行为，类似 align-items, 但不是设置子元素对齐，而是设置行对齐
flex-flow	flex-direction 和 flex-wrap 的简写
order	设置弹性盒子的子元素排列顺序。
align-self	在弹性子元素上使用。覆盖容器的 align-items 属性。
flex	设置弹性盒子的子元素如何分配空间。

CSS3 框大小 CSS3 多媒体查询

点我分享笔记

反馈/建议

CSS3 多媒体查询

CSS2 多媒体类型

@media 规则在 CSS2 中有介绍，针对不同媒体类型可以定制不同的样式规则。

例如：你可以针对不同的媒体类型(包括显示器、便携设备、电视机，等等)设置不同的样式规则。

但是这些多媒体类型在很多设备上支持还不够友好。

CSS3 多媒体查询

CSS3 的多媒体查询继承了 CSS2 多媒体类型的所有思想： 取代了查找设备的类型，CSS3 根据设置自适应显示。

媒体查询可用于检测很多事情，例如：

viewport(视窗) 的宽度与高度

设备的宽度与高度

朝向 (智能手机横屏，竖屏) 。

分辨率

目前很多针对苹果手机，Android 手机，平板等设备都会使用到多媒体查询。

浏览器支持

表格中的数字表示支持该属性的第一个浏览器的版本号。

属性					
@media	21.0	9.0	3.5	4.0	9.0

多媒体查询语法

多媒体查询由多种媒体组成，可以包含一个或多个表达式，表达式根据条件是否成立返回 true 或 false。

```
@media not|only mediatype and (expressions) {
CSS 代码...;
}
```

如果指定的多媒体类型匹配设备类型则查询结果返回 true，文档会在匹配的设备上显示指定样式效果。

除非你使用了 not 或 only 操作符，否则所有的样式会适应在所有设备上显示效果。

not: not是用来排除掉某些特定的设备的，比如 @media not print（非打印设备）。

only: 用来定某种特别的媒体类型。对于支持Media Queries的移动设备来说，如果存在only关键字，移动设备的Web浏览器会忽略only关键字并直接根据后面的表达式应用样式文件。对于不支持Media Queries的设备但能够读取Media Type类型的Web浏览器，遇到only关键字时会忽略这个样式文件。

all: 所有设备，这个应该经常看到。

你也可以在不同的媒体上使用不同的样式文件：

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">
```

CSS3 多媒体类型

值	描述
all	用于所有多媒体类型设备
print	用于打印机
screen	用于电脑屏幕，平板，智能手机等。
speech	用于屏幕阅读器

多媒体查询简单实例

使用多媒体查询可以在指定的设备上使用对应的样式替代原有的样式。

以下实例中在屏幕可视窗口尺寸大于 480 像素的设备上修改背景颜色:

实例

```
@media screen and (min-width: 480px) {
    body {
        background-color: lightgreen;
    }
}
```

尝试一下 »

以下实例在屏幕可视窗口尺寸大于 480 像素时将菜单浮动到页面左侧：

实例

```
@media screen and (min-width: 480px) {
    #leftsidebar {width: 200px; float: left;}
    #main {margin-left:216px;}
}
```

尝试一下 »

CSS3 @media 参考

更多多媒体查询内容可以参考 @media 规则。

CSS3 弹性盒子 CSS3 多媒体查询实例

点我分享笔记

反馈/建议

CSS3 多媒体查询 CSS3 rotation-point

CSS3 多媒体查询实例

本章节我们将为大家演示一些多媒体查询实例。

开始之前我们先制作一个电子邮箱的链接列表。HTML 代码如下：

实例 1

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style-type: none;
}

ul li a {
    color: green;
    text-decoration: none;
    padding: 3px;
    display: block;
}
</style>
</head>
<body>

<ul>
  <li><a data-email="johndoe@example.com" href="mailto:johndoe@example.com">John Doe</a></li>
  <li><a data-email="marymoe@example.com" href="mailto:marymoe@example.com">Mary Moe</a></li>
  <li><a data-email="amandapanda@example.com" href="mailto:amandapanda@example.com">Amanda Panda</a></li>
</ul>

</body>
</html>
```

尝试一下 »

注意 data-email 属性。在 HTML 中我们可以使用带 data- 前缀的属性来存储信息。

520 到 699px 宽度 - 添加邮箱图标

当浏览器的宽度在 520 到 699px, 邮箱链接前添加邮件图标：

实例 2

```
@media screen and (max-width: 699px) and (min-width: 520px) {
    ul li a {
        padding-left: 30px;
        background: url(email-icon.png) left center no-repeat;
    }
}
```

尝试一下 »

700 到 1000px - 添加文本前缀信息

当浏览器的宽度在 700 到 1000px, 会在邮箱链接前添加 "Email: ":

实例 3

```
@media screen and (max-width: 1000px) and (min-width: 700px) {
    ul li a:before {
        content: "Email: ";
        font-style: italic;
        color: #666666;
    }
}
```

尝试一下 »

大于 1001px 宽度 - 添加邮件地址

当浏览器的宽度大于 1001px 时，会在链接后添加邮件地址接。

我们会使用 data- 属性来为每个人名后添加邮件地址：

实例 4

```
@media screen and (min-width: 1001px) {
    ul li a:after {
        content: " (" attr(data-email) ")";
        font-size: 12px;
        font-style: italic;
        color: #666666;
    }
}
```

尝试一下 »

大于 1151px 宽度 - 添加图标

当浏览器的宽度大于 1001px 时，会在人名前添加图标。

实例中，我们没有编写额外的查询块，我们可以在已有的查询媒体后使用逗号分隔来添加其他媒体查询 (类似 OR 操作符):

实例 5

```
@media screen and (max-width: 699px) and (min-width: 520px), (min-width: 1151px) {
    ul li a {
        padding-left: 30px;
        background: url(email-icon.png) left center no-repeat;
    }
}
```

尝试一下 »

实例

更多实例

在一个网页的侧栏上使用邮件列表链接

该实例在网页的左侧栏添加了邮件链接列表。

CSS3 多媒体查询 CSS3 rotation-point

点我分享笔记

反馈/建议