**CSS3新增了哪些新特性？**

|  |
| --- |
| https://camo.githubusercontent.com/176aeca505f99a27fa7f37cebffeddebddeda2488d29a4ed167d4693da875fdc/68747470733a2f2f7374617469632e7675652d6a732e636f6d2f64353866366466302d396235652d313165622d616239302d6439616538313462323430642e706e67 |

## 一、是什么

css，即层叠样式表（Cascading Style Sheets）的简称，是一种标记语言，由浏览器解释执行用来使页面变得更美观

css3是css的最新标准，是向后兼容的，CSS1/2 的特性在 CSS3 里都是可以使用的

而 CSS3 也增加了很多新特性，为开发带来了更佳的开发体验

## 二、选择器

css3中新增了一些选择器，主要为如下图所示：

[](https://camo.githubusercontent.com/36d7517e24a6c49d584314f814f738eb9b93d11bf0f8ded20a1922fd00e5ef98/68747470733a2f2f7374617469632e7675652d6a732e636f6d2f65333638636632302d396235652d313165622d383566362d3666616337376330633962332e706e67)

## 三、新样式

## 边框

## css3新增了三个边框属性，分别是：

### border-radius：创建圆角边框

#### 可以设置所有圆角也可以只设置一个圆角如：

#### 左上角圆角：

|  |
| --- |
| {border-top-left-radius:30px;} 效果： |

#### 右上角：

|  |
| --- |
| {border-top-right-radius:50px;}效果： |

#### 右下角：

|  |
| --- |
| {border-bottom-right-radius:50px;} |

#### 左下角

|  |
| --- |
| {border-bottom-left-radius:50px;} |

#### 可以将上面四个组成一个复合写法

|  |
| --- |
| {border-radius：左上角 右上角 右下角 左下角;} |

#### 可以只写一个值，表示所有的位置都使用这个值

|  |
| --- |
| {border-radius：30px;} 左上角 右上角 右下角 左下角30px |

#### 也可以写2个值，第一个值设置给左上角和右下角，第二个值设置给右上角和左下角

#### 还可以写3个值，分别设置左上角，右上角和左下角 右下角

## 还可以设置两个值用”/”分割，绘制椭圆圆角

|  |
| --- |
| { border-radius：10px/50px;} |
|  |

#### 也可以两组四个数，用”/”隔开

#### 注意：也可以设置百分比，如果给一个正方形设置{border-radius:50%;}它就变成一个圆

### 实例

|  |  |
| --- | --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>          .box{              width: 300px;              height: 300px;              background-color: pink;              /\* border-radius: 30px 60px 30px; \*/              /\* border-top-left-radius: 30px;              border-top-right-radius: 30px;              border-bottom-right-radius: 30px;              border-bottom-left-radius: 30px; \*/              border-radius: 50%;          }      </style>  </head>  <body>      <div class="box"></div>  </body>  </html> | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>          .box{              width: 300px;              height: 300px;              background-color: pink;              border-radius: 30px 60px 30px;              /\* border-top-left-radius: 30px;              border-top-right-radius: 30px;              border-bottom-right-radius: 30px;              border-bottom-left-radius: 30px; \*/              /\* border-radius: 50%; \*/          }      </style>  </head>  <body>      <div class="box"></div>  </body>  </html> |
|  |  |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>          .box{              width: 300px;              height: 300px;              background-color: pink;              border-radius: 30px 60px;              /\* border-radius: 30px; \*/              /\* border-radius: 30px 60px 30px; \*/              /\* border-top-left-radius: 30px;              border-top-right-radius: 30px;              border-bottom-right-radius: 30px;              border-bottom-left-radius: 30px; \*/              /\* border-radius: 50%; \*/          }      </style>  </head>  <body>      <div class="box"></div>  </body>  </html> | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>          .box{              width: 300px;              height: 300px;              background-color: pink;              border-radius: 30px;              /\* border-radius: 30px 60px 30px; \*/              /\* border-top-left-radius: 30px;              border-top-right-radius: 30px;              border-bottom-right-radius: 30px;              border-bottom-left-radius: 30px; \*/              /\* border-radius: 50%; \*/          }      </style>  </head>  <body>      <div class="box"></div>  </body>  </html> |
|  |  |

### box-shadow：为元素添加阴影-盒子阴影

#### box-shadow按顺时针方向来设置四个角落

#### 语法：

|  |
| --- |
| /\* x 偏移量 | y 偏移量 | 阴影颜色 \*/  box-shadow: 60px -16px teal;  /\* x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影颜色 \*/  box-shadow: 10px 5px 5px black;  /\* x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/  box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);  /\* 插页 (阴影向内) | x 偏移量 | y 偏移量 | 阴影颜色 \*/  box-shadow: inset 5em 1em gold;  /\* 任意数量的阴影，以逗号分隔 \*/  box-shadow: 3px 3px red, -1em 0 0.4em olive;  /\* 全局关键字 \*/  box-shadow: inherit;  box-shadow: initial;  box-shadow: unset;  Copy to Clipboard  向元素添加单个 box-shadow 效果时使用以下规则：   * 当给出两个、三个或四个 [<length>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/length)值时。   + 如果只给出两个值，那么这两个值将会被当作 <offset-x><offset-y> 来解释。   + 如果给出了第三个值，那么第三个值将会被当作<blur-radius>解释。   + 如果给出了第四个值，那么第四个值将会被当作<spread-radius>来解释。 * 可选，inset关键字。 * 可选，<color>值。   若要对同一个元素添加多个阴影效果，请使用逗号将每个阴影规则分隔开。 |

设置元素阴影，设置属性如下：

* 水平阴影
* 垂直阴影
* 模糊距离(虚实)
* 阴影尺寸(影子大小)
* 阴影颜色
* 内/外阴影

其中水平阴影和垂直阴影是必须设置的

#### 实例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>             .div1{              width: 300px;              height: 300px;              background-color: pink;              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              box-shadow: 3px 3px 2px 2px gray;           }      </style>  </head>  <body>      <div class="box div1"></div>  </body>  </html> |
|  |

#### 实例2.实际开发中的常用阴影效果

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>           .div1{              width: 300px;              height: 300px;              background-color: pink;              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              box-shadow: 0px 3px 5px 0px #888;           }      </style>  </head>  <body>      <div class="box div1"></div>  </body>  </html> |
|  |

#### 还可以实现鼠标悬停时候出现阴影的效果

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>           .div1{              width: 300px;              height: 300px;              background-color: pink;              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              /\* box-shadow: 0px 3px 5px 0px #888; \*/           }           .div1:hover{              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              box-shadow: 0px 3px 5px 0px #888;           }      </style>  </head>  <body>      <div class="box div1"></div>  </body>  </html> |
| |  |  | | --- | --- | | 悬停前： | 悬停时： | |

#### 还可以给阴影添加过渡效果

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>           .div1{              width: 300px;              height: 300px;              background-color: pink;              transition: box-shadow 1s;           }           .div1:hover{              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              box-shadow: 0px 3px 5px 0px #888;           }      </style>  </head>  <body>      <div class="box div1"></div>  </body>  </html> |
| 效果:鼠标移动到元素上面，会在1秒内慢慢出现阴影，鼠标离阴影开会在1秒内慢慢消失 |

#### 还可以和伪类:active配合使用实现按钮的效果

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>           .div1{              width: 300px;              height: 300px;              background-color: pink;              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              /\* box-shadow: 0px 3px 5px 0px #888; \*/              transition: box-shadow 0.6s;           }           .div1:hover{              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              box-shadow: 0px 3px 5px 0px #888;           }             .div1:active{              box-shadow:none;           }      </style>  </head>  <body>      <div class="box div1"></div>  </body>  </html> |
| 出现阴影后按下左键阴影显示松开左键阴影回来（前提是鼠标还在元素上面） |

#### border-shadow还有第六给值，如果设置为inset就是将他设置为内阴影，如果没有默认是外阴影

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>           .div1{              width: 300px;              height: 300px;              background-color: pink;                /\* box-shadow: 0px 3px 5px 0px #888; \*/              transition: box-shadow 1s;           }           .div1:hover{              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              /\* box-shadow: 0px 3px 5px 0px #888; \*/              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色  是否内阴影（inset是不写是否）\*/              box-shadow: 0px 3px 5px 0px #888 inset;           }             .div1:active{              box-shadow:none;           }      </style>  </head>  <body>      <div class="box div1"></div>  </body>  </html> |
|  |

### border-image：使用图片来绘制边框

### 背景

新增了几个关于背景的属性，分别是background-clip、background-origin、background-size和background-break

#### background-clip

用于确定背景画区，有以下几种可能的属性：

* background-clip: border-box; 背景从border开始显示
* background-clip: padding-box; 背景从padding开始显示
* background-clip: content-box; 背景显content区域开始显示
* background-clip: no-clip; 默认属性，等同于border-box

通常情况，背景都是覆盖整个元素的，利用这个属性可以设定背景颜色或图片的覆盖范围

#### background-origin

当我们设置背景图片时，图片是会以左上角对齐，但是是以border的左上角对齐还是以padding的左上角或者content的左上角对齐? border-origin正是用来设置这个的

* background-origin: border-box; 从border开始计算background-position
* background-origin: padding-box; 从padding开始计算background-position
* background-origin: content-box; 从content开始计算background-position

默认情况是padding-box，即以padding的左上角为原点

#### background-size

background-size属性常用来调整背景图片的大小，主要用于设定图片本身。有以下可能的属性：

* background-size: contain; 缩小图片以适合元素（维持像素长宽比）
* background-size: cover; 扩展元素以填补元素（维持像素长宽比）
* background-size: 100px 100px; 缩小图片至指定的大小
* background-size: 50% 100%; 缩小图片至指定的大小，百分比是相对包 含元素的尺寸

### background-break

元素可以被分成几个独立的盒子（如使内联元素span跨越多行），background-break 属性用来控制背景怎样在这些不同的盒子中显示

* background-break: continuous; 默认值。忽略盒之间的距离（也就是像元素没有分成多个盒子，依然是一个整体一样）
* background-break: bounding-box; 把盒之间的距离计算在内；
* background-break: each-box; 为每个盒子单独重绘背景

### 可以将所有的背景实现合成为一个复合写法

#### { background : 颜色 图片url repeat/no-repeat x轴位置 y轴位置 / x轴大小 y轴大小；}

### 文字

### word-wrap

语法：word-wrap: normal|break-word

* normal：使用浏览器默认的换行
* break-all：允许在单词内换行

### text-overflow

text-overflow设置或检索当当前行超过指定容器的边界时如何显示，属性有两个值选择：

* clip：修剪文本
* ellipsis：显示省略符号来代表被修剪的文本

### text-shadow文字阴影

text-shadow可向文本应用阴影。能够规定水平阴影、垂直阴影、模糊距离，以及阴影的颜色

#### text-shadow

**text-shadow** 为文字添加阴影。可以为文字与 [decoration](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-decoration) 添加多个阴影，阴影值之间用逗号隔开。每个阴影值由元素在 X 和 Y 方向的偏移量、模糊半径和颜色值组成。

##### [语法](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-shadow#%E8%AF%AD%E6%B3%95)

**/\* offset-x | offset-y | blur-radius | color \*/**

**text-shadow: 1px 1px 2px black;**

**/\* color | offset-x | offset-y | blur-radius \*/**

**text-shadow: #fc0 1px 0 10px;**

**/\* offset-x | offset-y | color \*/**

**text-shadow: 5px 5px #558abb;**

**/\* color | offset-x | offset-y \*/**

**text-shadow: white 2px 5px;**

**/\* offset-x | offset-y**

**/\* Use defaults for color and blur-radius \*/**

**text-shadow: 5px 10px;**

**/\* Global values \*/**

**text-shadow: inherit;**

**text-shadow: initial;**

**text-shadow: unset;**

**Copy to Clipboard**

当阴影大于一个时要用逗号区别开阴影之间的参数。

每个阴影都有两到三个 <length> 参数，以及一个与阴影颜色相关的 <color> 参数。前两个 <length> 参数，是以“文字中心”为原点的坐标轴，分别为 x 轴 <offset-x> 和 y 轴 <offset-y> 的值；如果有第三个 <length> 参数，则第三个数值为形成阴影效果的半径的数值 <blur-radius>。

当所给的阴影大于一个时，阴影应用的顺序为从前到后，第一个指定的阴影在顶部。

这个属性同时适用于 [::first-line](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::first-line) 以及 [::first-letter](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::first-letter) [伪元素](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Pseudo-elements)。

##### [值](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-shadow#%E5%80%BC)

[<color>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/color_value)

可选。阴影的颜色。可以在偏移量之前或之后指定。如果没有指定颜色，则使用用户代理自行选择的颜色，因此需要跨浏览器的一致性时，应该明确定义它。

<offset-x> <offset-y>

必选。这些 [<length>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/length) 值指定阴影相对文字的偏移量。<offset-x> 指定水平偏移量，若是负值则阴影位于文字左侧。<offset-y> 指定垂直偏移量，若是负值则阴影位于文字上方。如果两者均为 0，则阴影位于文字正后方，如果设置了 <blur-radius> 则会产生模糊效果。

<blur-radius>

可选。这是 [<length>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/length) 值。如果没有指定，则默认为 0。值越大，模糊半径越大，阴影也就越大越淡（wider and lighter）。

#### [形式定义](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-shadow#%E5%BD%A2%E5%BC%8F%E5%AE%9A%E4%B9%89)

|  |  |
| --- | --- |
| [**初始值**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/initial_value) | none |
| **适用元素** | all elements. It also applies to [::first-letter](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::first-letter) and [::first-line](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::first-line). |
| [**是否是继承属性**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Inheritance) | yes |
| [**计算值**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/computed_value) | a color plus three absolute lengths |
| **Animation type** | a [shadow list](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow#interpolation) |

#### [形式语法](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-shadow#%E5%BD%A2%E5%BC%8F%E8%AF%AD%E6%B3%95)

text-shadow =   
 none [|](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#single_bar)  
 [[](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#brackets) [<color>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/color_value)[?](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#question_mark) [&&](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#double_ampersand) [<length>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/length)[{2,3}](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#curly_braces) []](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#brackets)[#](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#hash_mark)

#### [示例](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-shadow#%E7%A4%BA%E4%BE%8B)

#### [简单的阴影](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-shadow#%E7%AE%80%E5%8D%95%E7%9A%84%E9%98%B4%E5%BD%B1)

.red-text-shadow {

text-shadow: red 0 -2px;

}

Copy to Clipboard

<p class="red-text-shadow">

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo

inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

</p>

Copy to Clipboard

#### [多重阴影](https://developer.mozilla.org/zh-CN/docs/Web/CSS/text-shadow#%E5%A4%9A%E9%87%8D%E9%98%B4%E5%BD%B1)

.white-with-blue-shadow {

text-shadow: 1px 1px 2px black, 0 0 1em blue, 0 0 0.2em blue;

color: white;

font: 1.5em Georgia, "Bitstream Charter", "URW Bookman L", "Century Schoolbook L", serif;

}

Copy to Clipboard

<p class="white-with-blue-shadow">

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore

veritatis et quasi architecto beatae vitae dicta sunt explicabo.

</p>

#### 小技巧：如果把x轴和y轴上面的偏移量都设置为0，然后设置一个红色，就会有类似于霓虹灯的效果

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>圆角阴影不透明度渐变</title>      <style>          /\* .box{              width: 300px;              height: 300px;              background-color: pink;              border-radius: 30px 60px;              /\* border-radius: 30px; \*/              /\* border-radius: 30px 60px 30px; \*/              /\* border-top-left-radius: 30px;              border-top-right-radius: 30px;              border-bottom-right-radius: 30px;              border-bottom-left-radius: 30px; \*/              /\* border-radius: 50%;          }\*/           .div1{              width: 300px;              height: 300px;              /\* background-color: pink; \*/                /\* box-shadow: 0px 3px 5px 0px #888; \*/              transition: box-shadow 1s;           }           .div1:hover{              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/              box-shadow: 0px 3px 5px 0px #888;              /\* 设置阴影 参数：x 偏移量 | y 偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色  是否内阴影（inset是不写是否）\*/              /\* box-shadow: 0px 3px 5px 0px #888 inset; \*/              /\* offset-x | offset-y | blur-radius | color  xy轴的偏移是必选其他是可选颜色可以在最前面或者最后面\*/              /\* text-shadow: 1px 2px #ccc; \*/             }           .div1 span{               /\* 把x轴和y轴上面的偏移量都设置为0，然后设置一个红色，就会有类似于霓虹灯的效果 \*/              text-shadow: 0px 0 2px red;              font-size: 50px;           }             .div1:active{              box-shadow:none;           }      </style>  </head>  <body>      <div class="box div1"><span>文字阴影效果</span></div>  </body>  </html> |
|  |

##### 偏移量可以是负数，他会适当阴影左移

##### 文字阴影可以实现类似于描边的效果

### text-decoration

CSS3里面开始支持对文字的更深层次的渲染，具体有三个属性可供设置：

* text-fill-color: 设置文字内部填充颜色
* text-stroke-color: 设置文字边界填充颜色
* text-stroke-width: 设置文字边界宽度

### 颜色和不透明度

css3新增了新的颜色表示方式rgba与hsla

* rgba分为两部分，rgb为颜色值，a为透明度
* hala分为四部分，h为色相，s为饱和度，l为亮度，a为透明度

#### opacity

#### [概述](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity#%E6%A6%82%E8%BF%B0)

opacity 属性指定了一个元素的**不透明度**。换言之，opacity 属性指定了一个元素后面的背景的被覆盖程度。

当 opacity 属性的值应用于某个元素上时，是把这个元素（包括它的内容）当成一个整体看待，即使这个值没有被子元素继承。因此，一个元素和它包含的子元素都会具有和元素背景相同的透明度，哪怕这个元素和它的子元素有不同的 opacity 属性值。

使用 opacity 属性，当属性值不为 1 时，会把元素放置在一个新的[层叠上下文](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context)中。

|  |  |
| --- | --- |
| [**初始值**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/initial_value) | 1 |
| **适用元素** | all elements |
| [**是否是继承属性**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Inheritance) | 否 |
| **Percentages** | map to the range [0,1] |
| [**计算值**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/computed_value) | The same as the specified value after clipping the [number](https://developer.mozilla.org/zh-CN/docs/Web/CSS/number) to the range [0.0, 1.0]. |
| **Animation type** | by computed value type |

#### [语法](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity#%E8%AF%AD%E6%B3%95)

/\* 完全不透明 \*/

opacity: 1;

opacity: 1.0;

/\* 半透明 \*/

opacity: 0.6;

/\* 完全透明 \*/

opacity: 0.0;

opacity: 0;

opacity: inherit;

Copy to Clipboard

##### [属性值](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity#%E5%B1%9E%E6%80%A7%E5%80%BC)

<number>

[<number>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/number) 是一个 0.0 到 1.0 范围内的数字值，这个数值既包含也代表通道的透明度，也就是 alpha 通道的值。任何一个溢出这个取值区间的值，尽管有效，但会被解析为在取值范围内最靠近它的值。

| **值** | **释义** |
| --- | --- |
| 0 | 元素完全透明 (即元素不可见). |
| 任何一个位于 0.0-1.0 之间的 [<number>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/number) | 元素半透明 (即元素后面的背景可见). |
| 1 | 元素完全不透明 (即元素后面的背景不可见). |

##### [形式语法](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity#%E5%BD%A2%E5%BC%8F%E8%AF%AD%E6%B3%95)

opacity =   
 <alpha-value>   
  
<alpha-value> =   
 [<number>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/number) [|](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax#single_bar)  
 [<percentage>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/percentage)

#### [示例](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity#%E7%A4%BA%E4%BE%8B)

##### [基本示例](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity#%E5%9F%BA%E6%9C%AC%E7%A4%BA%E4%BE%8B)

div { background-color: yellow; }

.light {

opacity: 0.2; /\* Barely see the text over the background \*/

}

.medium {

opacity: 0.5; /\* See the text more clearly over the background \*/

}

.heavy {

opacity: 0.9; /\* See the text very clearly over the background \*/

}

Copy to Clipboard

<div class="light">You can barely see this.</div>

<div class="medium">This is easier to see.</div>

<div class="heavy">This is very easy to see.</div>

Copy to Clipboard

##### [:hover 时 opacity 的不同](https://developer.mozilla.org/zh-CN/docs/Web/CSS/opacity#hover_%E6%97%B6_opacity_%E7%9A%84%E4%B8%8D%E5%90%8C)

img.opacity {

opacity: 1;

filter: alpha(opacity=100); /\* IE8 and lower \*/

zoom: 1; /\* Triggers "hasLayout" in IE 7 and lower \*/

}

img.opacity:hover {

opacity: 0.5;

filter: alpha(opacity=50);

zoom: 1;

}

<img src="//developer.mozilla.org/media/img/mdn-logo.png"

alt="MDN logo" width="128" height="146"

class="opacity">

#### rgba

用法：

#box{

background-color: rgba(0, 0, 0, .5);

}

RGBA 颜色值是 RGB 颜色值的扩展，带有一个 alpha 通道 - 它规定了对象的不透明度。

RGBA 颜色值得到以下浏览器的支持：IE9+、Firefox 3+、Chrome、Safari 以及 Opera 10+。

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）与 1.0（完全不透明）的数字。

**那么对于IE8及以下需要做以下兼容：**

#box{

filter: progid:DXImageTransform.Microsoft.Gradient(startColorStr=#80000000,endColorStr=#80000000);

}

其中：#88000000 的前两位数字控制透明度，取值16进制从00 -> FF（越小越透明），00表示完全透明，FF就是全不透明，后面六位是色值。

**※注意：**  
如果在IE9里面同时使用这RGBA两种方法时，会造成冲突而无法做到透明度的实现。  
而对于opacity是可以两个一起写，没有冲突问题！

##### ****opacity 和 rgba 的区别****

为此我们设置了两个盒子来作为对比

html代码：

<div id="box1">

<div class="pane"></div>

box1-opacity演示效果

</div>

<div id="box2">

<div class="pane"></div>

box2-rgba演示效果

</div>

css代码：

#box1{

width: 100px;

height: 100px;

color:black;

background-color:rgb(125,25,0);

opacity: 0.5;

}

#box2{

margin-top: 10px;

width: 100px;

height: 100px;

color:black;

background-color: rgba(125,25,0, .5);

}

.pane{

width: 20px;

height: 20px;

background-color:red;

}

结果如图：  


##### 从上面的结果我们可以看到 opacity 可以影响字体以及红色小方块的透明度，而 rgba 不会。如果需要给一个颜色设置不透明度，但是又不想它的子元素跟着一起变化，就可以使用rgba来设置，有些元素如图片等只能使用opcity来设置不透明度

##### 子元素会继承父元素的 opacity 属性

### 四、transition 过渡

transition属性可以被指定为一个或多个CSS属性的过渡效果，多个属性之间用逗号进行分隔，必须规定两项内容：

* 过度效果
* 持续时间

语法如下：

transition： CSS属性，花费时间，效果曲线(默认ease)，延迟时间(默认0)

上面为简写模式，也可以分开写各个属性

transition-property: width;

transition-duration: 1s;

transition-timing-function: linear;

transition-delay: 2s;

**使用 CSS 过渡**

**CSS 过渡**提供了一种在更改 CSS 属性时控制动画速度的方法。其可以让属性变化成为一个持续一段时间的过程，而不是立即生效的。比如，将一个元素的颜色从白色改为黑色，通常这个改变是立即生效的，使用 CSS 过渡后该元素的颜色将逐渐从白色变为黑色，按照一定的曲线速率变化。这个过程可以自定义。

通常将两个状态之间的过渡称为**隐式过渡**，因为开始与结束之间的状态由浏览器决定。

#### 可过渡样式

不是所有的CSS样式都可以过渡，只有具有中间值的属性才有过渡效果。

**颜色：color background-color border-color outline-color**

**位置：background-position left right top botton**

**长度：**

**[1]max-height min-height max-width min-width height width**

**[2]border-width margin padding outline-width outline-offset**

**[3]font-size line-height text-indent vertical-align**

**[4]border-spacing letter-spacing word-spacing**

**数字: opacity visibility z-index font-weight zoom**

**组合: text-shadow transform box-shadow clip**

**其他: gradient，2d转换**

#### 不能过渡的属性

##### display，overflow等等没有数字变化的属性

CSS 过渡可以决定哪些属性发生动画效果 (明确地列出这些属性)，何时开始 (设置 delay），持续多久 (设置 duration) 以及如何动画 (定义*timing function*，比如匀速地或先快后慢)。

[**哪些 CSS 属性可用于过渡？**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#%E5%93%AA%E4%BA%9B_css_%E5%B1%9E%E6%80%A7%E5%8F%AF%E7%94%A8%E4%BA%8E%E8%BF%87%E6%B8%A1%EF%BC%9F)

Web 作者可以定义哪一属性需以何种方式用于动画，由此允许创造复杂的过渡。然而因为为某些属性赋予动画无意义，所以这些属性[无动画性](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_animated_properties)。

**备注：** auto 值常常较复杂，规范指出不要在它上动画。一些用户代理，比如基于 Gecko 的，遵循这点；一些，比如基于 WebKit 的，没有这么严格限制。在 auto 上动画结果可能不可预期，这取决于浏览器及其版本，应当避免使用。

同时应当留意这种情形，在插入元素（如 .appendChild()）或改变属性 display: none 后立即使用过渡，元素将视为没有开始状态，始终处于结束状态。简单的解决办法，改变属性前用 window.setTimeout() 延迟几毫秒。

[**多个属性一起动画示例**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#%E5%A4%9A%E4%B8%AA%E5%B1%9E%E6%80%A7%E4%B8%80%E8%B5%B7%E5%8A%A8%E7%94%BB%E7%A4%BA%E4%BE%8B)

[**HTML**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#html)

<body>

<p>盒子的多个属性一起动画：width, height, background-color, transform. 将光标悬停在盒子上查看动画。</p>

<div class="box"></div>

</body>

[**CSS**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#css)

.box {

border-style: solid;

border-width: 1px;

display: block;

width: 100px;

height: 100px;

background-color: #0000FF;

-webkit-transition:width 2s, height 2s,

background-color 2s, -webkit-transform 2s;

transition:width 2s, height 2s, background-color 2s, transform 2s;

}

.box:hover {

background-color: #FFCCCC;

width:200px;

height:200px;

-webkit-transform:rotate(180deg);

transform:rotate(180deg);

}

[**定义过渡**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#%E5%AE%9A%E4%B9%89%E8%BF%87%E6%B8%A1)

CSS 过渡由简写属性 [transition](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition) 定义是最好的方式，可以避免属性值列表长度不一，节省调试时间。

也可以用下面子属性来控制过渡的各部分：

**(注意下面示例中过渡无限循环是为了说明例子，过渡只是从开始到结束变化，如果需要循环，查看**[**animation (en-US)**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation)**。)**

[transition-property](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition-property)

指定哪个或哪些 CSS 属性用于过渡。只有指定的属性才会在过渡中发生动画，其他属性仍如通常那样瞬间变化。

[transition-duration](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition-duration)

指定过渡的时长。或者为所有属性指定一个值，或者指定多个值，为每个属性指定不同的时长。

[transition-timing-function](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition-timing-function)

指定一个函数，定义属性值怎么变化。缓动函数 *Timing functions* 定义属性如何计算。多数 [timing functions (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/easing-function) 由四点定义一个 bezier 曲线。也可以从 [Easing Functions Cheat Sheet](https://easings.net/) 选择缓动效果。

[transition-delay](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition-delay)

指定延迟，即属性开始变化时与过渡开始发生时之间的时长。

简写语法：

div {

transition: <property> <duration> <timing-function> <delay>;

}

Copy to Clipboard

[**检测过渡是否完成**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#%E6%A3%80%E6%B5%8B%E8%BF%87%E6%B8%A1%E6%98%AF%E5%90%A6%E5%AE%8C%E6%88%90)

当过渡完成时触发一个事件，在符合标准的浏览器下，这个事件是 transitionend, 在 WebKit 下是 webkitTransitionEnd. 详情查看页面底部的兼容性表格。 transitionend 事件提供两个属性:

propertyName

字符串，指示已完成过渡的属性。

elapsedTime

浮点数，指示当触发这个事件时过渡已运行的时间（秒）。这个值不受 [transition-delay](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition-delay) 影响。

照例可以用 [addEventListener()](https://developer.mozilla.org/zh-CN/docs/Web/API/EventTarget/addEventListener) 方法来监听这个事件：

el.addEventListener("transitionend", updateTransition, true);

**备注：** 如果取消了过渡则不会触发 transitionend 事件，因为在过渡完成前动画的属性值改变了。

[**当属性值列表长度不一致时**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#%E5%BD%93%E5%B1%9E%E6%80%A7%E5%80%BC%E5%88%97%E8%A1%A8%E9%95%BF%E5%BA%A6%E4%B8%8D%E4%B8%80%E8%87%B4%E6%97%B6)

以 [transition-property](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition-property) 的值列表长度为标准，如果某个属性值列表长度短于它的，则重复其值以长度一致，例如：

div {

transition-property: opacity, left, top, height;

transition-duration: 3s, 5s;

}

将按下面这样处理：

div {

transition-property: opacity, left, top, height;

transition-duration: 3s, 5s, 3s, 5s;

}

类似地，如果某个属性的值列表长于 [transition-property](https://developer.mozilla.org/zh-CN/docs/Web/CSS/transition-property) 的，将被截短。例如：

div {

transition-property: opacity, left;

transition-duration: 3s, 5s, 2s, 1s;

}

将按下面这样处理：

div {

transition-property: opacity, left;

transition-duration: 3s, 5s;

}

[**简单例子**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#%E7%AE%80%E5%8D%95%E4%BE%8B%E5%AD%90)

这个例子实现这样的效果：4s 过渡改变字体大小，2s 延迟——在元素上鼠标悬停时与开始动画效果之间：

#delay1 {

position: relative;

transition-property: font-size;

transition-duration: 4s;

transition-delay: 2s;

font-size: 14px;

}

#delay1:hover {

transition-property: font-size;

transition-duration: 4s;

transition-delay: 2s;

font-size: 36px;

}

[**高亮菜单过渡效果**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#%E9%AB%98%E4%BA%AE%E8%8F%9C%E5%8D%95%E8%BF%87%E6%B8%A1%E6%95%88%E6%9E%9C)

CSS 的一个常用地方是当鼠标悬停在菜单上时高亮此菜单，使用 transition 效果更佳。

[**HTML**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#html_2)

<div class="sidebar">

<p><a class="menuButton" href="home">Home</a></p>

<p><a class="menuButton" href="about">About</a></p>

<p><a class="menuButton" href="contact">Contact Us</a></p>

<p><a class="menuButton" href="links">Links</a></p>

</div>

[**CSS**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#css_2)

.menuButton {

position: relative;

transition-property: background-color, color;

transition-duration: 1s;

transition-timing-function: ease-out;

text-align: left;

background-color: grey;

left: 5px;

top: 5px;

height: 26px;

color: white;

border-color: black;

font-family: sans-serif;

font-size: 20px;

text-decoration: none;

box-shadow: 2px 2px 1px black;

padding: 2px 4px;

border: solid 1px black;

}

.menuButton:hover {

position: relative;

transition-property: background-color, color;

transition-duration: 1s;

transition-timing-function: ease-out;

background-color:white;

color:black;

box-shadow: 2px 2px 1px black;

}

这个 CSS 设置了菜单的外观，在[:hover](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:hover)时，菜单里的元素的文字、颜色及背景色都发生了变化。

[**transition 让 JavaScript 效果更平滑**](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions#transition_%E8%AE%A9_javascript_%E6%95%88%E6%9E%9C%E6%9B%B4%E5%B9%B3%E6%BB%91)

transition 是非常好的工具，可以让 JavaScript 效果平滑而不用修改 JavaScript。看下面例子。

<p>随便点击某处来移动球</p>

<div id="foo"></div>

使用 JavaScript 将球移动到一个位置：

var f = document.getElementById('foo');

document.addEventListener('click', function(ev){

f.style.left = (ev.clientX-25)+'px';

f.style.top = (ev.clientY-25)+'px';

},false);

使用 CSS 来平滑移动，只用简单地添加一个过渡 :

p {

padding-left: 60px;

}

#foo {

border-radius: 50px;

width: 50px;

height: 50px;

background: #c00;

position: absolute;

top: 0;

left: 0;

transition: all 1s;

}

演示： <http://jsfiddle.net/RwtHn/5/>

### 五、transform 转换

transform属性允许你旋转，缩放，倾斜或平移给定元素

transform-origin：转换元素的位置（围绕那个点进行转换），默认值为(x,y,z):(50%,50%,0)，也叫做变换基点

使用方式：

* transform: translate(120px, 50%)：位移
* transform: scale(2, 0.5)：缩放
* transform: rotate(0.5turn)：旋转
* transform: skew(30deg, 20deg)：倾斜

#### 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。

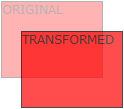
OperaSafariChromeFirefoxInternet Explorer

#### 实例

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

[**尝试一下 »**](https://www.runoob.com/try/try.php?filename=trycss3_transform1)

#### 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。

OperaSafariChromeFirefoxInternet Explorer

#### 实例

div {

transform: translate(50px,100px);

-ms-transform: translate(50px,100px); /\* IE 9 \*/ 可以不要

-webkit-transform: translate(50px,100px); /\* Safari and Chrome \*/ 可以不要

}

[**尝试一下 »**](https://www.runoob.com/try/try.php?filename=trycss3_transform_translate)

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。

#### x轴方向平移：.in{transform: translateX(300px);}

#### y轴方向平移：.in{transform: translateY(100px);}

#### 可以一起写 .in{transform: translate(300px, 100px);}

#### 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。

OperaSafariChromeFirefoxInternet Explorer

#### 实例

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

[**尝试一下 »**](https://www.runoob.com/try/try.php?filename=trycss3_transform_rotate)

rotate值（30deg）元素顺时针旋转30度。

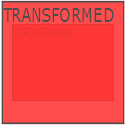
绕x轴旋转：div { transform: rotateX(30deg);

绕y轴旋转：div { transform: rotateY(30deg);

绕z轴旋转：div { transform: rotateZ(30deg);

默认旋转（绕z轴旋转就是默认值）div { transform: rotate(30deg);

#### 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：

OperaSafariChromeFirefoxInternet Explorer

#### 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 »**](https://www.runoob.com/try/try.php?filename=trycss3_transform_scale)

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。

x轴缩放 : transform: scaleX(2);

y轴缩放 : transform: scaleY(2);

x，y轴同时缩放:transform: scale(2,3);

#### skew() 方法-拉伸，也就是斜切

#### 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。

OperaSafariChromeFirefoxInternet Explorer

#### 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 »**](https://www.runoob.com/try/try.php?filename=trycss3_transform_skew)

**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。

OperaSafariChromeFirefoxInternet Explorer

#### 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 »**](https://www.runoob.com/try/try.php?filename=trycss3_transform_matrix1)

#### 新转换属性

以下列出了所有的转换属性:

|  |  |  |
| --- | --- | --- |
| **Property** | **描述** | **CSS** |
| [transform](https://www.runoob.com/cssref/css3-pr-transform.html) | 适用于2D或3D转换的元素 | 3 |
| [transform-origin](https://www.runoob.com/cssref/css3-pr-transform-origin.html) | 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。 |

#### 一般来说，转换会搭配:hover和过渡（transition）来使用

#### 转换不会影响其他元素

#### 可以在一个转换里面做两个以上的操作，他们之间用空格隔开

##### 如：

##### .out:hover .in{

##### transform: translateX(300px) translateY(300px);

##### }

#### transform-origin 转换基点

#### 格式：ransform-origin: x轴 y轴；

##### 取值：

###### 1.英文描述：top left right bottom center

###### 2.百分比：如50% 50%--》在中间

###### 3.具体的值px为单位如：100px 50px

#### transform的复合写法

例如：

##### .box{ transform : rotate(30deg) translateX(50px); }

#### 可以写多个变换

transform: translateX(300px) rotate(30deg) scale(2);

##### 注意：上面的变换中比较有用的是缩放和旋转

### 六、animation 动画

动画这个平常用的也很多，主要是做一个预设的动画。和一些页面交互的动画效果，结果和过渡应该一样，让页面不会那么生硬

animation也有很多的属性

* animation-name：动画名称
* animation-duration：动画持续时间
* animation-timing-function：动画时间函数
* animation-delay：动画延迟时间
* animation-iteration-count：动画执行次数，可以设置为一个整数，也可以设置为infinite，意思是无限循环
* animation-direction：动画执行方向
* animation-paly-state：动画播放状态
* animation-fill-mode：动画填充模式

#### animation实例1

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>css3 animation 动画</title>      <style>            p{              animation-duration: 3s;              animation-name: slidein;              animation-iteration-count: infinite;              animation-direction: alternate;          }            @keyframes slidein{              from{                   margin-left: 100%;                   width: 300%;              }              to{                   margin-left: 0%;                   width: 100%;              }          }      </style>  </head>  <body>      <div>          <p>              The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice.          </p>      </div>  </body>  </html> |
| 效果：标签p的文字会先从右边移动到左边，然后再从左边移动到右边，会一直进行下去因为设置了infinite |

#### 添加动画事件监听器

我们使用 JavaScript 代码监听所有三种可能的动画事件，setup() 方法设置事件监听器，当文档第一次加载完成时执行该方法。

var e = document.getElementById("watchme");

e.addEventListener("animationstart", listener, false);

e.addEventListener("animationend", listener, false);

e.addEventListener("animationiteration", listener, false);

e.className = "slidein";

Copy to Clipboard

以上是非常标准的代码写法，setup() 最后设置动画元素的 class 为 slidein，启动动画。为什么这样做？因为animationstart事件在动画一开始时就被触发，在我们的示例中，该事件在我们的代码执行前就被触发，所以我们自己通过设置元素的的class来启动动画。

#### 接收事件

事件传递给listener()函数，代码如下所示

function listener(e) {

var l = document.createElement("li");

switch(e.type) {

case "animationstart":

l.innerHTML = "Started: elapsed time is " + e.elapsedTime;

break;

case "animationend":

l.innerHTML = "Ended: elapsed time is " + e.elapsedTime;

break;

case "animationiteration":

l.innerHTML = "New loop started at time " + e.elapsedTime;

break;

}

document.getElementById("output").appendChild(l);

}

Copy to Clipboard

这段代码同样非常简单，简单地通过[event.type](https://developer.mozilla.org/zh-CN/docs/Web/API/Event/type)来判断发生的是何种事件，然后添加对应的注解到[<ul>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/ul)中。

输出结果如下所示：

* Started: elapsed time is 0
* New loop started at time 3.01200008392334
* New loop started at time 6.00600004196167
* Ended: elapsed time is 9.234000205993652

注意以上时间非常接近预期时间，但不是完全相等。也要注意在最后一个周期完成后，不会触发animationiteration事件，而触发animationend事件。

#### HTML 代码

下面是示例中的用到的 HTML 代码：

<body>

<h1 id="watchme">Watch me move</h1>

<p>This example shows how to use CSS animations to make <code>h1</code> elements

move across the page.</p>

<p>In addition, we output some text each time an animation event fires, so you can see them in action.</p>

<ul id="output">

</ul>

</body>

### 七、渐变

颜色渐变是指在两个颜色之间平稳的过渡，css3渐变包括

* linear-gradient：线性渐变

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

* radial-gradient：径向渐变

linear-gradient(0deg, red, green);

#### 使用 CSS 渐变

**CSS 渐变** [<image>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/image) 类型的一种特殊类型 [<gradient>](https://developer.mozilla.org/zh-CN/docs/Web/CSS/gradient) 表示，由两种或多种颜色之间的渐进过渡组成。您可以选择三种类型的渐变：线性 (由 [linear-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/linear-gradient) 函数创建)，径向 (由 [radial-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/radial-gradient) 函数创建) 和圆锥 (由 [conic-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/conic-gradient) 函数创建)。您还可以使用 [repeating-linear-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient) 和 [repeating-radial-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-radial-gradient) 函数创建重复渐变。

渐变可以在任何使用 <image> 的地方使用，例如在背景中。由于渐变是动态生成的，因此它们可以消除对传统用于实现类似效果的栅格图像文件的需求。此外，由于渐变是由浏览器生成的，因此在放大时它们看起来比栅格图像更好，并且可以动态调整大小。

我们将从线性渐变开始介绍，然后以线性渐变为例介绍所有渐变类型支持的功能，然后继续介绍径向渐变，圆锥渐变和重复渐变。

##### [使用线性渐变](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E4%BD%BF%E7%94%A8%E7%BA%BF%E6%80%A7%E6%B8%90%E5%8F%98)

线性渐变创建了一条沿直线前进的颜色带。

###### [基础线性渐变](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E5%9F%BA%E7%A1%80%E7%BA%BF%E6%80%A7%E6%B8%90%E5%8F%98)

要创建最基本的渐变类型，您只需指定两种颜色即可。这些被称为色标。至少指定两个色标，也可以指定任意数量。

.simple-linear {

background: linear-gradient(blue, pink);

}

|  |
| --- |
|  |

###### [改变渐变方向](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E6%94%B9%E5%8F%98%E6%B8%90%E5%8F%98%E6%96%B9%E5%90%91)

默认情况下，线性渐变的方向是从上到下，你可以指定一个值来改变渐变的方向。

.horizontal-gradient {

background: linear-gradient(to right, blue, pink);

}

|  |
| --- |
|  |

###### [对角线渐变](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E5%AF%B9%E8%A7%92%E7%BA%BF%E6%B8%90%E5%8F%98)

你甚至可以设置渐变方向为从一个对角到另一个对角。

.diagonal-gradient {

background: linear-gradient(to bottom right, blue, pink);

}

|  |
| --- |
|  |

###### [设置渐变角度](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E8%AE%BE%E7%BD%AE%E6%B8%90%E5%8F%98%E8%A7%92%E5%BA%A6)

如果你想要更精确地控制渐变的方向，你可以给渐变设置一个具体的角度。

.angled-gradient {

background: linear-gradient(70deg, blue, pink);

}

|  |
| --- |
|  |

在使用角度的时候，0deg 代表渐变方向为从下到上，90deg 代表渐变方向为从左到右，诸如此类正角度都属于顺时针方向。而负角度意味着逆时针方向。

##### [声明颜色和创建效果](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E5%A3%B0%E6%98%8E%E9%A2%9C%E8%89%B2%E5%92%8C%E5%88%9B%E5%BB%BA%E6%95%88%E6%9E%9C)

所有的 CSS 渐变类型都是一个位置依赖的颜色范围。CSS 渐变产生的颜色可以随位置不断变化，从而产生平滑的颜色过渡。也可以创建纯色带和两种颜色之间的硬过渡。以下内容适用于所有渐变函数：

##### [使用多种颜色](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E4%BD%BF%E7%94%A8%E5%A4%9A%E7%A7%8D%E9%A2%9C%E8%89%B2)

无需局限于使用两种颜色，你想使用多少种颜色都可以！默认情况下，所设置颜色会均匀分布在渐变路径中。

.auto-spaced-linear-gradient {

background: linear-gradient(red, yellow, blue, orange);

}

|  |
| --- |
|  |

##### [颜色终止位置](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E9%A2%9C%E8%89%B2%E7%BB%88%E6%AD%A2%E4%BD%8D%E7%BD%AE)

你不需要让你设置的颜色在默认位置终止。你可以通过给每个颜色设置 0，1% 或者 2% 或者其他的绝对数值来调整它们的位置。如果你将位置设置为百分数， 0% 表示起始点，而 100% 表示终点，但是如果需要的话你也可以设置这个范围之外的其他值来达到你想要的效果。如果有些位置你没有明确设置，那么它将会被自动计算，第一种颜色会在 0% 处停止，而最后一种颜色是 100%，至于其他颜色则是在它邻近的两种颜色的中间停止。

.multicolor-linear {

background: linear-gradient(to left, lime 28px, red 77%, cyan);

}

|  |
| --- |
|  |

##### [创建实线](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E5%88%9B%E5%BB%BA%E5%AE%9E%E7%BA%BF)

要在两种颜色之间创建一条硬线，即创建一个条纹而不是逐渐过渡，可以将相邻的颜色停止设置为相同的位置。在此示例中，两种颜色在 50% 标记处共享一个颜色停止点，即渐变的一半：

.striped {

background: linear-gradient(to bottom left, cyan 50%, palegoldenrod 50%);

}

|  |
| --- |
|  |

##### [渐变提示](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E6%B8%90%E5%8F%98%E6%8F%90%E7%A4%BA)

默认情况下，渐变会平滑地从一种颜色过渡到另一种颜色。你可以通过设置一个值来将渐变的中心点移动到指定位置。在如下示例中，我们将渐变的中心点由 50% 设为 10%。

.color-hint {

background: linear-gradient(blue, 10%, pink);

}

.simple-linear {

background: linear-gradient(blue, pink);

}

|  |
| --- |
|  |

##### [创建色带和条纹](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#%E5%88%9B%E5%BB%BA%E8%89%B2%E5%B8%A6%E5%92%8C%E6%9D%A1%E7%BA%B9)

要在渐变中包含一个实心的非过渡颜色区域，请包含颜色起止点的两个位置。颜色起止点可以有两个位置，这相当于两个连续颜色在不同位置具有相同的颜色起止点。颜色将在第一个颜色起止点时达到完全饱和，保持该饱和度到第二个颜色起止点，并通过相邻颜色起止点的第一个位置过渡到相邻颜色起止点的颜色。

.multiposition-stops {

background: linear-gradient(to left,

lime 20%, red 30%, red 45%, cyan 55%, cyan 70%, yellow 80% );

background: linear-gradient(to left,

lime 20%, red 30% 45%, cyan 55% 70%, yellow 80% );

}

.multiposition-stop2 {

background: linear-gradient(to left,

lime 25%, red 25%, red 50%, cyan 50%, cyan 75%, yellow 75% );

background: linear-gradient(to left,

lime 25%, red 25% 50%, cyan 50% 75%, yellow 75% );

}

|  |
| --- |
|  |

In the first example above, the lime goes from the 0% mark, which is implied, to the 20% mark, transitions from lime to red over the next 10% of the width of the gradient, reach solid red at the 30% mark, and staying solid red up until 45% through the gradient, where it fades to cyan, being fully cyan for 15% of the gradient, and so on.

In the second example, the second color stop for each color is at the same location as the first color stop for the adjacent color, creating a striped effect.

In both examples, the gradient is written twice: the first is the CSS Images Level 3 method of repeating the color for each stop and the second example is the CSS Images Level 4 multiple color stop method of including two color-stop-lengths in a linear-color-stop declaration.

##### [Controlling the progression of a gradient](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#controlling_the_progression_of_a_gradient)

By default, a gradient evenly progresses between the colors of two adjacent color stops, with the midpoint between those two color stops being the midpoint color value. You can control the interpolation, or progression, between two color stops by including a color hint location. In this example, the color reaches the midpoint between lime and cyan 20% of the way through the gradient rather than 50% of the way through. The second example does not contain the hint to hilight the difference the color hint can make:

.colorhint-gradient {

background: linear-gradient(to top, black, 20%, cyan);

}

.regular-progression {

background: linear-gradient(to top, black, cyan);

}

|  |
| --- |
|  |

##### [Overlaying gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#overlaying_gradients)

Gradients support transparency, so you can stack multiple backgrounds to achieve some pretty fancy effects. The backgrounds are stacked from top to bottom, with the first specified being on top.

.layered-image {

background: linear-gradient(to right, transparent, mistyrose),

url("critters.png");

}

|  |
| --- |
|  |

##### [Stacked gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#stacked_gradients)

You can even stack gradients with other gradients. As long as the top gradients aren't entirely opaque, the gradients below will still be visible.

.stacked-linear {

background:

linear-gradient(217deg, rgba(255,0,0,.8), rgba(255,0,0,0) 70.71%),

linear-gradient(127deg, rgba(0,255,0,.8), rgba(0,255,0,0) 70.71%),

linear-gradient(336deg, rgba(0,0,255,.8), rgba(0,0,255,0) 70.71%);

}

|  |
| --- |
|  |

#### repeating-linear-gradient()

The **repeating-linear-gradient()** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) [function](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Functions) creates an image consisting of repeating linear gradients. It is similar to [linear-gradient()](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/linear-gradient) and takes the same arguments, but it repeats the color stops infinitely in all directions so as to cover its entire container. The function's result is an object of the [<gradient>](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient) data type, which is a special kind of [<image>](https://developer.mozilla.org/en-US/docs/Web/CSS/image).

The length of the gradient that repeats is the distance between the first and last color stop. If the first color does not have a color-stop-length, the color-stop-length defaults to 0. With each repetition, the positions of the color stops are shifted by a multiple of the length of the basic linear gradient. Thus, the position of each ending color stop coincides with a starting color stop; if the color values are different, this will result in a sharp visual transition. This can be altered with repeating the first color again as the last color.

As with any gradient, a repeating linear gradient has [no intrinsic dimensions](https://developer.mozilla.org/en-US/docs/Web/CSS/image#description); i.e., it has no natural or preferred size, nor a preferred ratio. Its concrete size will match the size of the element it applies to.

Because <gradient>s belong to the <image> data type, they can only be used where <image>s can be used. For this reason, repeating-linear-gradient() won't work on [background-color](https://developer.mozilla.org/en-US/docs/Web/CSS/background-color) and other properties that use the [<color>](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value) data type.

##### [Syntax](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient#syntax)

/\* A repeating gradient tilted 45 degrees,

starting blue and finishing red, repeating 3 times \*/

repeating-linear-gradient(45deg, blue, red 33.3%);

/\* A repeating gradient going from the bottom right to the top left,

starting blue and finishing red, repeating every 20px \*/

repeating-linear-gradient(to left top, blue, red 20px);

/\* A gradient going from the bottom to top,

starting blue, turning green after 40%,

and finishing red. This gradient doesn't repeat because

the last color stop defaults to 100% \*/

repeating-linear-gradient(0deg, blue, green 40%, red);

/\* A gradient repeating five times, going from the left to right,

starting red, turning green, and back to red \*/

repeating-linear-gradient(to right, red 0%, green 10%, red 20%);

Copy to Clipboard

##### [Values](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient#values)

<side-or-corner>

The position of the gradient line's starting point. If specified, it consists of the word to and up to two keywords: one indicates the horizontal side (left or right), and the other the vertical side (top or bottom). The order of the side keywords does not matter. If unspecified, it defaults to to bottom.

The values to top, to bottom, to left, and to right are equivalent to the angles 0deg, 180deg, 270deg, and 90deg respectively. The other values are translated into an angle.

[<angle>](https://developer.mozilla.org/en-US/docs/Web/CSS/angle)

The gradient line's angle of direction. A value of 0deg is equivalent to to top; increasing values rotate clockwise from there.

<linear-color-stop>

A color-stop's [<color>](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value) value, followed by one or two optional stop positions, (each being either a [<percentage>](https://developer.mozilla.org/en-US/docs/Web/CSS/percentage) or a [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length) along the gradient's axis). A percentage of 0%, or a length of 0, represents the start of the gradient; the value 100% is 100% of the image size, meaning the gradient will not repeat.

<color-hint>

Th color-hint is an interpolation hint defining how the gradient progresses between adjacent color stops. The length defines at which point between two color stops the gradient color should reach the midpoint of the color transition. If omitted, the midpoint of the color transition is the midpoint between two color stops.

**Note:** Rendering of [color stops in CSS gradients](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient#gradient_with_multiple_color_stops) follows the same rules as color stops in [SVG gradients](https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Gradients).

#### [Examples](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient#examples)

##### [Zebra stripes](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient#zebra_stripes)

body {

background-image: repeating-linear-gradient(

-45deg,

transparent,

transparent 20px,

black 20px,

black 40px

);

/\* with multiple color stop lengths \*/

background-image: repeating-linear-gradient(

-45deg,

transparent 0 20px,

black 20px 40px

);

}

|  |
| --- |
|  |

##### [Ten repeating horizontal bars](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient#ten_repeating_horizontal_bars)

body {

background-image: repeating-linear-gradient(

to bottom,

rgb(26, 198, 204),

rgb(26, 198, 204) 7%,

rgb(100, 100, 100) 10%

);

}

|  |
| --- |
|  |

Because the last color stop is 10% and the gradient is vertical, each gradient in the repeated gradient is 10% of the height, fitting 10 horizontal bars.

**Note:** Please see [Using CSS gradients](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Images/Using_CSS_gradients) for more examples.

### 径向渐变

### 语法

background-image: radial-gradient(*shape* *size* at *position*, *start-color*, ..., *last-color*);

background-image: radial-gradient(形状（默认是椭圆） 范围 at x轴 y轴, *start-color*, ..., *last-color*);

形状的值有两个：circle和ellipse

范围有四个值:

farthest-corner

closest-corner

farthest-side

closest-side

默认地，*shape* 为椭圆形，*size* 为最远角，*position* 为中心。

### 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：

### 实例

#grad {

background-image: radial-gradient(red, yellow, green);

}

|  |
| --- |
| 效果 |

### 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：

### 实例

#grad {

background-image: radial-gradient(red 5%, yellow 15%, green 60%);

}

|  |
| --- |
| 效果： |

## 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：

### 实例

#grad {

background-image: radial-gradient(circle, red, yellow, green);

}

|  |
| --- |
| 效果： |

## 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：

#grad1 {

background-image: radial-gradient(closest-side at 60% 55%, red, yellow, black);

}

#grad2 {

background-image: radial-gradient(farthest-side at 60% 55%, red, yellow, black);

}

## 重复径向渐变

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

### 实例

重复的径向渐变：

#grad {

background-image: repeating-radial-gradient(red, yellow 10%, green 15%);

}

|  |
| --- |
| 效果： |

#### [Using radial gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#using_radial_gradients)

Radial gradients are similar to linear gradients, except that they radiate out from a central point. You can dictate where that central point is. You can also make them circular or elliptical.

##### [A basic radial gradient](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#a_basic_radial_gradient)

As with linear gradients, all you need to create a radial gradient are two colors. By default, the center of the gradient is at the 50% 50% mark, and the gradient is elliptical matching the aspect ratio of it's box:

.simple-radial {

background: radial-gradient(red, blue);

}

|  |
| --- |
|  |

##### [Positioning radial color stops](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#positioning_radial_color_stops)

Again like linear gradients, you can position each radial color stop with a percentage or absolute length.

.radial-gradient {

background: radial-gradient(red 10px, yellow 30%, #1e90ff 50%);

}

|  |
| --- |
|  |

##### 可以设置径向渐变的圆心的位置，使用at x轴方向 y轴方向来设置单位可以是像素或者百分比

##### [Positioning the center of the gradient](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#positioning_the_center_of_the_gradient)

You can position the center of the gradient with keyterms, percentage, or absolute lengths, length and percentage values repeating if only one is present, otherwise in the order of position from the left and position from the top.

.radial-gradient {

background: radial-gradient(at 0% 30%, red 10px, yellow 30%, #1e90ff 50%);

}

|  |
| --- |
|  |

##### [Sizing radial gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#sizing_radial_gradients)

Unlike linear gradients, you can specify the size of radial gradients. Possible values include closest-corner, closest-side, farthest-corner, and farthest-side, with farthest-corner being the default.

##### Example: closest-side for ellipses

This example uses the closest-side size value, which means the size is set by the distance from the starting point (the center) to the closest side of the enclosing box.

.radial-ellipse-side {

background: radial-gradient(ellipse closest-side,

red, yellow 10%, #1e90ff 50%, beige);

}

|  |
| --- |
|  |

##### Example: farthest-corner for ellipses

This example is similar to the previous one, except that its size is specified as farthest-corner, which sets the size of the gradient by the distance from the starting point to the farthest corner of the enclosing box from the starting point.

.radial-ellipse-far {

background: radial-gradient(ellipse farthest-corner at 90% 90%,

red, yellow 10%, #1e90ff 50%, beige);

}

|  |
| --- |
|  |

##### Example: closest-side for circles

This example uses closest-side, which makes the circle's size to be the distance between the starting point (the center) and the closest side. The circle's radius is the distance between the center of the gradient and the closest edge, which due to the positioning of the 25% from the top and 25% from the bottom, is closest to the bottom, since the height in this case is narrower than the width.

.radial-circle-close {

background: radial-gradient(circle closest-side at 25% 75%,

red, yellow 10%, #1e90ff 50%, beige);

}

|  |
| --- |
|  |

#### [Stacked radial gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#stacked_radial_gradients)

Just like linear gradients, you can also stack radial gradients. The first specified is on top, the last on the bottom.

.stacked-radial {

background:

radial-gradient(circle at 50% 0,

rgba(255,0,0,.5),

rgba(255,0,0,0) 70.71%),

radial-gradient(circle at 6.7% 75%,

rgba(0,0,255,.5),

rgba(0,0,255,0) 70.71%),

radial-gradient(circle at 93.3% 75%,

rgba(0,255,0,.5),

rgba(0,255,0,0) 70.71%) beige;

border-radius: 50%;

}

|  |
| --- |
|  |

#### [Using repeating gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#using_repeating_gradients)

The [linear-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/linear-gradient) and [radial-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/radial-gradient) properties don't support automatically repeated color stops. However, the [repeating-linear-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient) and [repeating-radial-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-radial-gradient) properties are available to offer this functionality.

The size of the gradient line that repeats is the length between the first color stop value and the last color stop length value. If the last color stop has just a color and no color stop length, the value defaults to 0, meaning the linear gradient will not repeat and the radial gradient will only repeat if the radius of the gradient is smaller than the length between the center of the gradient and the farthest corner.

#### [Repeating linear gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#repeating_linear_gradients)

This example uses [repeating-linear-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient) to create a gradient that progresses repeatedly in a straight line. The colors get cycled over again as the gradient repeats. In this case the gradient line is 10px long.

.repeating-linear {

background: repeating-linear-gradient(-45deg, red, red 5px, blue 5px, blue 10px);

}

|  |
| --- |
|  |

#### [Multiple repeating linear gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#multiple_repeating_linear_gradients)

Similar to regular linear and radial gradients, you can include multiple gradients, one on top of the other. This only makes sense if the gradients are partially transparent allowing subsequent gradients to show through the transparent areas, or if you include different [background-sizes](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-size), optionally with different [background-position](https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-position) property values, for each gradient image. We are using transparency.

In this case the gradient lines are 300px, 230px, and 300px long.

.multi-repeating-linear {

background:

repeating-linear-gradient(190deg, rgba(255, 0, 0, 0.5) 40px,

rgba(255, 153, 0, 0.5) 80px, rgba(255, 255, 0, 0.5) 120px,

rgba(0, 255, 0, 0.5) 160px, rgba(0, 0, 255, 0.5) 200px,

rgba(75, 0, 130, 0.5) 240px, rgba(238, 130, 238, 0.5) 280px,

rgba(255, 0, 0, 0.5) 300px),

repeating-linear-gradient(-190deg, rgba(255, 0, 0, 0.5) 30px,

rgba(255, 153, 0, 0.5) 60px, rgba(255, 255, 0, 0.5) 90px,

rgba(0, 255, 0, 0.5) 120px, rgba(0, 0, 255, 0.5) 150px,

rgba(75, 0, 130, 0.5) 180px, rgba(238, 130, 238, 0.5) 210px,

rgba(255, 0, 0, 0.5) 230px),

repeating-linear-gradient(23deg, red 50px, orange 100px,

yellow 150px, green 200px, blue 250px,

indigo 300px, violet 350px, red 370px);

}

|  |
| --- |
|  |

#### [Plaid gradient](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#plaid_gradient)

To create plaid we include several overlapping gradients with transparency. In the first background declaration we listed every color stop separately. The second background property declaration using the multiple position color stop syntax:

.plaid-gradient {

background:

repeating-linear-gradient(90deg, transparent, transparent 50px,

rgba(255, 127, 0, 0.25) 50px, rgba(255, 127, 0, 0.25) 56px,

transparent 56px, transparent 63px,

rgba(255, 127, 0, 0.25) 63px, rgba(255, 127, 0, 0.25) 69px,

transparent 69px, transparent 116px,

rgba(255, 206, 0, 0.25) 116px, rgba(255, 206, 0, 0.25) 166px),

repeating-linear-gradient(0deg, transparent, transparent 50px,

rgba(255, 127, 0, 0.25) 50px, rgba(255, 127, 0, 0.25) 56px,

transparent 56px, transparent 63px,

rgba(255, 127, 0, 0.25) 63px, rgba(255, 127, 0, 0.25) 69px,

transparent 69px, transparent 116px,

rgba(255, 206, 0, 0.25) 116px, rgba(255, 206, 0, 0.25) 166px),

repeating-linear-gradient(-45deg, transparent, transparent 5px,

rgba(143, 77, 63, 0.25) 5px, rgba(143, 77, 63, 0.25) 10px),

repeating-linear-gradient(45deg, transparent, transparent 5px,

rgba(143, 77, 63, 0.25) 5px, rgba(143, 77, 63, 0.25) 10px);

background:

repeating-linear-gradient(90deg, transparent 0 50px,

rgba(255, 127, 0, 0.25) 50px 56px,

transparent 56px 63px,

rgba(255, 127, 0, 0.25) 63px 69px,

transparent 69px 116px,

rgba(255, 206, 0, 0.25) 116px 166px),

repeating-linear-gradient(0deg, transparent 0 50px,

rgba(255, 127, 0, 0.25) 50px 56px,

transparent 56px 63px,

rgba(255, 127, 0, 0.25) 63px 69px,

transparent 69px 116px,

rgba(255, 206, 0, 0.25) 116px 166px),

repeating-linear-gradient(-45deg, transparent 0 5px,

rgba(143, 77, 63, 0.25) 5px 10px),

repeating-linear-gradient(45deg, transparent 0 5px,

rgba(143, 77, 63, 0.25) 5px 10px);

}

|  |
| --- |
|  |

#### [Repeating radial gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#repeating_radial_gradients)

This example uses [repeating-radial-gradient (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-radial-gradient) to create a gradient that radiates repeatedly from a central point. The colors get cycled over and over as the gradient repeats.

.repeating-radial {

background: repeating-radial-gradient(black, black 5px, white 5px, white 10px);

}

|  |
| --- |
|  |

#### [Multiple repeating radial gradients](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#multiple_repeating_radial_gradients)

.multi-target {

background:

repeating-radial-gradient(ellipse at 80% 50%,rgba(0,0,0,0.5),

rgba(0,0,0,0.5) 15px, rgba(255,255,255,0.5) 15px,

rgba(255,255,255,0.5) 30px) top left no-repeat,

repeating-radial-gradient(ellipse at 20% 50%,rgba(0,0,0,0.5),

rgba(0,0,0,0.5) 10px, rgba(255,255,255,0.5) 10px,

rgba(255,255,255,0.5) 20px) top left no-repeat yellow;

background-size: 200px 200px, 150px 150px;

}

|  |
| --- |
|  |

#### [Plaid gradient](https://developer.mozilla.org/zh-CN/docs/Web/CSS/CSS_Images/Using_CSS_gradients#plaid_gradient_2)

To create plaid we include several overlapping gradients with transparency. In the first background declaration we listed every color stop separately. The second background property declaration using the multiple position color stop syntax:

<div class="plaid-gradient"></div>

div {

width: 200px;

height: 200px;

}

.plaid-gradient {

background:

repeating-linear-gradient(90deg, transparent, transparent 50px,

rgba(255, 127, 0, 0.25) 50px, rgba(255, 127, 0, 0.25) 56px,

transparent 56px, transparent 63px,

rgba(255, 127, 0, 0.25) 63px, rgba(255, 127, 0, 0.25) 69px,

transparent 69px, transparent 116px,

rgba(255, 206, 0, 0.25) 116px, rgba(255, 206, 0, 0.25) 166px),

repeating-linear-gradient(0deg, transparent, transparent 50px,

rgba(255, 127, 0, 0.25) 50px, rgba(255, 127, 0, 0.25) 56px,

transparent 56px, transparent 63px,

rgba(255, 127, 0, 0.25) 63px, rgba(255, 127, 0, 0.25) 69px,

transparent 69px, transparent 116px,

rgba(255, 206, 0, 0.25) 116px, rgba(255, 206, 0, 0.25) 166px),

repeating-linear-gradient(-45deg, transparent, transparent 5px,

rgba(143, 77, 63, 0.25) 5px, rgba(143, 77, 63, 0.25) 10px),

repeating-linear-gradient(45deg, transparent, transparent 5px,

rgba(143, 77, 63, 0.25) 5px, rgba(143, 77, 63, 0.25) 10px);

background:

repeating-linear-gradient(90deg, transparent 0 50px,

rgba(255, 127, 0, 0.25) 50px 56px,

transparent 56px 63px,

rgba(255, 127, 0, 0.25) 63px 69px,

transparent 69px 116px,

rgba(255, 206, 0, 0.25) 116px 166px),

repeating-linear-gradient(0deg, transparent 0 50px,

rgba(255, 127, 0, 0.25) 50px 56px,

transparent 56px 63px,

rgba(255, 127, 0, 0.25) 63px 69px,

transparent 69px 116px,

rgba(255, 206, 0, 0.25) 116px 166px),

repeating-linear-gradient(-45deg, transparent 0 5px,

rgba(143, 77, 63, 0.25) 5px 10px),

repeating-linear-gradient(45deg, transparent 0 5px,

rgba(143, 77, 63, 0.25) 5px 10px);

|  |
| --- |
|  |

### 还可以利用渐变来实现一个滤镜效果

#### 加滤镜前的图片

|  |
| --- |
|  |

#### 滤镜代码

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>图片滤镜效果</title>      <style>          .box{              width: 1200px;              height: 350px;              margin: 0 auto;              /\* 利用渐变实现滤镜效果 \*/              /\* background-image: url(./banner1.jpg); \*/              background-image: linear-gradient(rgba(255,255,0,0.3),rgba(255,97,0,0.3)),url(./banner1.jpg);          }      </style>  </head>  <body>      <div class="box"></div>  </body>  </html> |
| 加了滤镜效果后的图片  |  | | --- | |  | |

#### 注意：一定要注意顺序，先写渐变，再放图片否则图片把渐变盖住了，就没有效果

窗体底端

## 10个demo示例学会CSS3 radial-gradient径向渐变

### 一、语法细节记不住怎么办？

实际开发的时候，当要使用radial-gradient径向渐变的时候，脑中会有大概的语法，但是细节却记不住，于是想快捷找个案例看看具体怎么用，然后直接套一下。通常一番搜索，会发现虽然是个简单需求，但是正好满足这个需求的页面却不好找，乱糟糟的。

下次遇到这种场景，直接来本站搜“径向渐变”，或者直接搜索“渐变”，就有专门展示radial-gradient径向渐变基本语法使用案例的文章。文章共展示了10例常见使用案例，相信一定可以覆盖你的使用场景的。

首先，假设我们使用同一段HTML作为示意：

<div class="radial-gradient"></div>

然后…

### 二、示例1：最基础最简单使用

CSS如下：

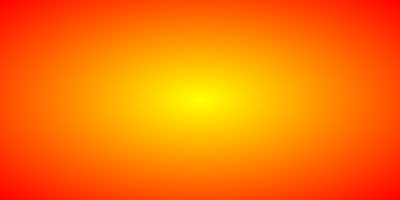
.radial-gradient {

width: 400px; height: 200px;

background: radial-gradient(yellow, red);

}

最终效果如下图：

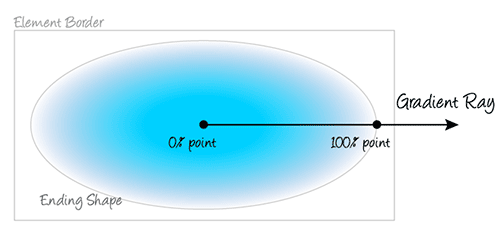


若要亲自确认效果，您可以狠狠地点击这里：[radial-gradient径向渐变最基本语法效果demo](https://www.zhangxinxu.com/study/201711/radial-gradient-simplest.html)

可见，对于径向渐变，在不指定渐变类型以及位置的情况下，其渐变距离和位置是由容器的尺寸决定的。

例如本例之中，容器的宽高比是2:1，最终渐变呈现出来的形状也是一个2:1的椭圆形，并且渐变颜色自动终止与容器的边缘。

原理示意如下：



如果我们想指定渐变的形状是一个圆形，形状不受外部容器尺寸影响，可以使用circle关键字。

### 三、示例2：简单的圆形渐变

CSS代码如下：

.radial-gradient {

width: 400px; height: 200px;

background: radial-gradient(circle, yellow, red);

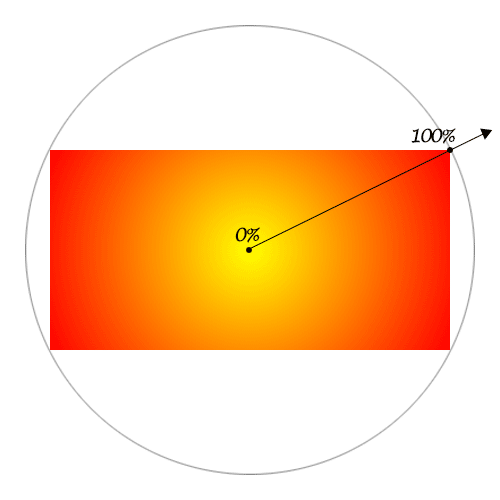
}

最终效果如下图：



若要亲自确认效果，您可以狠狠地点击这里：[radial-gradient径向渐变简单圆形渐变效果demo](https://www.zhangxinxu.com/study/201711/radial-gradient-circle-at-center.html)

其渐变范围（渐变结束线）示意如下图，会发现既不是按照宽度来的，也不是按照高度来的，是按照最远边角距离作为渐变结束线的：



要证明上面结论比较简单，我们可以计算下对角线一半长度，为：Math.sqrt(200\*200 + 400\*400) ≈ 223.6，于是，我们设置：

.radial-gradient {

width: 400px; height: 200px;

background: radial-gradient(223.6px circle, yellow, red);

}

会发现，最终效果和上面的效果截图几乎就是一模一样的。而200px的半径：

.radial-gradient {

width: 400px; height: 200px;

background: radial-gradient(200px circle, yellow, red);

}

取色就会发现边缘颜色差异较大，说明默认不是按照最短边来渲染的。

### 四、示例3：指定渐变起始点位置

起始点位置可以通过at来指定，例如：

.radial-gradient {

width: 400px; height: 200px;

background: radial-gradient(circle at 50px 50px, yellow, red);

}

效果如下图所示：



若要亲自确认效果，您可以狠狠地点击这里：[径向渐变at确定渐变起始点demo](https://www.zhangxinxu.com/study/201711/radial-gradient-circle-at.html)

如果希望渐变是和容器保持一致比例的椭圆，则circle可以缺省，也就是直接radial-gradient(at 50px 50px, yellow, red)也是可以的。

50px 50px我们也可以使用百分比值代替，例如：

.radial-gradient {

width: 400px; height: 200px;

background: radial-gradient(circle at 12.5% 25%, yellow, red);

}

效果是一模一样的。

### 五、示例4：指定渐变终止点位置

radial-gradient径向渐变支持4个关键字可以指定渐变终止点位置，见下表：

|  |  |
| --- | --- |
| **关键字** | **描述** |
| closest-side | 渐变中心距离容器最近的边作为终止位置。 |
| closest-corner | 渐变中心距离容器最近的角作为终止位置。 |
| farthest-side | 渐变中心距离容器最远的边作为终止位置。 |
| farthest-corner | 渐变中心距离容器最远的角作为终止位置。 |

根据上面的示意效果可以看出默认值是farthest-corner。

现在，我们对CSS进行调整，如下：

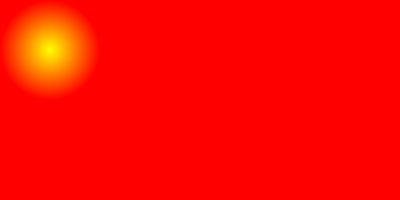
.radial-gradient {

width: 400px; height: 200px;

background: radial-gradient(closest-side circle at 50px 50px, yellow, red);

}

也就是圆形渐变终止于最短边，结果效果如下图：



若要亲自确认效果，您可以狠狠地点击这里：[径向渐变终止于最短边demo](http://www.zhangxinxu.com/study/201711/radial-gradient-circle-closest-side.html)

### 六、示例5：指定渐变颜色断点

如果指定多个颜色，但未指定具体断点的位置，则这些颜色会均匀分配0%~100%的渐变区域，例如，下面4色渐变：

.radial-gradient {

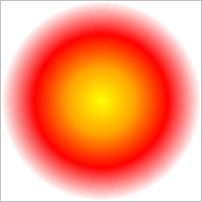
width: 200px; height: 200px;

border: 1px solid silver;

background: radial-gradient(closest-side circle, yellow, orange, red, white);

}

结果如下图所示：



但从肉眼视觉我们是看不出是不是均匀分配渐变区域，但是我们可以通过其他方式验证我们的观点，如下指定颜色断点位置的CSS：

.radial-gradient2 {

width: 200px; height: 200px;

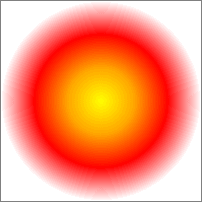
border: 1px solid gray;

background: radial-gradient(closest-side circle, yellow, orange 33.33%, red 66.666%, white);

}

由于渐变颜色默认第一个颜色位置是0%，最后一个颜色位置是100%，因此上面yellow和white的断点位置可以缺省。

然后发现效果和上面的是一致的：



我做了个更能直观体验一致性的demo页面，您可以狠狠地点击这里：[radial-gradient多个颜色均匀分配渐变区域demo](http://www.zhangxinxu.com/study/201711/radial-gradient-circle-colors.html)

按下第一个圈圈渐变，会让后面的圈圈瞬间覆盖在上面。如果两个渐变颜色有差异，此时肉眼可以感觉到明显变化；但是实际操作下来，就好像后面渐变突然消失一般，这就说明上下两个渐变实际上效果是一致的。

### 七、示例6：椭圆类型的径向渐变

对于圆形界面，我们只需要指定一个半径就可以了，但是对于椭圆类型的径向渐变，我们需要同时指定横轴和纵轴的长度，例如：

.radial-gradient {

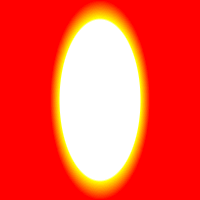
width: 200px; height: 200px;

background: radial-gradient(50px 100px ellipse, transparent 40px, yellow 41px, red);

}

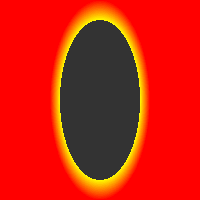
50px 100px ellipse中第一个数值50px表示横轴半径，100px表示纵轴半径，于是这段语句表示绘制一个长度100px高度200px的椭圆。

效果如下图：



若要亲自确认效果，您可以狠狠地点击这里：[radial-gradient径向渐变椭圆渐变demo](http://www.zhangxinxu.com/study/201711/radial-gradient-ellipse.html)

需要注意的是，在上面示意CSS代码中，透明到黄色分界那里有一个1px的偏差过渡，也就是transparent 40px, yellow 41px中yellow是41px，而不是设置的40px，原因在于在Chrome下，如果颜色直接0偏差过渡，会有比较严重的锯齿，类似下图这样（背景色设为#333）：



通过有1像素或者半像素的过渡缓冲可以有效避免锯齿的出现。

### 八、示例7：可累加的径向渐变背景图

我们可以把多个径向渐变累加在一起实现某些效果，例如下面CSS：

.radial-gradient {

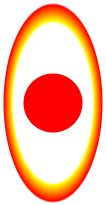
width: 200px; height: 200px;

background: radial-gradient(50px 100px ellipse, transparent 40px, yellow 41px, red 49px, transparent 50px),

radial-gradient(30px circle, red, red 29px, transparent 30px);

}

实现了“邪王真眼”效果：

若要亲自感受效果，您可以狠狠地点击这里：[radial-gradient多个径向渐变累加demo](http://www.zhangxinxu.com/study/201711/radial-gradient-multiple.html)

### 九、示例8：渐变背景的尺寸控制

配合background-size的尺寸控制和背景重复特性，我们可以实现渐变式的复杂纹理效果：

.radial-gradient {

width: 200px; height: 200px;

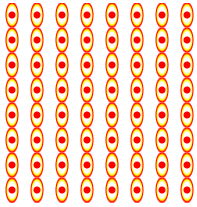
background: radial-gradient(5px 10px ellipse, transparent 4px, yellow 5px, red 6px, transparent 7px),

radial-gradient(3px circle, red, red 3px, transparent 4px);

background-size: 25px;

}

于是就可以看到一大波写轮眼：



通常我们使用径向渐变构建一些简单实用图形的时候，background-size往往是关键属性，例如，我们要实现个水波效果，可以这样：

.radial-gradient {

width: 200px; height: 100px;

background: red;

position: relative;

}

.radial-gradient:after {

content: '';

position: absolute;

height: 10px;

left:0 ; right: 0;

bottom: -10px;

background: radial-gradient(20px 15px ellipse at top, red 10px, transparent 11px);

background-size: 20px 10px;

}

然后就有下图所示的效果：



实时效果体验可以狠狠地点击这里：[径向渐变尺寸控制下的波纹效果demo](http://www.zhangxinxu.com/study/201711/radial-gradient-wave.html)

### 十、示例9：可重复的径向渐变

如果想要实现可重复的径向渐变，亦可以使用原生的repeating-radial-gradient()方法，特别适合实现从中心扩散的光波效果。

然而，相比重复线性渐变repeating-linear-gradient()方法，repeating-radial-gradient()的实际应用场景实际上很有限。因为，实际开发的时候，我们很少用到从中心扩散的光波效果。

除了一些本身就是波纹类型效果，如水波，声波或者唱片纹理：

如下CSS（取自<https://codepen.io/thebabydino/embed/HjJlL>）：

.radial-gradient {

position: relative;

width: 262px; height: 262px;

border-radius: 50%;

background: linear-gradient(30deg, transparent 40%, rgba(42, 41, 40, .85) 40%) no-repeat 100% 0, linear-gradient(60deg, rgba(42, 41, 40, .85) 60%, transparent 60%) no-repeat 0 100%, repeating-radial-gradient(#2a2928, #2a2928 4px, #ada9a0 5px, #2a2928 6px);

background-size: 50% 100%, 100% 50%, 100% 100%;

}

.radial-gradient:after {

position: absolute;

top: 50%; left: 50%;

margin: -35px;

border: solid 1px #d9a388;

width: 68px; height: 68px;

border-radius: 50%;

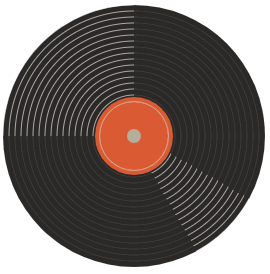
box-shadow: 0 0 0 4px #da5b33, inset 0 0 0 27px #da5b33;

background: #b5ac9a;

content: '';

}

效果如下图：



效果体验，可以狠狠地点击这里：[可重复径向渐变下的胶片效果](http://www.zhangxinxu.com/study/201711/radial-gradient-repeating.html)

### 十一、示例10：作为border-image的径向渐变

径向渐变不仅可以作为background的背景图，还可以作为border-image的边框图使用。

例如：

.radial-gradient {

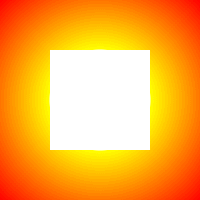
width: 100px; height: 100px;

border: 50px solid;

border-image: radial-gradient(circle, transparent 50px, yellow 51px, red) 50 stretch;

}

效果：



天然镂空效果，只可惜border-image是无法和border-radius同时生效的，否则，border-image可就要牛炸天了。

效果体验，可以狠狠地点击这里：[radial-gradient径向渐变与border-image呈现demo](http://www.zhangxinxu.com/study/201711/radial-gradient-border-image.html)

### 关于渐变，是线性渐变用的多径向渐变不常用

### 八、其他

关于css3其他的新特性还包括flex弹性布局、Grid栅格布局，这两个布局在以前就已经讲过，这里就不再展示

除此之外，还包括多列布局、媒体查询、混合模式等等......

## 参考文献

* <https://juejin.cn/post/6844903518520901639#heading-1>
* <https://www.w3school.com.cn/css/index.asp>