

网易微专业之《前端开发工程师》

学习笔记

开始时间：2015.12.17

《页面制作》

CSS（七）

变形 transform-2d 变形

在 CSS3 中，我们可以使用 transform 属性来实现文字或图像的的各种变形效果，如**位移、缩放、旋转、倾斜**等。

transform 属性变形方法

方法或属性	说明
translate()	位移
scale()	缩放
rotate()	旋转
skew()	倾斜
transform-origin	中心原点

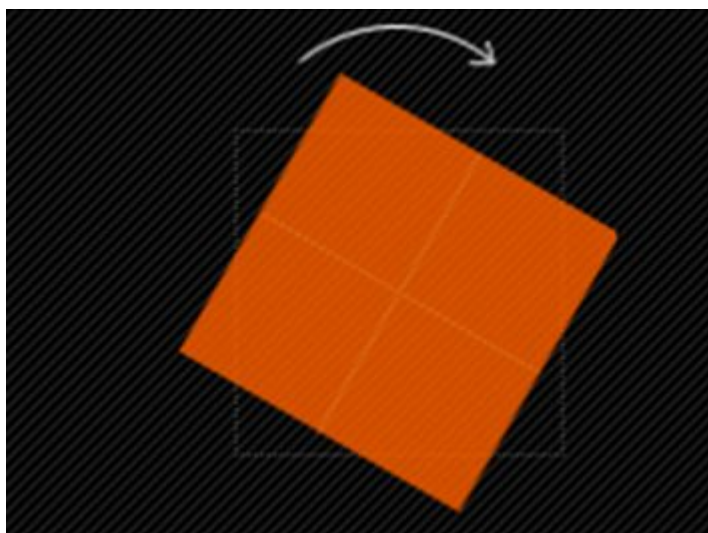
旋转 rotate()方法

在 CSS3 中，我们可以使用 rotate()方法来将元素相对中心原点进行旋转。

语法：transform: rotate(度数);

说明：度数指的是元素相对中心原点进行旋转的度数，单位为 deg。其中，deg 是 degree（度数）的缩写。

它主要在二维空间内进行操作，设置一个角度值，用来指定旋转的幅度。如果这个值为**正值**，元素相对原点中心**顺时针**旋转；如果这个值为**负值**，元素相对原点中心**逆时针**旋转。如下图所示：



位移 `translate()` 方法

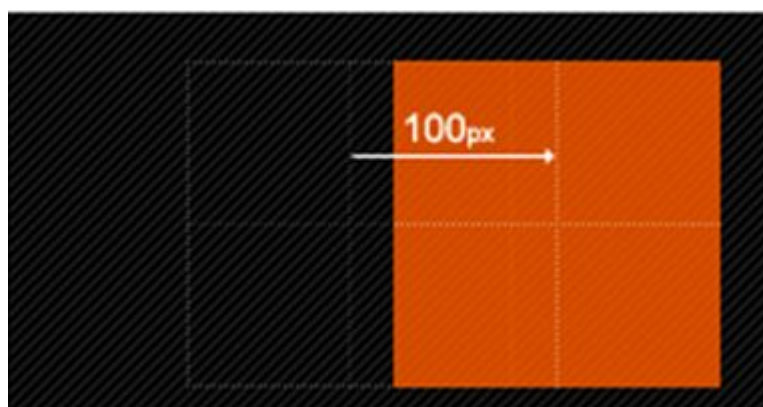
在 CSS3 中，我们可以使用 `translate()` 方法将元素沿着水平方向（X 轴）和垂直方向（Y 轴）移动。

对于位移 `translate()` 方法，我们分为 3 种情况：

（1）**`translateX(x)`**：元素仅在水平方向移动（X 轴移动）；

语法：`transform: translateX(x);`

说明：在 CSS3 中，所有变形方法都是属于 `transform` 属性，因此所有关于变形的地方都要加上“`transform:`”，以表示“变形”处理。x 表示元素在水平方向（X 轴）的移动距离，单位为 px、em 或百分比等。

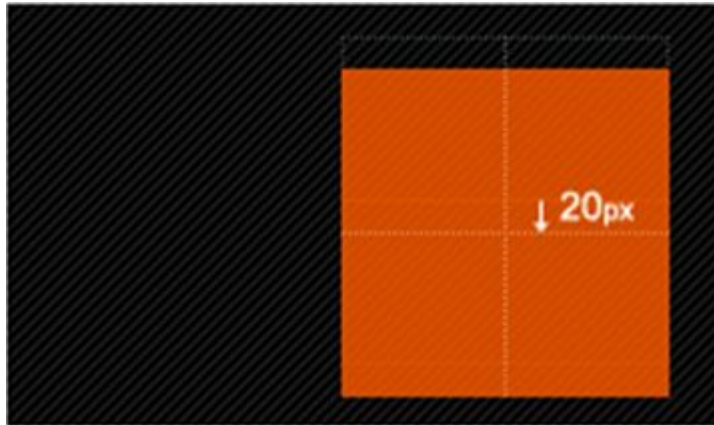


（2）**`translateY(y)`**：元素仅在垂直方向移动（Y 轴移动）；

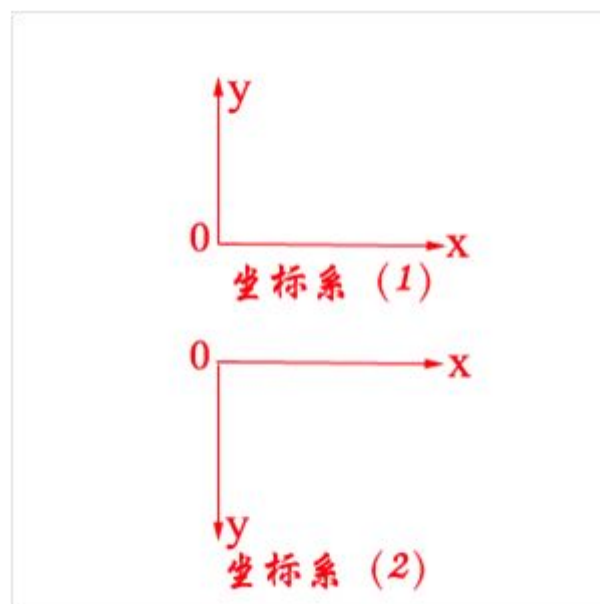
语法：transform:translateY(y)

说明：y 表示元素在垂直方向（y 轴）的移动距离，单位为 px、em 或百分比等。

当 y 为正时，表示元素在垂直方向向下移动；当 y 为负时，表示元素在垂直方向向上移动。



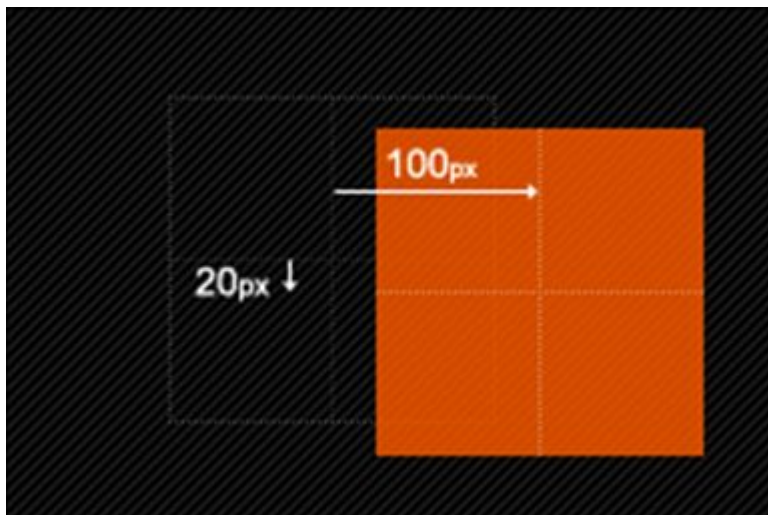
在 W3C 规定中，出于人的习惯是从上到下阅读，所选取的坐标系为下图中的第 2 种坐标系，因此 x 轴正方向向右，而 y 轴正方向向下。而你所想到的坐标系应该是下图中的第 1 种坐标系，这种坐标系是“数学形式”的坐标系，只适合在数学应用上。大家一定要搞清楚这一点！



(3) translate(x,y) 元素在水平方向和垂直方向同时移动(X 轴和 Y 轴同时移动)；

语法：transform:translate(x,y)

说明：x 表示元素在水平方向（x 轴）的移动距离，y 表示元素在垂直方向（y 轴）的移动距离。注意，Y 是一个可选参数，如果没有设置 Y 值，则表示元素仅仅沿着 X 轴正方形移动。



缩放 scale()方法

缩放，指的是“缩小”和“放大”。在 CSS3 中，我们可以使用 scale()方法来将元素根据中心原点进行缩放。

缩放 scale()方法也有 3 种情况：

(1) scaleX(x)：元素仅水平方向缩放（X 轴缩放）；

语法：transform:scaleX(x)

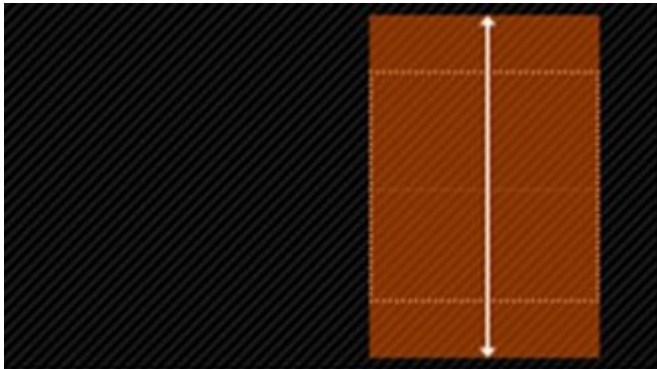
说明：x 表示元素沿着水平方向（X 轴）缩放的倍数，如果大于 1 就代表放大；如果小于 1 就代表缩小。



(2) `scaleY(y)` : 元素仅垂直方向缩放 (Y 轴缩放) ;

语法 : `transform:scaleY(y)`

说明 : `y` 表示元素沿着垂直方向 (Y 轴) 缩放的倍数 , 如果大于 1 就代表放大 ; 如果小于 1 就代表缩小。

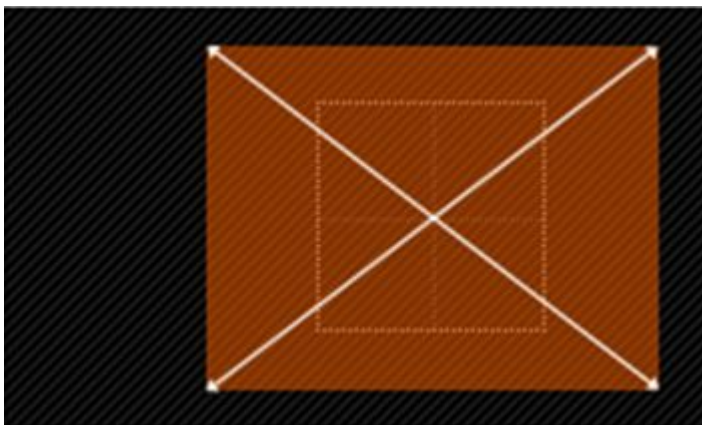


(3) `scale(x,y)` : 元素水平方向和垂直方向同时缩放 (X 轴和 Y 轴同时缩放) ;

语法 : `transform:scale(x,y)`

说明 : `x` 表示元素沿着水平方向 (X 轴) 缩放的倍数 , `y` 表示元素沿着垂直方向 (Y 轴) 缩放的倍数。

注意 , `Y` 是一个可选参数 , 如果没有设置 `Y` 值 , 则表示 `X`、`Y` 两个方向的缩放倍数是一样的 (同时放大相同倍数)。



倾斜 skew()方法

它可以将一个对象以其中心位置围绕着 **X 轴**和 **Y 轴**按照一定的角度倾斜。这与 rotate()函数的旋转不同，rotate()函数只是旋转，而不会改变元素的形状。skew()函数不会旋转，而只会改变元素的形状。

在 CSS3 中，我们可以使用 skew()方法将元素倾斜显示。

skew()方法跟 translate()方法、scale()方法一样，也有 3 种情况：

(1) skewX(x)：使元素在水平方向倾斜 (X 轴倾斜) ；`transform:skewX(x);`

(2) skewY(y)：使元素在垂直方向倾斜 (Y 轴倾斜) ；`transform:skewY(y);`

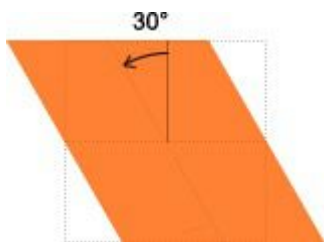
(3) skew(x,y)：使元素在水平方向和垂直方向同时倾斜 (X 轴和 Y 轴同时倾斜) ；

`transform:skew(x,y);`

说明：第一个参数对应 X 轴，第二个参数对应 Y 轴。如果第二个参数未提供，则值为 0，也就是 Y 轴方向上无斜切。

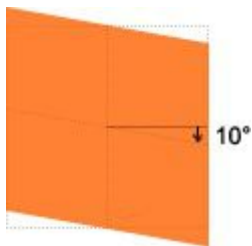
理解：skew 的默认原点 transform-origin 是这个物件的中心点

skewX(30deg) 如下图： (沿着 X 轴倾斜，沿着 Y 轴呈逆时针旋转 30 度。)

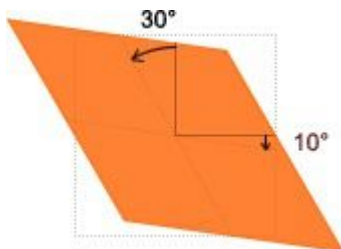


skewY(10deg) 如下图： (沿着 y 轴倾斜，沿着 X 轴呈顺时针旋转 10 度。)

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>



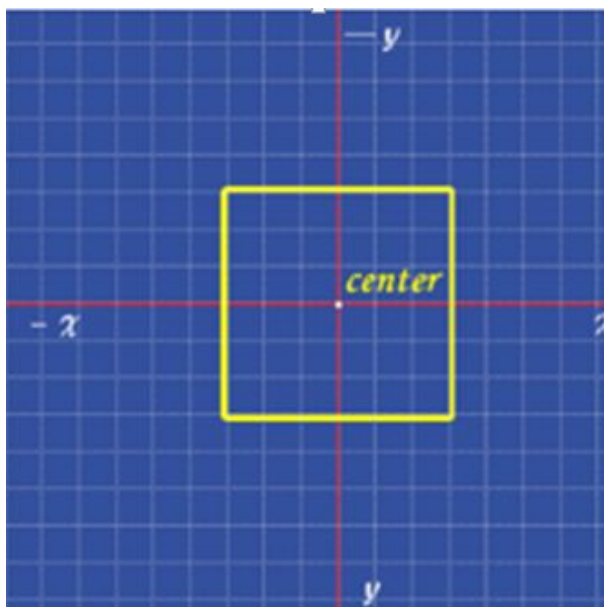
skew(30deg, 10deg) 如下图：



transform-origin 属性

任何一个元素都有一个中心原点，默认情况下，元素的中心原点位于 X 轴和 Y 轴的 50%处，

如下图所示：



默认情况下，**CSS3** 变形进行的位移、缩放、旋转、倾斜都是以元素的中心原点(50%,50%)

进行变形。但很多时候，我们可以通过 transform-origin 属性来改变元素变形时的中心原点位置。

语法：

transform-origin: [<percentage> | <length> | left | center① | right] [<percentage> | <length> | top | center② | bottom]?

默认值：50% 50%，效果等同于 center center

不管 transform-origin 取值为长度值还是关键字，都需要设置水平方向和垂直方向的值。

transform-origin 属性取值

关键字	百分比	说明
top left	0 0	左上
top center	50% 0	靠上居中
top right	100% 0	右上
left center	0 50%	靠左居中
center center	50% 50%	正中
right center	100% 50%	靠右居中
bottom left	0 100%	左下
bottom center	50% 100%	靠下居中
bottom right	100% 100%	右下

参考知识：

1. <http://www.zhangxinxu.com/study/201206/css3-transform-matrix-skew.html>

2. 变形--矩阵 matrix()

matrix() 是一个含六个值的(a,b,c,d,e,f)变换矩阵，用来指定一个 2D 变换，相当于直接应用一个[a b c d e f]变换矩阵。就是基于水平方向（X 轴）和垂直方向（Y 轴）重新定位元素，此属性值使用涉及到数学中的矩阵，我在这里只是简单的说一下 CSS3 中的 transform 有这么一个属性值，如果需要深入了解，需要对数学矩阵有一定的知识。

Matrix 深入理解：

<http://www.zhangxinxu.com/wordpress/2012/06/css3-transform-matrix-%E7%9F%A9%E9%98%B5/>

示例演示：通过 matrix()函数来模拟 transform 中 translate()位移的效果。

HTML 代码：

```
<div class="wrapper">
  <div></div>
</div>
```


CSS 代码：

```
.wrapper {  
  width: 300px;  
  height: 200px;  
  border: 2px dotted red;  
  margin: 40px auto;  
}  
  
.wrapper div {  
  width: 300px;  
  height: 200px;  
  background: orange; -webkit-transform: matrix(1, 0, 0, 1, 50, 50);  
  -moz-transform: matrix(1, 0, 0, 1, 50, 50);  
  transform: matrix(1, 0, 0, 1, 50, 50);  
}
```

演示结果：



transform-3d 变形

Perspective 属性-透视

语法：Perspective:none | <length>

perspective 属性定义 3D 元素距视图的距离，以像素计。

该属性允许改变 3D 元素查看 3D 元素的视图。当为元素定义 **perspective** 属性时，其子元素会获得透视效果，而不是元素本身。

理解：

当 **perspective:none/0** 时，相当于没有设 **perspective(length)**。比如要建立一个立方体，长宽高都是 200px。如果 **perspective < 200px**，那就相当于站在盒子里面看的结果，如果 **perspective** 非常大那就是站在非常远的地方看（立方体已经成了小正方形了）。

当元素没有设置 **perspective(length)** 时，所有后代元素被压缩在同一个二维平面上，不存在景深的效果。如果设置 **perspective(length)** 后，将会看到三维的效果。默认的透视视角

中心在容器(是 perspective 所在的元素，不是他的后代元素)的中点，也就是 perspective-origin: 50% 50%。当然你也可以自己设置，比如：左上角 -webkit-perspective-origin: 0px 0px;。

Perspective-origin 属性

perspective-origin 属性定义 3D 元素所基于的 X 轴和 Y 轴。该属性允许您改变 3D 元素的底部位置。

定义时的 perspective -origin 属性，它是一个元素的子元素，透视图，而不是元素本身。

语法：perspective-origin: x-axis y-axis;

值	描述
x-axis	定义该视图在 x 轴上的位置。默认值：50%。 可能的值： left center right length %
y-axis	定义该视图在 y 轴上的位置。默认值：50%。 可能的值： top center bottom length %

Translate3d-指定对象的 3D 位移。

语法：translate3d(<translation-value>,<translation-value>,<length>)

第 1 个参数对应 X 轴，第 2 个参数对应 Y 轴，第 3 个参数对应 Z 轴，参数不允许省略。

translate3d()

云课堂

translate3d(<translation-value> , <translation-value> ,
<length>)

translateX(<translation-value>)

translateY(<translation-value>)

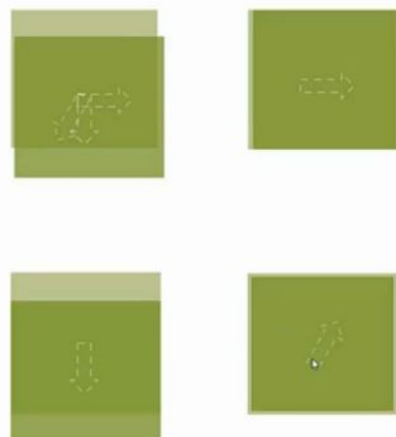
translateZ(<length>)

transform : translate3d(10px , 20% , 50px) ;

transform : translateX(10px) ;

transform : translateY(20%) ;

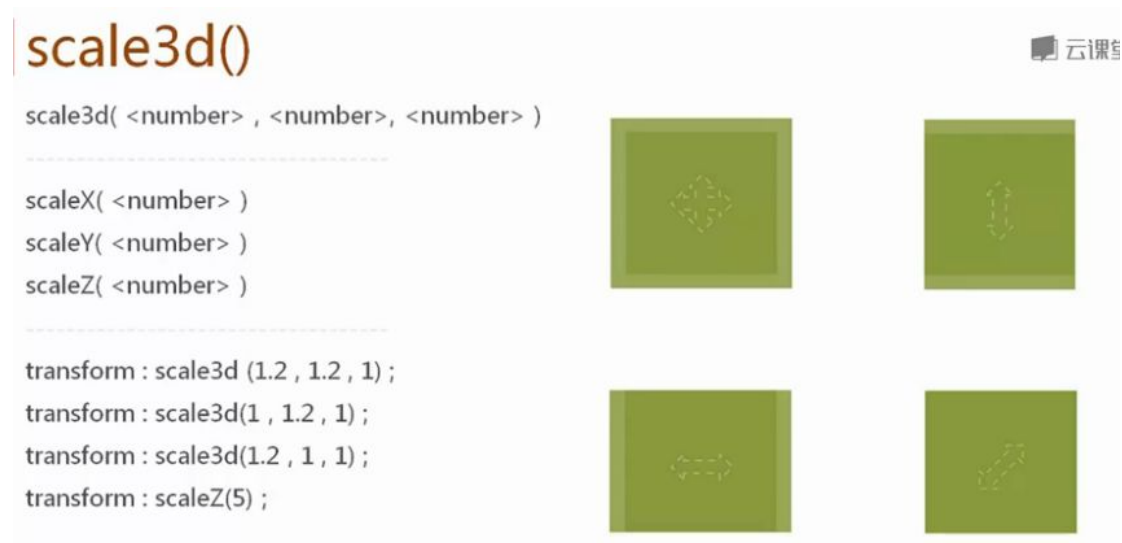
transform : translateZ(-100px) ;



Scale3d()-指定对象的 3D 缩放

语法: `scale3d(<number>, <number>, <number>)`

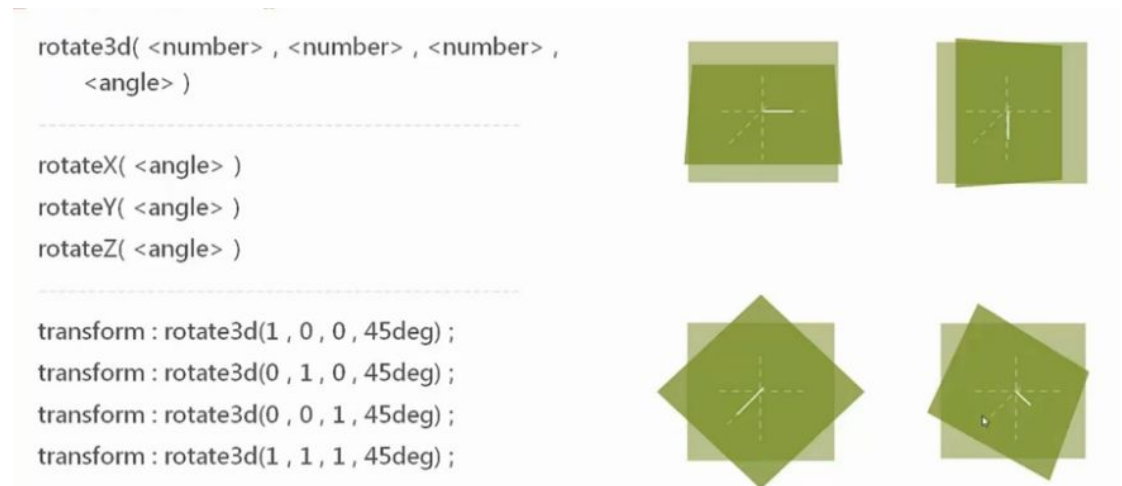
第 1 个参数对应 X 轴，第 2 个参数对应 Y 轴，第 3 个参数对应 Z 轴，参数不允许省略



Rotate3d()-指定对象的 3D 旋转角度。

语法: `rotate3d(<number>, <number>, <number>, <angle>)`

其中前 3 个参数分别表示旋转的方向 x, y, z，第 4 个参数表示旋转的角度，参数不允许省略



Transform-style

`transform-style` 属性指定嵌套元素是怎样在三维空间中呈现。

语法: `transform-style: flat|preserve-3d;`

`transform-style : flat | preserve-3d`

`transform-style : flat ;`

`transform-style : preserve-3d ;`



- flat 值为默认值，表示所有子元素在 2D 平面呈现。
- preserve-3d 表示所有子元素在 3D 空间中呈现。

也就是说，如果对一个元素设置了 transform-style 的值为 flat，则该元素的所有子元素都将被平展到该元素的 2D 平面中进行呈现。沿着 X 轴或 Y 轴方向旋转该元素将导致位于正或负 Z 轴位置的子元素显示在该元素的平面上，而不是它的前面或者后面。如果对一个元素设置了 transform-style 的值为 preserve-3d，它表示不执行平展操作，他的所有子元素位于 3D 空间中。

Backface-visibility

backface-visibility 属性定义当元素不面向屏幕时是否可见。

如果在旋转元素不希望看到其背面时，该属性很有用。

语法:backface-visibility: visible|hidden;

`backface-visibility : visible | hidden`

`backface-visibility : visible ;`

`backface-visibility : hidden ;`



参考资源：

1. CSS 3D transform 变换：

<http://www.zhangxinxu.com/wordpress/2012/09/css3-3d-transform-perspective-animate-transition/>

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

<http://www.zhangxinxu.com/wordpress/2010/11/css3-transitions-transforms%E5%92%8Canimation%E4%BD%BF%E7%94%A8%E7%AE%80%E4%BB%8B%E4%B8%8E%E5%BA%94%E7%94%A8%E5%B1%95%E7%A4%BA/>

<http://www.787866.com/2083.html>

2. CSS 动画视图在线工具：

<http://www.w3cways.com/css3-animation-tool>

CSS（八）

过渡 & 动画

Transition 过渡

在 CSS3 中，我们可以使用 `transition` 属性来将元素的某一个属性从“一个属性值”在指定的时间内平滑地过渡到“另外一个属性值”来实现动画效果。

CSS `transform` 属性所实现的元素变形，呈现的仅仅是一个“结果”，而 CSS `transition` 呈现的是一种过渡“过程”，通俗点说就是一种动画转换过程，如渐显、渐隐、动画快慢等。

语法： `transition: 属性 持续时间 过渡方法 延迟时间;`

在 CSS 中创建简单的过渡效果可以从以下几个步骤来实现：

1. 在默认样式中声明元素的初始状态样式；
2. 声明过渡元素最终状态样式，比如悬浮状态；
3. 在默认样式中通过添加过渡函数，添加一些不同的样式。

CSS3 的过渡 `transition` 属性是一个复合属性，主要包含 4 个子属性：

transition-property：对元素的哪一个属性进行操作；

使用 `transition-property` 属性来单独设定过渡动画所要操作的那个属性。

语法：

```
transition-property: none | <single-transition-property>[ , <single-transition-property> ]*  
<single-transition-property> = all | <IDENT>
```

默认值：all。默认为所有可以进行过渡的 CSS 属性。

说明： `transition-property` 属性的取值是一个“CSS 属性名”。

对于 CSS3 过渡动画，大多数情况下都是配合 `hover 伪类` 来使用。其对应具有过渡的 CSS 属性主要有：

background-color↵	background-position↵	border-bottom-color↵	border-bottom-width↵
border-left-color↵	border-left-width↵	border-right-color↵	border-right-width↵
border-spacing↵	border-top-color↵	border-top-width↵	bottom↵
clip↵	color↵	font-size↵	font-weight↵
height↵	left↵	letter-spacing↵	line-height↵
margin-bottom↵	margin-left↵	margin-right↵	margin-top↵
max-height↵	max-width↵	min-height↵	min-width↵
opacity↵	outline-color↵	outline-width↵	padding-bottom↵
padding-left↵	padding-right↵	padding-top↵	right↵
text-indent↵	text-shadow↵	vertical-align↵	visibility↵
width↵	word-spacing↵	z-index↵	↵

transition-duration : 过渡的持续时间；

语法： transition-duration: 时间；

说明： transition-duration 属性取值是一个时间，单位为 s（秒），可以为小数如 0.5s。如果提供多个属性值，以逗号进行分隔。

transition-timing-function : 过渡使用的方法（函数）；

使用 transition-timing-function 属性来定义过渡方式。所谓的“过渡方式”主要用来指定动画在过渡时间内的速率。

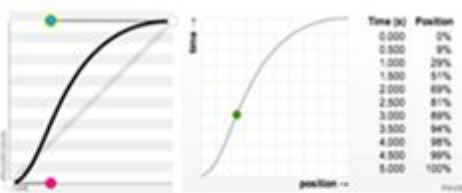
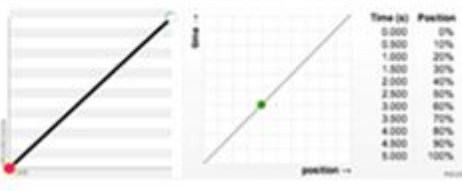
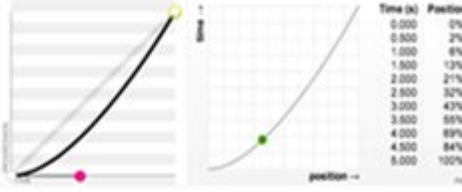


语法：

transition-timing-function:

```
<single-transition-timing-function>[, <single-transition-timing-function>]*
<single-transition-timing-function> = ease | linear | ease-in | ease-out |
ease-in-out | step-start | step-end | steps(<integer>[, [ start | end ] ]?) |
cubic-bezier(<number>, <number>, <number>, <number>)
```

默认值: ease

transition-timing-function 属性取值共有 5 种，具体如下：

函数	功能描述	图例																								
ease	默认值，元素样式从初始状态过渡到终止状态时速度由快到慢，逐渐变慢。	 <table><thead><tr><th>Time (s)</th><th>Position</th></tr></thead><tbody><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>9%</td></tr><tr><td>1.000</td><td>29%</td></tr><tr><td>1.500</td><td>51%</td></tr><tr><td>2.000</td><td>69%</td></tr><tr><td>2.500</td><td>81%</td></tr><tr><td>3.000</td><td>89%</td></tr><tr><td>3.500</td><td>94%</td></tr><tr><td>4.000</td><td>96%</td></tr><tr><td>4.500</td><td>98%</td></tr><tr><td>5.000</td><td>100%</td></tr></tbody></table>	Time (s)	Position	0.000	0%	0.500	9%	1.000	29%	1.500	51%	2.000	69%	2.500	81%	3.000	89%	3.500	94%	4.000	96%	4.500	98%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	9%																									
1.000	29%																									
1.500	51%																									
2.000	69%																									
2.500	81%																									
3.000	89%																									
3.500	94%																									
4.000	96%																									
4.500	98%																									
5.000	100%																									
linear	元素样式从初始状态过渡到终止状态速度是恒速。	 <table><thead><tr><th>Time (s)</th><th>Position</th></tr></thead><tbody><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>10%</td></tr><tr><td>1.000</td><td>20%</td></tr><tr><td>1.500</td><td>30%</td></tr><tr><td>2.000</td><td>40%</td></tr><tr><td>2.500</td><td>50%</td></tr><tr><td>3.000</td><td>60%</td></tr><tr><td>3.500</td><td>70%</td></tr><tr><td>4.000</td><td>80%</td></tr><tr><td>4.500</td><td>90%</td></tr><tr><td>5.000</td><td>100%</td></tr></tbody></table>	Time (s)	Position	0.000	0%	0.500	10%	1.000	20%	1.500	30%	2.000	40%	2.500	50%	3.000	60%	3.500	70%	4.000	80%	4.500	90%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	10%																									
1.000	20%																									
1.500	30%																									
2.000	40%																									
2.500	50%																									
3.000	60%																									
3.500	70%																									
4.000	80%																									
4.500	90%																									
5.000	100%																									
ease-in	元素样式从初始状态过渡到终止状态时，速度越来越快，呈一种加速状态。常称这种效果为渐显效果。	 <table><thead><tr><th>Time (s)</th><th>Position</th></tr></thead><tbody><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>2%</td></tr><tr><td>1.000</td><td>6%</td></tr><tr><td>1.500</td><td>13%</td></tr><tr><td>2.000</td><td>21%</td></tr><tr><td>2.500</td><td>32%</td></tr><tr><td>3.000</td><td>43%</td></tr><tr><td>3.500</td><td>55%</td></tr><tr><td>4.000</td><td>69%</td></tr><tr><td>4.500</td><td>84%</td></tr><tr><td>5.000</td><td>100%</td></tr></tbody></table>	Time (s)	Position	0.000	0%	0.500	2%	1.000	6%	1.500	13%	2.000	21%	2.500	32%	3.000	43%	3.500	55%	4.000	69%	4.500	84%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	2%																									
1.000	6%																									
1.500	13%																									
2.000	21%																									
2.500	32%																									
3.000	43%																									
3.500	55%																									
4.000	69%																									
4.500	84%																									
5.000	100%																									
ease-out	元素样式从初始状态过渡到终止状态时，速度越来越慢，呈一种减速状态。常称这种效果为渐隐效果。	 <table><thead><tr><th>Time (s)</th><th>Position</th></tr></thead><tbody><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>2%</td></tr><tr><td>1.000</td><td>6%</td></tr><tr><td>1.500</td><td>13%</td></tr><tr><td>2.000</td><td>21%</td></tr><tr><td>2.500</td><td>32%</td></tr><tr><td>3.000</td><td>43%</td></tr><tr><td>3.500</td><td>55%</td></tr><tr><td>4.000</td><td>69%</td></tr><tr><td>4.500</td><td>84%</td></tr><tr><td>5.000</td><td>100%</td></tr></tbody></table>	Time (s)	Position	0.000	0%	0.500	2%	1.000	6%	1.500	13%	2.000	21%	2.500	32%	3.000	43%	3.500	55%	4.000	69%	4.500	84%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	2%																									
1.000	6%																									
1.500	13%																									
2.000	21%																									
2.500	32%																									
3.000	43%																									
3.500	55%																									
4.000	69%																									
4.500	84%																									
5.000	100%																									
ease-in-out	元素样式从初始状态到终止状态时，先加速再减速。常称这种效果为渐显渐隐效果。	 <table><thead><tr><th>Time (s)</th><th>Position</th></tr></thead><tbody><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>2%</td></tr><tr><td>1.000</td><td>6%</td></tr><tr><td>1.500</td><td>19%</td></tr><tr><td>2.000</td><td>33%</td></tr><tr><td>2.500</td><td>50%</td></tr><tr><td>3.000</td><td>67%</td></tr><tr><td>3.500</td><td>81%</td></tr><tr><td>4.000</td><td>92%</td></tr><tr><td>4.500</td><td>96%</td></tr><tr><td>5.000</td><td>100%</td></tr></tbody></table>	Time (s)	Position	0.000	0%	0.500	2%	1.000	6%	1.500	19%	2.000	33%	2.500	50%	3.000	67%	3.500	81%	4.000	92%	4.500	96%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	2%																									
1.000	6%																									
1.500	19%																									
2.000	33%																									
2.500	50%																									
3.000	67%																									
3.500	81%																									
4.000	92%																									
4.500	96%																									
5.000	100%																									

取值：

- linear: 线性过渡。等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)
- ease: 平滑过渡。等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)
- ease-in: 由慢到快。等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)
- ease-out: 由快到慢。等同于贝塞尔曲线(0, 0, 0.58, 1.0)
- ease-in-out: 由慢到快再到慢。等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)
- step-start: 等同于 steps(1, start)
- step-end: 等同于 steps(1, end)
- steps(<integer>[, [start | end]]?): 接受两个参数的步进函数。第一个参数必须为正整数，指定函数的步数。第二个参数取值可以是 start 或 end，指定每一步的值发生变化的时间点。第二个参数是可选的，默认值为 end。
- cubic-bezier(<number>, <number>, <number>, <number>): 特定的贝塞尔曲线类型，

4 个数值需在 [0, 1] 区间内

transition-delay：可选属性，指定过渡开始出现的延迟时间；

语法：**transition-delay**:时间;

说明：transition-delay 属性取值是一个时间，单位为 s（秒），可以为小数如 0.5s。

transition-delay 属性默认值为 0，也就是说当我们没有设置 transition-delay 属性时，过渡动画就没有延迟时间。如果提供多个属性值，以逗号进行分隔。

案例：

```
<style type="text/css">
  div
  {
    display:inline-block;
    width:100px;
    height:100px;
    border-radius:0;
    background-color:#14C7F3;
    transition-property:border-radius,background-color;
    transition-duration:1s ;
    transition-timing-function:linear;
    transition-delay:2s;
  }
  div:hover
  {
    border-radius:50px;
    background-color:red;
  }
</style>
</head>
<body>
  <div></div>
```

transition

如果想要使用 **transition 属性** 同时对多个属性进行实现平滑过渡效果，只需要在

transition-property 属性 添加多个属性名即可，其中属性名之间用英文逗号隔开。然后各自可以有各自不同的延续时间和其时间的速率变换方式。但需要值得注意的一点：第一个时间的值为 transition-duration，第二个为 transition-delay。

例如：a{ transition: background 0.8s ease-in 0.3,color 0.6s ease-out 0.3;}

Animation 动画

在 CSS3 中，动画效果使用 animation 属性来实现。animation 属性和 transition 属性功能是一样的，都是通过改变元素的“属性值”来实现动画效果。但是这两者又有很大的区别：transition 属性只能通过指定属性的开始值与结束值，然后在这两个属性值之间进行平滑过渡来实现动画效果，因此只能实现简单的动画效果。animation 属性则通过定义多个关键帧以及定义每个关键帧中元素的属性值来实现复杂的动画效果。transition 属性只能实现简单的动画（一个），而 animation 属性却可以实现复杂的动画（一系列）。

定义动画@keyframes (keyframes:关键帧)

使用 animation 属性定义 CSS3 动画需要 2 步：

- (1) 定义动画；
- (2) 调用动画；

在 CSS3 中，在使用动画之前，我们必须使用@keyframes 规则定义动画。

语法：

```
@keyframes <identifier> '{' <keyframes-blocks> '}' ;  
<keyframes-blocks>: [ [ from | to | <percentage> ] { sRules } ] [ [ , from | to  
| <percentage> ] { sRules } ]*
```

取值：

- <identifier>: identifier 定义一个动画名称
 - <keyframes-blocks>: 定义动画在每个阶段的样式，即帧动画。
1. 定义动画时，简单的动画可以直接使用关键字 from 和 to，即从一种状态过渡到另一种状态：

示例代码：

```
@keyframes testanimations {  
    from { opacity: 1; }  
    to { opacity: 0; }  
}
```

其中 testanimations 是该动画的名字，该动画表示某个东西将逐渐消失。

2. 如果复杂的动画，可以混合<percentage>去设置某个时间段内的任意时间点的样式：

示例代码：

```
@keyframes testanimations {  
    from { transform: translate(0, 0); }  
    20% { transform: translate(20px, 20px); }  
    40% { transform: translate(40px, 0); }  
    60% { transform: translate(60px, 20); }  
    80% { transform: translate(80px, 0); }  
    to { transform: translate(100px, 20px); }  
}
```

3. 当然，也可以不使用关键字 from 和 to，而都使用<percentage>：

示例代码：

```
@keyframes testanimations{
    0% { transform: translate(0, 0); }
    20% { transform: translate(20px, 20px); }
    40% { transform: translate(40px, 0); }
    60% { transform: translate(60px, 20px); }
    80% { transform: translate(80px, 0); }
    100% { transform: translate(100px, 20px); }
}
```

注意，这里的 0%不能简写成 0。

使用@keyframes 规则时，如果仅仅只有 0%和 100%这两个百分比的话，这时 0%和 100%还可以使用关键词 from 和 to 来代表，其中 0%对应的是 from，100%对应的是 to。

调用动画 animation-name 属性

在 CSS3 中，使用@keyframes 规则定义的动画并不会自动执行，我们还需要使用 animation-name 属性来调用动画，之后动画才会生效。

语法：animation-name: 动画名;

说明：注意，animation-name 调用的动画名需要和@keyframes 规则定义的动画名称完全一致（区分大小写），如果不一致将不具有任何动画效果。为了浏览器兼容性，针对 Chrome 和 Safari 浏览器需要加上-webkit-前缀，而针对 Firefox 浏览器需要加上-moz-。

持续时间 animation-duration 属性

在 CSS3 中，我们可以使用 animation-duration 属性来设置动画的持续时间，也就是完成从 0%到 100%所使用的总时间。

语法：animation-duration: <time>[, <time>]*

说明：animation-duration 属性取值是一个时间，单位为 s（秒），可以为小数如 0.5s。

播放方式 animation-timing-function 属性

在 CSS3 中，我们可以使用 animation-timing-function 属性来设置动画的播放方式，所谓的“播放方式”主要用来指定动画在播放时间内的速率。其中，

语法：

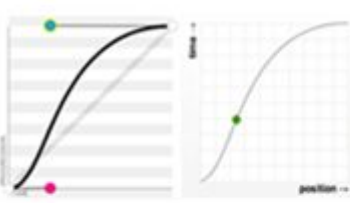
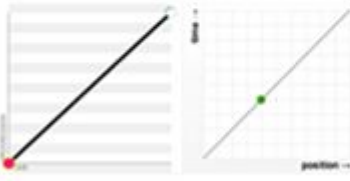
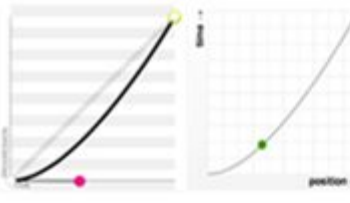
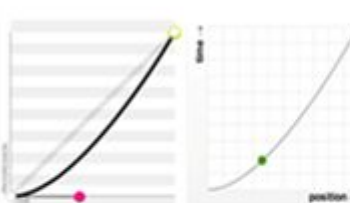
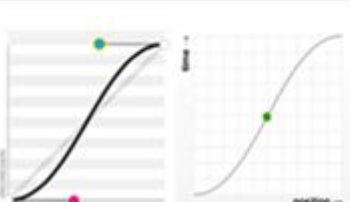
animation-timing-function:

<single-animation-timing-function> [, <single-animation-timing-function>]*

<single-animation-timing-function> = ease | linear | ease-in | ease-out |
ease-in-out | step-start | step-end | steps(<integer> [, [start | end]]?) |
cubic-bezier(<number>, <number>, <number>, <number>)

默认值: ease

说明：animation-timing-function 属性取值跟 transition-timing-function 属性取值一样，共有 5 种，具体如下：

函数	功能描述	图例																								
ease	默认值，元素样式从初始状态过渡到终止状态时速度由快到慢，逐渐变慢。	 <table><tr><th>Time (s)</th><th>Position</th></tr><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>9%</td></tr><tr><td>1.000</td><td>29%</td></tr><tr><td>1.500</td><td>51%</td></tr><tr><td>2.000</td><td>69%</td></tr><tr><td>2.500</td><td>81%</td></tr><tr><td>3.000</td><td>89%</td></tr><tr><td>3.500</td><td>94%</td></tr><tr><td>4.000</td><td>96%</td></tr><tr><td>4.500</td><td>98%</td></tr><tr><td>5.000</td><td>100%</td></tr></table>	Time (s)	Position	0.000	0%	0.500	9%	1.000	29%	1.500	51%	2.000	69%	2.500	81%	3.000	89%	3.500	94%	4.000	96%	4.500	98%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	9%																									
1.000	29%																									
1.500	51%																									
2.000	69%																									
2.500	81%																									
3.000	89%																									
3.500	94%																									
4.000	96%																									
4.500	98%																									
5.000	100%																									
linear	元素样式从初始状态过渡到终止状态速度是恒速。	 <table><tr><th>Time (s)</th><th>Position</th></tr><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>10%</td></tr><tr><td>1.000</td><td>20%</td></tr><tr><td>1.500</td><td>30%</td></tr><tr><td>2.000</td><td>40%</td></tr><tr><td>2.500</td><td>50%</td></tr><tr><td>3.000</td><td>60%</td></tr><tr><td>3.500</td><td>70%</td></tr><tr><td>4.000</td><td>80%</td></tr><tr><td>4.500</td><td>90%</td></tr><tr><td>5.000</td><td>100%</td></tr></table>	Time (s)	Position	0.000	0%	0.500	10%	1.000	20%	1.500	30%	2.000	40%	2.500	50%	3.000	60%	3.500	70%	4.000	80%	4.500	90%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	10%																									
1.000	20%																									
1.500	30%																									
2.000	40%																									
2.500	50%																									
3.000	60%																									
3.500	70%																									
4.000	80%																									
4.500	90%																									
5.000	100%																									
ease-in	元素样式从初始状态过渡到终止状态时，速度越来越快，呈一种加速状态。常称这种效果为渐显效果。	 <table><tr><th>Time (s)</th><th>Position</th></tr><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>2%</td></tr><tr><td>1.000</td><td>6%</td></tr><tr><td>1.500</td><td>13%</td></tr><tr><td>2.000</td><td>21%</td></tr><tr><td>2.500</td><td>32%</td></tr><tr><td>3.000</td><td>43%</td></tr><tr><td>3.500</td><td>56%</td></tr><tr><td>4.000</td><td>69%</td></tr><tr><td>4.500</td><td>84%</td></tr><tr><td>5.000</td><td>100%</td></tr></table>	Time (s)	Position	0.000	0%	0.500	2%	1.000	6%	1.500	13%	2.000	21%	2.500	32%	3.000	43%	3.500	56%	4.000	69%	4.500	84%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	2%																									
1.000	6%																									
1.500	13%																									
2.000	21%																									
2.500	32%																									
3.000	43%																									
3.500	56%																									
4.000	69%																									
4.500	84%																									
5.000	100%																									
ease-out	元素样式从初始状态过渡到终止状态时，速度越来越慢，呈一种减速状态。常称这种效果为渐隐效果。	 <table><tr><th>Time (s)</th><th>Position</th></tr><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>2%</td></tr><tr><td>1.000</td><td>6%</td></tr><tr><td>1.500</td><td>13%</td></tr><tr><td>2.000</td><td>21%</td></tr><tr><td>2.500</td><td>32%</td></tr><tr><td>3.000</td><td>43%</td></tr><tr><td>3.500</td><td>56%</td></tr><tr><td>4.000</td><td>69%</td></tr><tr><td>4.500</td><td>84%</td></tr><tr><td>5.000</td><td>100%</td></tr></table>	Time (s)	Position	0.000	0%	0.500	2%	1.000	6%	1.500	13%	2.000	21%	2.500	32%	3.000	43%	3.500	56%	4.000	69%	4.500	84%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	2%																									
1.000	6%																									
1.500	13%																									
2.000	21%																									
2.500	32%																									
3.000	43%																									
3.500	56%																									
4.000	69%																									
4.500	84%																									
5.000	100%																									
ease-in-out	元素样式从初始状态到终止状态时，先加速再减速。常称这种效果为渐显渐隐效果。	 <table><tr><th>Time (s)</th><th>Position</th></tr><tr><td>0.000</td><td>0%</td></tr><tr><td>0.500</td><td>2%</td></tr><tr><td>1.000</td><td>8%</td></tr><tr><td>1.500</td><td>19%</td></tr><tr><td>2.000</td><td>33%</td></tr><tr><td>2.500</td><td>50%</td></tr><tr><td>3.000</td><td>67%</td></tr><tr><td>3.500</td><td>81%</td></tr><tr><td>4.000</td><td>92%</td></tr><tr><td>4.500</td><td>98%</td></tr><tr><td>5.000</td><td>100%</td></tr></table>	Time (s)	Position	0.000	0%	0.500	2%	1.000	8%	1.500	19%	2.000	33%	2.500	50%	3.000	67%	3.500	81%	4.000	92%	4.500	98%	5.000	100%
Time (s)	Position																									
0.000	0%																									
0.500	2%																									
1.000	8%																									
1.500	19%																									
2.000	33%																									
2.500	50%																									
3.000	67%																									
3.500	81%																									
4.000	92%																									
4.500	98%																									
5.000	100%																									

延迟时间 animation-delay 属性

在 CSS3 中，我们可以使用 animation-delay 属性来定义动画播放的延迟时间。

语法：animation-delay:时间;

说明：animation-delay 属性取值是一个时间，单位为 s(秒)，可以为小数如 0.5s。

animation-delay 属性默认值为 0，也就是说当我们没有设置 animation-delay 属性时，CSS3 动画就没有延迟时间。

播放次数 animation-iteration-count 属性

在 CSS3 中，我们可以使用 animation-iteration-count 属性来定义动画的播放次数。

语法：animation-iteration-count: infinite | <number> [, infinite | <number>]*

说明：animation-iteration-count 属性取值有 2 种：

- 正整数;
- infinite;

animation-iteration-count 属性默认值为 1。也就是默认情况下，动画从开始到结束只播放一次。“animation-iteration-count:n”表示动画播放 n 次，n 为正整数；当 animation-iteration-count 属性取值为 infinite 时，动画会无限次地循环播放。

播放方向 animation-direction 属性

在 CSS3 中，我们可以使用 animation-direction 属性定义动画的播放方向。

语法：animation-direction:

<single-animation-direction> [, <single-animation-direction>]*

<single-animation-direction> = normal | reverse | alternate | alternate-reverse

animation-direction 属性取值

属性值	说明
normal	每次循环都向正方向播放（默认值）
reverse	每次循环都向反方向播放
alternate	播放次数是奇数时，动画向原方向播放；播放次数是偶数时，动画向反方向播放
alternate-reverse	动画先反运行再正方向运行，并持续交替运行

播放状态 animation-play-state 属性

在 CSS3 中，我们可以使用 `animation-play-state` 属性来定义动画的播放状态。

语法：`animation-play-state`: 取值;

说明：`animation-play-state` 属性取值只有 2 个：`running` 和 `paused`。

属性值	说明
<code>running</code>	播放动画（默认值）
<code>paused</code>	暂停动画

时间外属性 `animation-fill-mode` 属性

在 CSS3 中，我们可以使用 `animation-fill-mode` 属性定义在动画开始之前和动画结束之后发生的事情。

语法：`animation-fill-mode`:取值;

`animation-fill-mode` 属性取值

属性值	说明
<code>none</code>	动画完成最后一帧时会反转回到初始帧处（默认值）
<code>forwards</code>	动画结束之后继续应用最后的关键帧位置
<code>backwards</code>	会在向元素应用动画样式时迅速应用动画的初始帧
<code>both</code>	元素动画同时具有 <code>forwards</code> 和 <code>backwards</code> 效果

在默认情况之下，动画不会影响它的关键帧之外的属性，使用 `animation-fill-mode` 属性可以修改动画的默认行为。简单的说就是告诉动画在第一关键帧上等待动画开始，或者在动画结束时停在最后一个关键帧上而不回到动画的第一帧上。或者同时具有这两个效果。

例如：让动画停在最后一帧处。代码如下：

```
animation-fill-mode:forwards;
```

`animation`

语法：**`animation`**: <single-animation>[,<single-animation>]*

<single-animation> = <single-animation-name> || <time> ||
<single-animation-timing-function> || <time> || <single-animation-iteration-count>
|| <single-animation-direction> || <single-animation-fill-mode> ||
<single-animation-play-state>

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

- 如果提供多组属性值，以逗号进行分隔。
- 如果只提供一个<time>参数，则为 <' animation-duration '> 的值定义；
- 如果提供二个<time>参数，则第一个为 <' animation-duration '> 的值定义，第二个为 <' animation-delay '> 的值定义

案例：

```
animation:animations 2s ease-out forwards;
@keyframes animations{
    0%{transform:translate(0);opacity:0;}
    50%{transform:translate(30px);opacity:1;}
    70%{transform:translate(35px);opacity:1;}
    100%{transform:translate(60px);opacity:0;}
}
.....
```