# CSS3 Animation 动画

|  |  |
| --- | --- |
| **属性** | **描述** |
| @keyframes | 关键帧：定义动画的变换规则 |
| animation-name | 规定 @keyframes 动画的名称。 |
| animation-duration | 规定动画完成一个周期所花费的秒或毫秒。默认是 0。 |
| animation-timing-function | 规定动画的速度曲线。默认是 "ease"。 |
| animation-delay | 规定动画何时开始。默认是 0。 |
| animation-iteration-count | 规定动画被播放的次数。默认是 1。 |
| animation-direction | 规定动画是否在下一周期逆向地播放。默认是 "normal"。 |
| animation | 所有动画属性的简写属性，**除了 animation-play-state 属性** |

## @keyframes

@keyframes **anim1** {

from{

background-color: red;

}

to{

background-color: blue;

}

}

@keyframes **anim2** {

0%{

background-color: red;

}

10%{

background-color: green;

}

100%{

background-color: blue;

}

}

@keyframes **anim3** {

from{

background-color: red;

}

10%{

background-color: green;

}

to{

background-color: blue;

}

}

当有几个帧样式一样时，可以用,来一起设置

@keyframes anim2 {

/\*使用,设置各帧的相同样式 0%{} 和 90{} 使用相同样式\*/

0%, 90%{

background: url("img/1.png") no-repeat;

background-size: 100%;

}

30%,50%{

background: url("img/2.png") no-repeat;

background-size: 100%;

}

70%,100%{

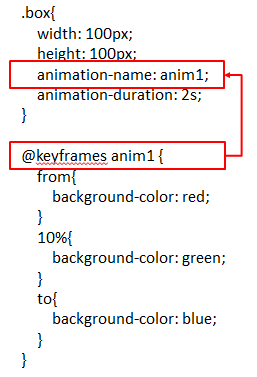
background: url("img/3.png") no-repeat;

background-size: 100%;

}

}

## animation-name



## animation-iteration-count

|  |  |
| --- | --- |
| **值** | **描述** |
| n | 定义动画播放次数的数值。 |
| infinite | 规定动画应该无限次播放。 |

## animation-direction

|  |  |
| --- | --- |
| **值** | **描述** |
| normal | 默认值。动画应该正常播放。 |
| alternate | 动画应该轮流反向播放。必须配合播放次数使用，播放  次数大于1 |

## animation-play-state

|  |  |
| --- | --- |
| **值** | **描述** |
| paused | 规定动画已暂停。 |
| running | 规定动画正在播放。 |

## animation-fill-mode

|  |  |
| --- | --- |
| **值** | **描述** |
| none | 不改变默认行为。 |
| forwards | 当动画完成后，保持最后一个属性值（在最后一个关键帧中  定义，关键帧必须是100%）。 |
| backwards | 在 animation-delay 所指定的一段时间内，在动画显示之前，  应用开始属性值（在第一个关键帧中定义，关键帧必须是0%）。 |
| both | 向前和向后填充模式都被应用。关键帧开始0%，最后100% |

## 动画属性简写

animation: name duration timing-function delay iteration-count direction;

请始终规定 animation-duration 属性，否则时长为 0，就不会播放动画了。

当有多个动画：animation: name duration timing-function delay iteration-count direction, name duration timing-function delay iteration-count direction;

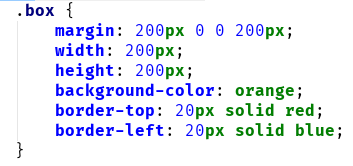
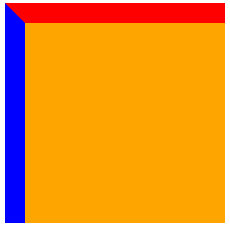
## 与opcity综合使用

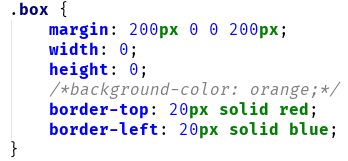
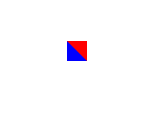
实例：淡入，淡出效果

# 边框

* CSS1 与 CSS2 的边框属性
  + border-width，设置元素四个边框的宽度
  + border-color，设置元素四个边框的颜色
  + border-style ，元素四个边框的样式。
  + 上述三个属性都遵循“**TRBL原则**”（上，右，下，左）。
  + 简写方式 border：*border-width*, *border-color,* *border-style*

## 利用CSS2边框制作三角形

## CSS3边框

* CSS3 新增的边框样式
  + border-(top | right | bottom | left)-color: 分别设置元素四个边框的颜色。由于浏览器兼容性问题，该属性极少在商业网站中使用。
  + border-image：使用图片作为元素的边框。浏览器支持较好，但商业网站应用很少，主要用在个人博客中。
  + border-radius：让元素显示圆角效果。浏览器支持较好，应用非常广泛。
  + box-shadow：让元素显示阴影效果。应用非常广泛，主要用来呈现元素立体效果。

### CSS3 边框颜色属性

* CSS3边框 语法：
  + border-top-color: [<color>]，上边框颜色
  + border-right-color: [<color>]，右边框颜色
  + border-bottom-color: [<color>]，下边框颜色
  + border-left-color: [<color>]，左边框颜色
  + 上述四个样式属性，仍然遵循TRBL原则
  + 简写方式，border-color: [ <color> | transparent ] {1, 4} | inherit，
* 特别说明：
* 1、该样式可以应用到所有的元素，甚至伪元素。例如::first-letter。
* 2、该样式具有动画效果。

**为边框设置多个颜色（仅Firefox浏览器支持）**

* + -moz-border-top-colors：[color]{1,n}
  + -moz-border-right-colors：[color]{1,n}
  + -moz-border-bottom-colors：[color]{1,n}
  + -moz-border-left-colors：[color]{1,n}
* 每个属性后面可以设置N个合法颜色值，每个颜色占1px。如果元素的边框设置了20px，而颜色只设置了10个，剩下的10px都使用最后一个颜色值。
* 注意：
  + 上述属性仅在Firefox3.0以上有效，它不是标准语法。而且目前仍在实验阶段，实际使用一定要慎重。
  + 上述样式的优先级比 border-[top | right | bottom | left ]-color 优先级高。
  + 不支持动画效果
* **示例代码：立体效果的边框**

-moz-border-bottom-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;

-moz-border-top-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;

-moz-border-left-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;

-moz-border-right-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;

需要配合border: 10px solid;一起使用。

注意：该样式仅在Firefox浏览器下可使用

### boder-image图片边框

#### border-image-source: url(图片路径)

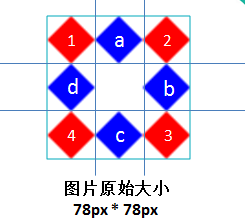
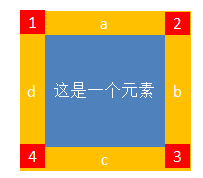
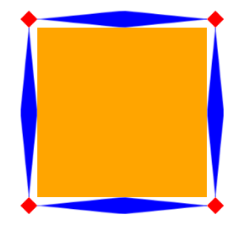
* + 设置图片路径。
* **图片边框样式生效的前提: 必须设置 border 的宽度 和 样式**
* 例如，border: 1px solid 或者 border: 2px dotted 等等

#### border-image-slice: [<percentage> | <number>]｛1,4｝ && fill

* 图片切割的方式。让元素的边框的 **四个角**，**四条边** 显示切割后的图片。
* 可选值
  + 关键字：fill （默认值） **图片填充边框的四个角**。
  + 数字{1, 4｝（不需要写单位，默认为px）
  + 百分比值 {1, 4} --- 基于图片的宽度或者高度

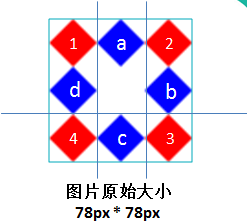
使用数值

* + 26
  + 等价于 26 26 26 26
* 切割顺序及位置
  + 距离图片上边 26px 位置切割
  + 距离图片右边 26px 位置切割
  + 距离图片下边 26px 位置切割
  + 距离图片左边 26px 位置切割
* 经过上述切割
  + 我们已经把一张图片切割成了9份
  + 按照其编号分别对应元素边框的4个角、4个边（中间的图不需要）

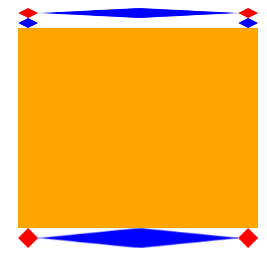
  浏览器实际效果，a,b,c,d默认拉伸了

使用数值

* + 52 26 26 26
* 切割顺序及位置
  + 距离图片上边 52px 位置切割
  + 距离图片右边 26px 位置切割
  + 距离图片下边 26px 位置切割
  + 距离图片左边 26px 位置切割



距离上边52px和距离下边26px两条线重合了，中间图片没有了，所以元素左右边框没有了



控制元素左上角边框的切割是：距离图片上边位置切割和距离图片左边位置切割

控制元素上边边框的切割是：距离图片上边位置切割、距离图片左边位置切割和距离图片右边位置切割

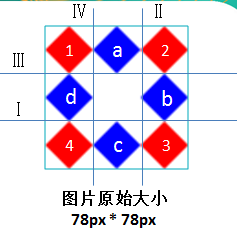
控制元素右上角边框的切割是：距离图片上边位置切割和距离图片右边位置切割

以此类推

只要切割控制区域有效，元素边框就可以设置，无效（如切割位置重叠）则没有图片样式，使用默认样式

使用数值

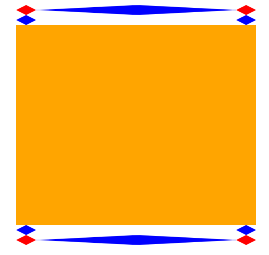
* + 52 26 52 26
* 切割顺序及位置
  + 1.距离图片上边 52px 位置切割
  + 2.距离图片右边 26px 位置切割
  + 3.距离图片下边 52px 位置切割
  + 4.距离图片左边 26px 位置切割



图片Ⅰ 、Ⅱ、Ⅲ、Ⅳ对应切割顺序

由于控制左上角的是Ⅰ 、Ⅳ，所以左上角显示的是图片的中的1、d；上边是a和中间空白；右上角是2、b；由于Ⅰ 和Ⅲ切割重复，所以是无效的，这两个切割控制的左右边框就没有图片样式，使用默认样式；控制右下角是Ⅱ、Ⅲ，所以右下角显示的是图片中的b、3；下边是空白和c；左下角是d、4。

关键是记住哪个区域是由哪几个切割线控制的，如果切割线重复切割了，区域无效，没有重复切割，区域有效

浏览器实际效果

只有一个数字：上右下左

两个数字：上下 左右

三个数字：上 左右 下

四个数字：上 右 下 左

#### border-image-width: [<length>]｛1,4｝

* + 用来表示背景图片显示的大小。
  + 与border-width设置效果相同，**但是不会导致元素的实际大小改变**。
* 样式值
  + 10px 四个图片边框都是10px
  + 10px 20px 上下边框 10px ， 左右边框 20px
  + 10px 20px 30px 上10px ,左右20px, 下30px
  + 10px 20px 30px 40px 上10px ,右20px, 下30px, 左40px

#### border-image-repeat: [stretch | round | repeat]

* + stretch: 默认值 **四个边个图片被拉伸**
  + round: 填满 **四个边的图片重复平铺 ，确保图片完整性。**
  + repeat: 重复 **四个边的图片重复平铺**

#### border-image-outset: [<length>]｛1,4｝

* 图片外边框的位置，只能为整数，位移不会改变盒子的大小。
* 样式值
  + 10px 四个图片边框都向外移动10px
  + 10px 20px 上下边框向外移动10px ， 左右边框向外移动20px
  + 10px 20px 30px 上边框向外移动10px ,左右边框向外移动20px, 下边框向外移动30px
  + 10px 20px 30px 40px 上边框向外移动10px ,右边框向外移动20px, 下边框向外移动 30px, 左边框向外移动 40px

#### 简写方式：

* border-image: <border-image-source>||<border-image-slice> [/<border-image-width> || <border-image-repeat>]

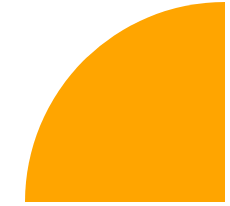
### boder-radius圆角

#### border-radius: none | <length> {1, 4} [/ <length> {1, 4} ]

* 1. 如果有反斜杠，前面设置的元素圆角水平方向半径，后面是垂直方向半径。
  2. 如果没有反斜杠，表示圆角水平、垂直方向半径相同。



* border-radius：
  + <length>{1} ， 四个角的值相同，也就是4个圆角。
* border-radius：
  + <length>{2} ， top-left等于bottom-right，并且取第一个值。top-right等于bottom-left，取第二个值。
* border-radius：
  + <length>{3} ，top-left取第一个值，top-right和bottom-left取第二个值，bottom-right取第三个值。
* border-radius：
  + <length>{4} ，四个角取不同的值，top-left取第一个，top-right取第二个，bottom-right取第三个，bottom-left取第四个。
* 如果不想给元素设置圆角，只需要 border-radius: 0或者none 即可。
* border-radius: 50%， 如果元素是正方形，显示圆形
* border-radius: 50%， 如果元素是长方形，显示椭圆
* border-radius: 一边是100%，其余都是0，如果元素是正方形，显示扇形



border-radius: 100% 0 0 0;



border-radius: 0 0 100% 0;

推荐使用%设置圆角半径，可以随着元素大小动态调整

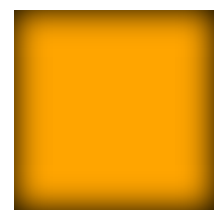
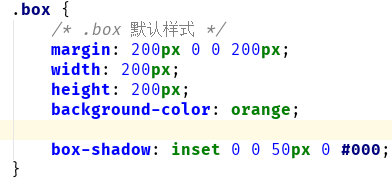
# CSS3 盒子阴影属性

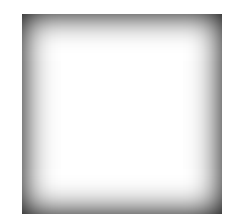
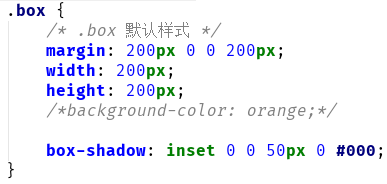
* 属性box-shadow property设置盒子的阴影
* box-shadow:
  + 参数1：投影方式 。默认外阴影，inset是内阴影。
  + 参数2： X轴偏移量 ，可以是负值
  + 参数3： Y轴偏移量 ，可以是负值
  + 参数4：阴影模糊半径
  + 参数5：阴影扩展半径，可以是负值
  + 参数6：阴影颜色
* none: 默认值，元素不设置阴影。
* inset：阴影类型，可选。如果不设置投影方式外部阴影，如果设置inset就是内 阴影
* x-offset:阴影水平偏移。正值，阴影在元素右侧，负值在元素左侧。
* y-offset:阴影垂直偏移。正式，阴影在元素底部，负值在元素顶部。
* blur-radius: 阴影的模糊半径，可选。只能为正值，如果设置为0，阴影不具备模糊效果。
* spread-radius: 阴影扩展半径。可选参数。其值可正可负。如果取正值，整个阴影都延展扩大，反之取负值，则整个阴影都缩小。
* color: 阴影的颜色。 可选参数
* 单边实影：只需要调整 x-offset 或者 y-offset 即可。
  + box-shaw: 0 -2px #666 上阴影
  + box-shaw: 2px 0 #666 右阴影
  + box-shaw: 0 2px #666 下阴影
  + box-shaw: -2px 0 #666 左阴影
* 单边阴影：调整 x-offset 或者 y-offset 设置偏移， 设置模糊半径实现模糊效果。
  + box-shaw: 0 -2px 5px #666 上模糊阴影
  + box-shaw: 2px 0 5px #666 右模糊阴影
  + box-shaw: 0 2px 5px #666 下模糊阴影
  + box-shaw: -2px 0 5px #666 左模糊阴影
* 单边阴影： 设置阴影深度。
  + box-shaw: 0 -2px 5px 6px #666 上模糊阴影
  + box-shaw: 2px 0 5px 6px #666 右模糊阴影
  + box-shaw: 0 2px 5px 6px #666 下模糊阴影
  + box-shaw: -2px 0 5px 6px #666 左模糊阴影
  + 阴影扩展半径是正值，与设置相同大小边框（border）的效果相同

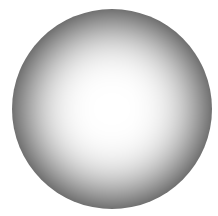
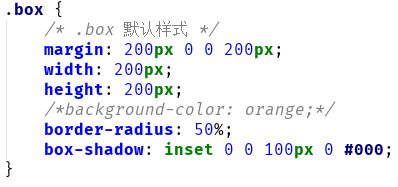
## 多阴影样式：

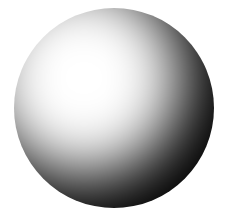
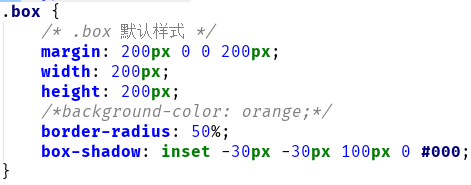
box-shadow: inset x-offset y-offset blur-radius spread-radius color，inset x-offset y-offset blur-radius spread-radius color

## 利用内阴影制作球体









## 利用阴影扩展半径制作一个阴影区域

.box {

margin: 200px 0 0 200px;

width: 300px;

height: 0px;

box-shadow:0 0 50px 1px red;

}



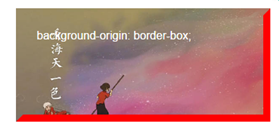
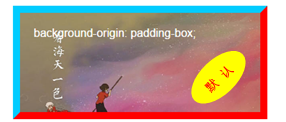
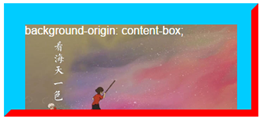
左边有一个红色阴影

# 背景

## 背景的基本属性

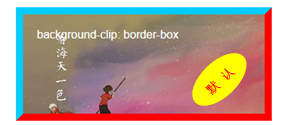
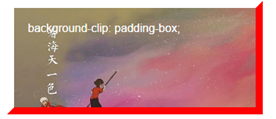
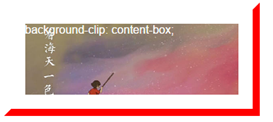
* background-position: 背景图片位置
  + top right ;
  + 50% 50%；
  + **0 0**
* background-repeat:
  + no-repeat;
  + repeat;（默认）
* background-attachment:
  + fixed;
  + scroll
* background-color:
  + #00CCFF;
  + rgb(255, 0, 0 );
* background-image:
  + url(“logo.png”)

## 背景原点属性

* background-origin: border-box;
  + 背景图的background-position起始点（默认左上角（0,0））是从元素边框外边缘开始。
  + 
* background-origin: padding-box;（默认值）
  + 背景图的background-position起始点（默认左上角（0,0））是从元素边框内边缘（padding的外边缘）开始。
  + 
* background-origin: content-box;
  + 背景图的background-position起始点（默认左上角（0,0））是从元素内容的边框外边缘（padding的内边缘）开始。
  + 

## 背景裁剪属性

如果不设置原点，图片默认是基于background-origin: padding-box;

* background-clip: border-box;（默认值）
  + 元素外边框外的背景图片都将被裁裁剪掉。
  + 
* background-clip: padding-box;
  + 元素padding区域外的背景图片都将被裁裁剪掉。
  + 
* background-clip: content-box;
  + 元素内容区域外背景图片都将被裁裁剪掉。
  + 

## 文本裁剪

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;



将背景图根据文字裁剪，有文字的才显示背景图

**注意：只有 webkit /blink内核浏览器实现的使用图片填充文本的效果**

**该样式在移动端使用较多**

## 背景尺寸

* background-size: auto; 默认值

保持背景图原始的高度与宽度。就是背景图保持原样，如果背景图大于父元素，只显示父元素大小对应的图片大小，其余隐藏，如果图片小于父元素大小，图片会平铺

* background-size: <length>; 取整数值，例如px。

设置后，会改变背景大小。如果只设置与一个值，代表背景的宽度，第二个值为auto（根据图片宽度等比计算高度）。

* background-size: <percentage>; 使用%0~100%的值。

设置后，其值是相对于背景图所在**元素的宽度**计算。如果元素的宽度是1000px，当background-size：50% 40%，此时背景图的宽度应该是500（1000\*50%）px, 高度是400（1000\*40%）px。当就设置一个百分比时，宽高都是这个百分比，会拉伸图片（宽高比例与原图不一致），图片完整

* background-size:cover;

背景图片填满整个元素，此时背景图是不居中显示，如果需要居中显示需要设置background-position: center;。填满父元素是图片根据左上角按图片宽高比例等比例缩放，直到填满整个父元素，图片不完整，不会拉伸图片



* background-size: contain;

维持： 1.背景图原来的宽高比，2.确保在元素中全部显示图片。和cover类似，只不过cover是填满父元素，contain是等比例缩放图片，当图片的一边和父元素的某一边重合，停止缩放，有可能没有填满父元素，图片完整，不拉伸图片，如果没有填满父元素，默认会平铺



左图设置了background-repeat: no-repeat;

所以没有平铺