Day12-CSS3新增属性变形及动画

大家都知道新增了很多属性，让我们的页面更加的丰富多彩，更加的绚丽，今天我们再给大家介绍三个属性，让大家眼前一亮，不用所谓的JS，也可以实现丰富多彩的动画了。

看今天的知识点

* + - 过渡
    - 动画
    - 变形

好了，我们赶紧看这些属性的用法吧：

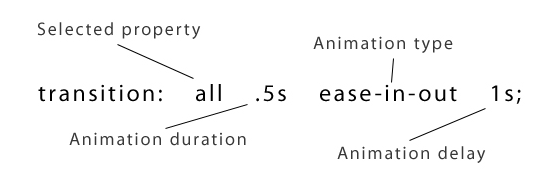
1. **transition（过渡）**

|  |  |
| --- | --- |
| **属性** | **描述** |
| [transition](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition.htm) | 复合属性。检索或设置对象变换时的过渡效果. |
| [transition-property](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition-property.htm) | 检索或设置对象中的参与过渡的属性. (默认值为all)  多个属性值用逗号分隔开表示可以同时为多个值设置过渡属性. |
| [transition-duration](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition-duration.htm) | 检索或设置对象过渡的持续时间. 默认值为0s，不能为负值。 |
| [transition-timing-function](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition-timing-function.htm) | 检索或设置对象中过渡的类型. 默认值为ease函数 |
| [transition-delay](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition-delay.htm) | 检索或设置对象延迟过渡的时间. 默认值为0s |
| 备注 | 注意：   * 如果只提供一个[<time>](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/values/time/time.htm)参数，则为 <' [transition-duration](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition-duration.htm) '> 的值定义；如果提供二个[<time>](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/values/time/time.htm)参数，则第一个为 <' [transition-duration](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition-duration.htm) '> 的值定义，第二个为 <' [transition-delay](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transition/transition-delay.htm) '> 的值定义 * 可以为同一元素的多个属性定义过渡效果   transition需要触发一 个事件(鼠标单击、鼠标放上去、获得焦点、被点击等)才会随时间改变其css属性。  一般的时候都使用他的复合属性。 |

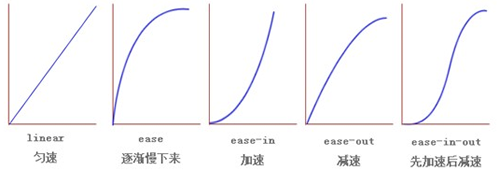
**语法：**

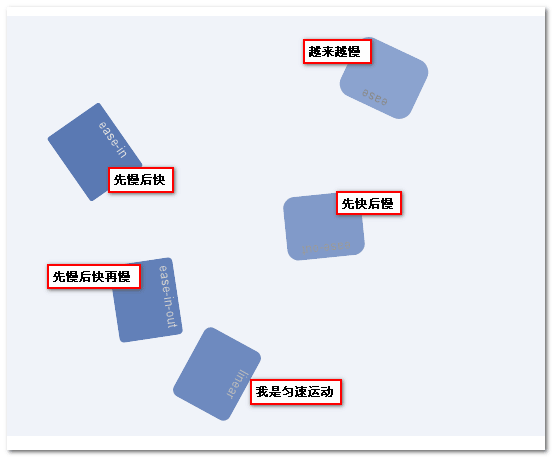
**transition的综合写法**

**选择器{transition:运动的样式 持续时间 运动形式 延迟时间;}**

****

**常用的过渡的类型：**

****

****

transition需要触发一 个事件(鼠标单击、鼠标放上去、获得焦点、被点击（:hover，:focus，:active，:checked）等)才会随时间改变其css属性。　　过渡transition的这四个子属性只有<transition-duration>是必需值且不能为0。 (设定值=延迟时间+持续时间)。

|  |
| --- |
| 不是所有的CSS样式值都可以过渡，只有具有中间值的属性才具备过渡效果。下面的属性可以进行过渡。  颜色: color background-color border-color outline-color  位置: backround-position left right top bottom  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 |

添加多个过渡属性：

|  |
| --- |
| #test1{  transition-property: width,background,opacity;  transition-duration: 2s,500ms;  transition-timing-function: linear,ease;  transition-delay: 200ms,0s;  }  /\*类似于\*/  #test2{  transition: width 2s linear 200ms,background 500ms ease 0s,opacity 2s linear 200ms;  } |

掌握过渡阶段：

1. 过渡开始时间=样式改变的时刻+过渡延迟时间；而过渡结束时间=过渡开始时间+过渡持续时间
2. 过渡起始值=过渡前的过渡属性值；而过渡结束值=过渡完成后的过渡属性值
3. 过渡分为两个阶段：前进(forward)和反向(reverse)。若前进阶段进行一段时间后进入反向阶段，则反向阶段的初始值是前进阶段结束时的瞬时值。
4. 以hover为例，若在元素非hover态时设置transition，相当于设置的反向状态。而前进和反向是一致的。而如果在元素hover态设置transition，则前进状态以hover态设置的为准，而反向状态以非hover态设置的为准。
5. 如果子元素和父元素过渡属性都一致。若触发子元素过渡时，父元素正在过渡，则将父元素过渡的中间态的值作为子元素过渡的初始值

【6】若过渡起始值或过渡开始值为auto，则浏览器不会自己计算成具体数字值，而是不发生过渡效果。所以要过渡某些属性，首先需要将其重置成具体数字值

案例代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>过渡属性</title>  <style>  div {  width: 100px;  height: 100px;  background: red;  transition: width 2s;  }  div:hover {  width: 300px;  }  </style>  </head>  <body>  <p><b>注意：</b>该实例无法在 Internet Explorer 9 及更早 IE 版本上工作。</p>  <div></div>  <p>鼠标移动到 div 元素上，查看过渡效果。</p>  </body>  </html> |

显示效果：





1. animation（动画）

|  |  |
| --- | --- |
| **属性** | **描述** |
| [animation](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation.htm) | 复合属性。检索或设置对象所应用的动画特效 |
| [animation-name](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-name.htm) | 检索或设置对象所应用的动画名称 |
| [animation-duration](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-duration.htm) | 检索或设置对象动画的持续时间 |
| [animation-timing-function](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-timing-function.htm) | 检索或设置对象动画的过渡类型 |
| [animation-delay](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-delay.htm) | 检索或设置对象动画延迟的时间 |
| [animation-iteration-count](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-iteration-count.htm) | 检索或设置对象动画的循环次数 |
| [animation-direction](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-direction.htm) | 检索或设置对象动画在循环中是否反向运动 |
| [animation-play-state](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-play-state.htm) | 检索或设置对象动画的状态 |
| [animation-fill-mode](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/animation/animation-fill-mode.htm) | 检索或设置对象动画时间之外的状态 |



transition需要触发一 个事件(鼠标单击、鼠标放上去、获得焦点、被点击等)才会随时间改变其css属性；而animation在不需要触发任何事件的情况下也可以显式的随着时间变化来 改变元素css的属性值，从而达到一种动画的效果。这样我们就可以直接在一个元素中调用animation的动画属性,基于这一点，css3的 animation就需要明确的动画属性值。

1. @keyframes规则一（使用from….to 函数定义关键帧）：

@keyframes myFrame{

from{width: 100px;}

to{width: 500px;}

}

1. @keyframes规则二（使用百分比定义关键帧）：

@keyframes myFrame{

0% {width: 50px; height: 100px;}

50% {width: 200px; height: 100px;}

100% {width: 200px; height: 200px;}

}

1. animation-iteration-count：动画执行的次数

默认值是 1

infinite：无限

1. animation-direction：动画播放的方向

normal:默认值

reverse：动画反向播放；

alternate：第一次正向播放，第二次反向播放

alternate-reverse:第一次反向播放，第二次正向播放

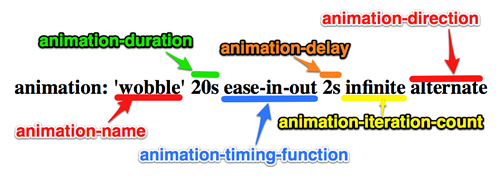
1. animation-fill-mode：动画播放前后的状态

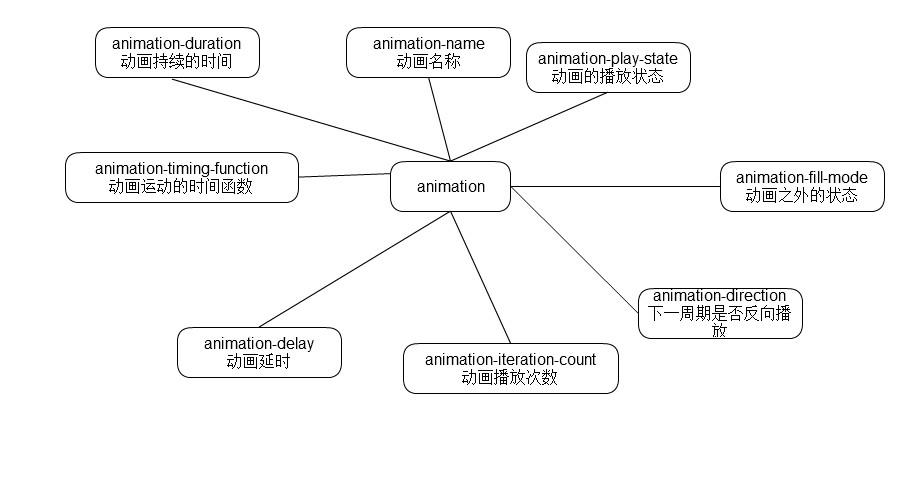
值：none|backwards|forwards|both

1. animation-play-state：控制动画运行或停止

值：running | paused

1. 语法





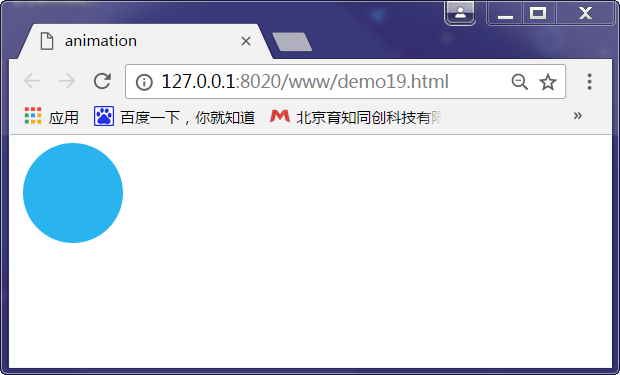
1. 注意事项：

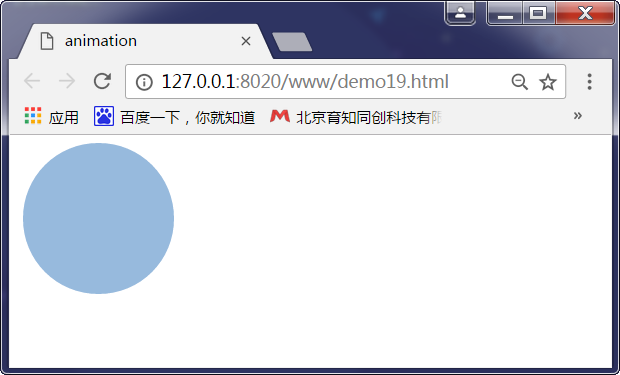
和animation相关的样式有8个，都可以写在一个animation属性中；至少要有animation-name和animation-duration两个样式animation-duration必须在animation-delay之前.

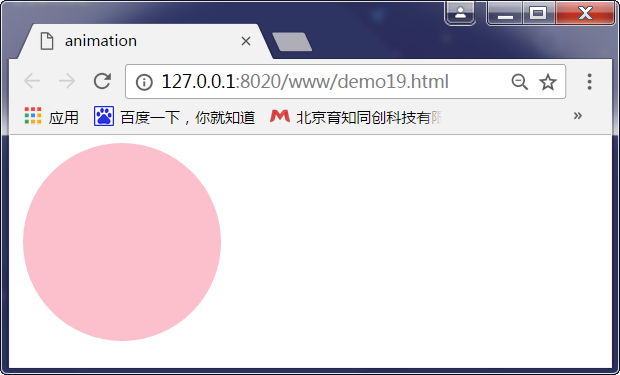
案例一（规则一）代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>animation</title>  <style type="text/css">  .ball {  width: 100px;  height: 100px;  border-radius: 100%;  background-color: #29B4F0;  margin: 5px;  animation: move1 2s ease 1s infinite;  }  @keyframes move1 {  from {  background: #29B4F0;  width:100px;  height: 100px;  }  to{  background: pink;  width:200px;  height: 200px;  }  </style>  </head>  <body>  <div class="ball"></div>  </body>  </html> |

显示效果：



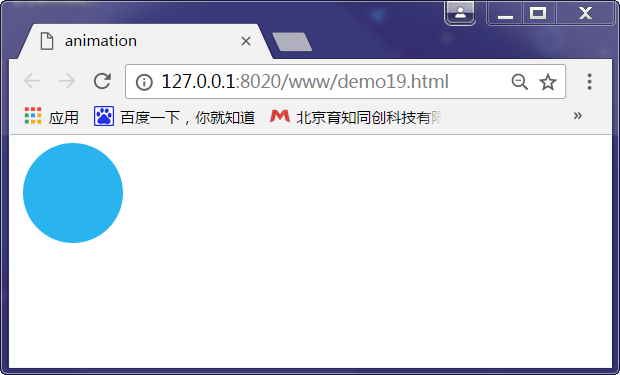


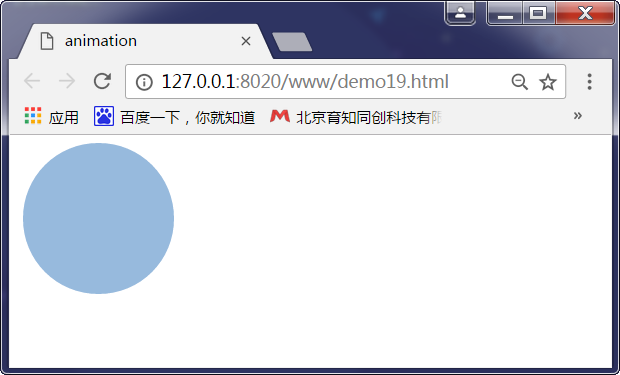


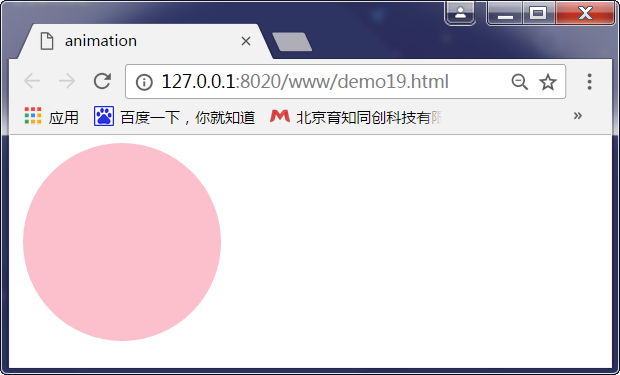
案例二（规则二）代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>animation</title>  <style type="text/css">  .ball {  width: 100px;  height: 100px;  border-radius: 100%;  background-color: #29B4F0;  margin: 5px;  animation: move1 2s ease 1s infinite;  }  @keyframes move1 {  0% {  background: #29B4F0;  width:100px;  height: 100px;  }  100% {  background: pink;  width:200px;  height: 200px;  }  </style>  </head>  <body>  <div class="ball"></div>  </body>  </html> |

显示效果：







Animation的重点和难点:

animation-delay  
这个属性是规定动画开始前等待几秒才开始。本来是很好理解的，但是当时就有个疑问：假如我的动画是连续执行好多次的情况下的话，是第一次执行前才会延迟还是每次执行前都会延迟呢？答案是：只有第一次会。

**animation-fill-mode**  
该属性有四个值  
none：默认值。不设置对象动画之外的状态  
forwards：设置对象状态为动画结束时的状态  
backwards：设置对象状态为动画开始时的状态  
both：设置对象状态为动画结束或开始的状态  
一开始比较纠结着三个到底有什么区别。网上也找了，发现网上说的有点错误。起码我试了一下forwards和both的效果是一模一样的。都是动画运行完了停在哪里就是哪里。至于backwards，就是总停在一开始的状态。

两段代码主要说明在forwards和both的情况下动画在设置了反向运行偶次数时，效果仍然一样。因为之前看到网上说的是forwards在偶次数反向是会停在关键帧的100%处，而不是0%处

**iteration-count**运行次数。  
这个本来没什么好说的，一目了然，但是无意中发现，有以下的特殊情况：  
**不能为负数  
可以为0**为0的时候，按1来算 **可以为小数** 小数的时候，不会取整。而是多少就运行多少，比如 .5 就运行到50%的位置。

1. **transform（**变形**）**

|  |  |
| --- | --- |
| **transform**属性 | |
| 2D Transform | |
| 属性值 | 描述 |
| [translate()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "translate)： | 指定对象的2D translation（2D平移）。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，则默认值为0 。 |
| [translatex()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "translatex)： | 指定对象X轴（水平方向）的平移 |
| [translatey()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "translatey)： | 指定对象Y轴（垂直方向）的平移 |
| [rotate()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "rotate)： | 指定对象的2D rotation（2D旋转），需先有 <' [transform-origin](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform-origin.htm) '> 属性的定义 |
| [scale()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "scale)： | 指定对象的2D scale（2D缩放）。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，则默认取第一个参数的值 |
| [scalex()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "scalex)： | 指定对象X轴的（水平方向）缩放 |
| [scaley()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "scaley)： | 指定对象Y轴的（垂直方向）缩放 |
| [skew()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "skew)： | 指定对象skew transformation（斜切扭曲）。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，则默认值为0 |
| [skewx()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "skewx)： | 指定对象X轴的（水平方向）扭曲 |
| [skewy()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "skewy)： | 指定对象Y轴的（垂直方向）扭曲 |
| [matrix()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "matrix)： | 以一个含六值的(a,b,c,d,e,f)变换矩阵的形式指定一个2D变换，相当于直接应用一个[a,b,c,d,e,f]变换矩阵 |
| 3D Transform | |
| [translate3d()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "translate3d)： | 指定对象的3D位移。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。 |
| [translatez()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "translatez)： | 指定对象Z轴的平移 |
| [rotate3d()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "rotate3d)： | 指定对象的3D旋转角度，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 |
| [rotatex()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "rotatex)： | 指定对象在x轴上的旋转角度 |
| [rotatey()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "rotatey)： | 指定对象在y轴上的旋转角度 |
| [rotatez()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "rotatey)： | 指定对象在z轴上的旋转角度 |
| [scale3d()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "scale3d)： | 指定对象的3D缩放。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 |
| [scalez()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "scalez)： | 指定对象的z轴缩放 |
| [matrix3d()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "matrix3d)： | 以一个4x4矩阵的形式指定一个3D变换 |

|  |  |
| --- | --- |
| **transform-origin**属性（**设置或检索对象以某个原点进行转换**） | |
| 属性值 | 描述 |
| [<percentage>](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/values/numeric/percentage.htm)： | 用百分比指定坐标值。可以为负值。 |
| [<length>](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/values/length/index.htm)： | 用长度值指定坐标值。可以为负值。 |
| left： | 指定原点的横坐标为left |
| center①： | 指定原点的横坐标为center |
| right： | 指定原点的横坐标为right |
| top： | 指定原点的纵坐标为top |
| center②： | 指定原点的纵坐标为center |
| bottom： | 指定原点的纵坐标为bottom |
| 说明 | * 该属性提供2个参数值。 * 如果提供两个，第一个用于横坐标，第二个用于纵坐标。 * 如果只提供一个，该值将用于横坐标；纵坐标将默认为50%。 |

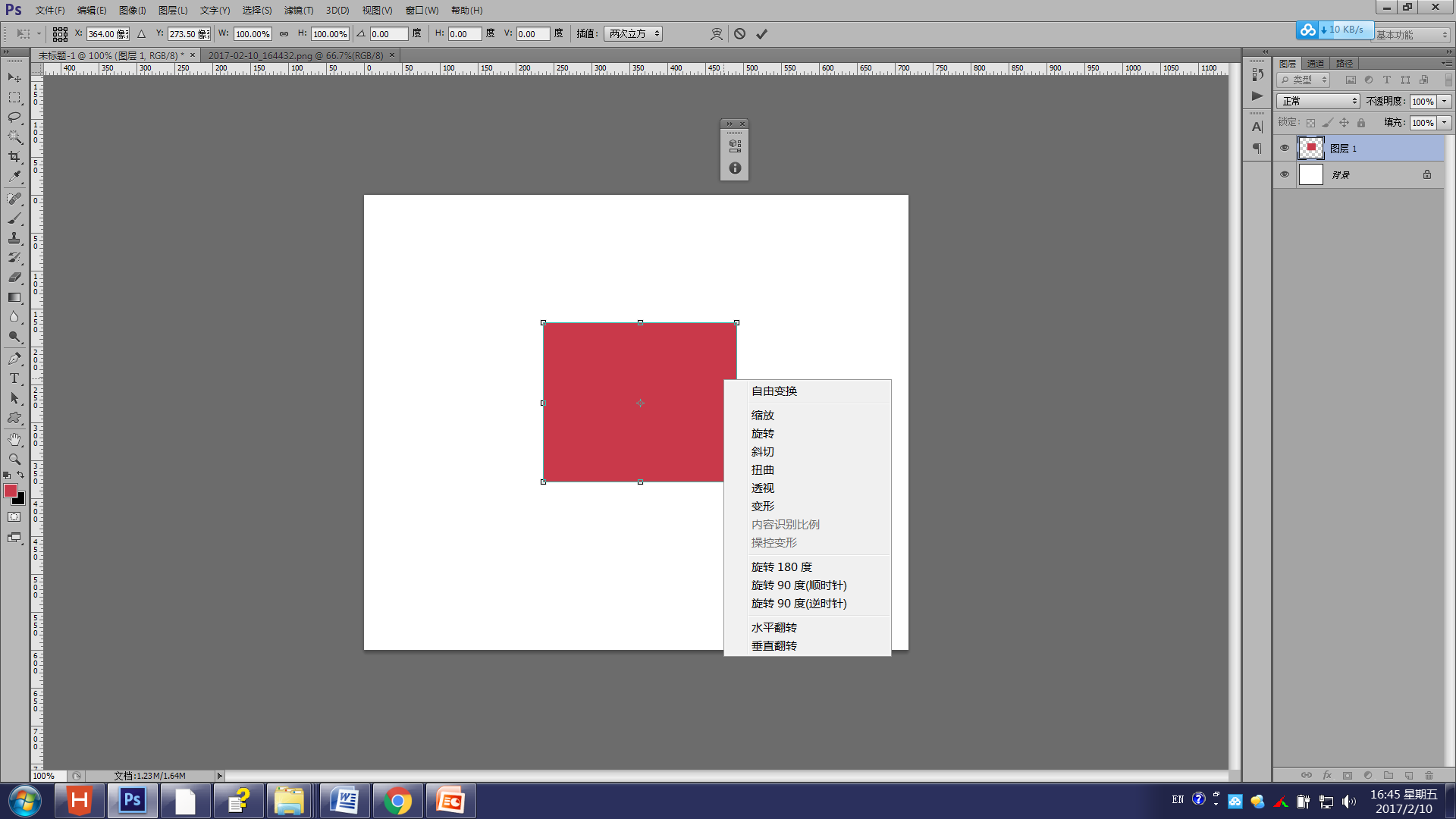
|  |  |
| --- | --- |
| [perspective()](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/properties/transform/transform.htm" \l "perspective)属性(指定透视距离) | |
| 属性值 | 描述 |
| none： | 不指定透视 |
| [<length>](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/values/length/index.htm) | 指定观察者距离「z=0」平面的距离，为元素及其内容应用透视变换。不允许负值 |
| perspective（**指定观察者与「z=0」平面的距离，使具有三维位置变换的元素产生透视效果。「z>0」的三维元素比正常大，而「z<0」时则比正常小，大小程度由该属性的值决定。**） | |

|  |  |
| --- | --- |
| perspective-origin属性(**指定透视点的位置**) | |
| 属性值 | 描述 |
| [<percentage>](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/values/numeric/percentage.htm)： | 用百分比指定透视点坐标值，相对于元素宽度。可以为负值。 |
| [<length>](mk:@MSITStore:E:\\教学资料\\0页面结构与表现\\第1节课\\API\\css3.chm::/values/length/index.htm)： | 用长度值指定透视点坐标值。可以为负值。 |
| left： | 指定透视点的横坐标为left |
| center①： | 指定透视点的横坐标为center |
| right： | 指定透视点的横坐标为right. |
| top： | 指定透视点的纵坐标为top |
| center②： | 指定透视点的纵坐标为center |
| bottom： | 指定透视点的纵坐标为bottom |
| 说明 | * 该属性提供2个参数值。 * 如果提供两个，第一个用于横坐标，第二个用于纵坐标。 * 如果只提供一个，该值将用于横坐标；纵坐标将默认为center。 |

|  |  |
| --- | --- |
| transform-style属性 | |
| 属性值 | 描述 |
| flat： | 指定子元素位于此元素所在平面内。 |
| preserve-3d： | 指定子元素定位在三维空间内。 |
| transform-style（**指定某元素的子元素是（看起来）位于三维空间内，还是在该元素所在的平面内被扁平化。**） | |

|  |  |
| --- | --- |
| backface-visibility:指定元素背面面向用户时是否可见 | |
| 属性值 | 描述 |
| visible | 指定元素背面可见，允许显示正面的镜像。 |
| hidden | 指定元素背面不可见 |
| 说明 | 决定一个元素背面面向用户时是否可见，需要直接在该元素上定义 <' backface-visibility '> 属性，而不能在其父元素上，因为该属性默认为不可继承 |

个人觉得这些平移、旋转、扭曲、缩放都来源于photoshop，只不过我现完全可以使用代码来实现了。



中心点

1. 2D变形

1.1 translate（）平移

**translate ：指定对象的2D translation（2D平移）。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，则默认值为0，例如translate(10px,10px)**

**-**

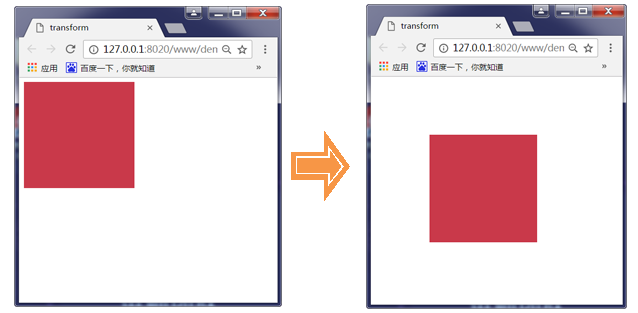
**+**

**+**

代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>transform</title>  <style type="text/css">  .box{  width: 200px;  height: 200px;  background: #c9394a;  transform: translate(100px,100px);  }  </style>  </head>  <body>  <div class="box"></div>  </body>  </html> |

显示效果



* 1. Rotate（）旋转

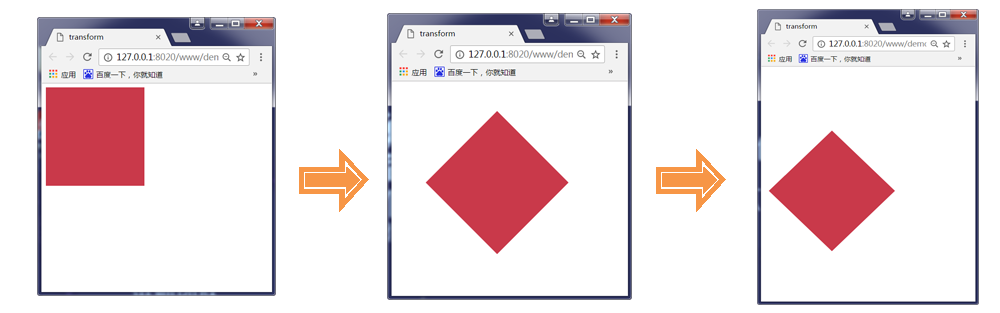
**Rotate：指定对象的2D rotation（2D旋转），需先有transform-origin属性的定义。**

**-**

**+**

代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>transform</title>  <style type="text/css">  .box{  margin: 150px;  width: 200px;  height: 200px;  background: #c9394a;  transform: rotate(45deg);  transform-origin: left top;  }  </style>  </head>  <body>  <div class="box"></div>  </body>  </html> |

显示效果:

* 1. Scale（）缩放

**指定对象的2D scale（2D缩放）。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，则默认取第一个参数的值 。例如： scale (0.5,0.5)**

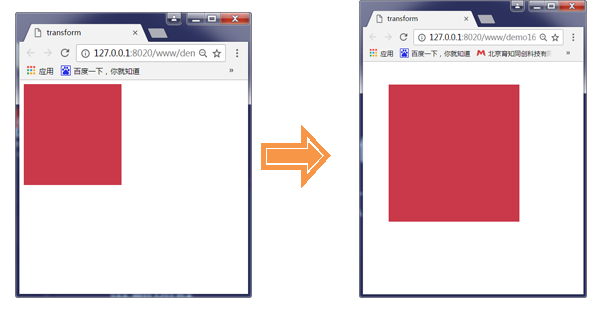
**+**

**-**

代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>transform</title>  <style type="text/css">  .box{  margin: 100px;  width: 200px;  height: 200px;  background: #c9394a;  transform: scale(1.5);  }  </style>  </head>  <body>  <div class="box"></div>  </body>  </html> |

显示效果



* 1. Skew（）扭曲

**指定对象skew transformation（斜切扭曲）。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，则默认值为0 。例如： skew (50deg,20deg)**

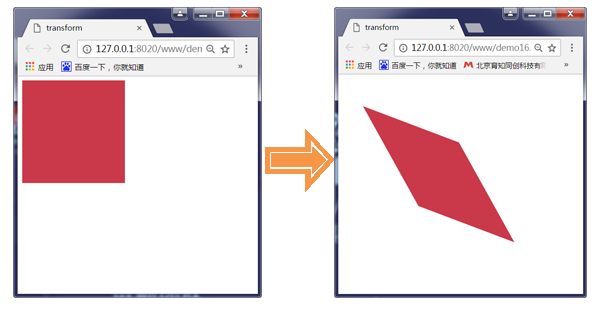
**-**

**+**

代码：

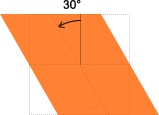
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>transform</title>  <style type="text/css">  .box{  margin: 100px;  width: 200px;  height: 200px;  background: #c9394a;  transform: skew(30deg,20deg);  }  </style>  </head>  <body>  <div class="box"></div>  </body>  </html> |

显示效果：

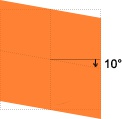


首先需要说明的是skew的默认原点transform-origin是这个物件的中心点。

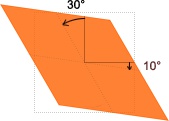
skewX(30deg) 如下图：

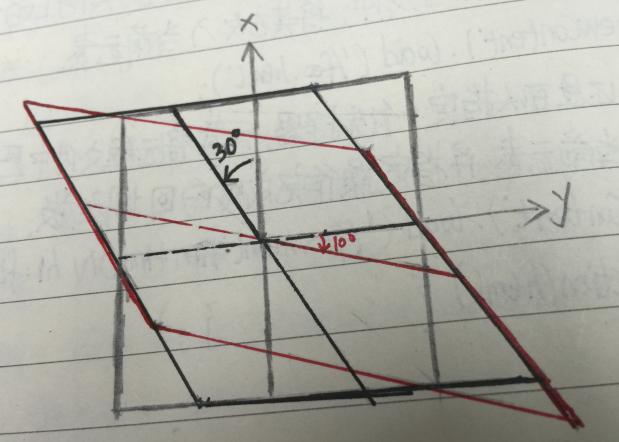


skewY(10deg) 如下图：



skew(30deg, 10deg) 如下图：

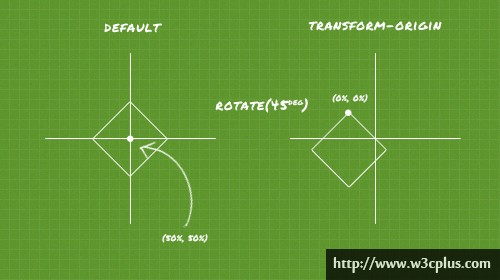




* 1. transform-origin原点

​ 默认的中心点是原点，可以根据实现需要更改他的原点,以旋转为例.我们没有使用transform-origin改变元素原点位置的情况下，CSS变形进行的旋转、移位、缩放等操作都是以元素自己中心（变形原点）位置进行变形的。但很多时候需要在不同的位置对元素进行变形操作，我们就可以使用transform-origin来对元素进行原点位置改变，使元素原点不在元素的中心位置，以达到需要的原点位置。

如果我们把元素变换原点（transform-origin）设置0（x） 0（y），这个时候元素的变换原点转换到元素的左顶角处，如下图所示。



正如上图所示，改变transform-origin属性的X轴和Ｙ轴的值就可以重置元素变形原点位置，其基本语法如下所示:

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

上面的语法让人看得发晕，其实可以将语法拆分成：

/\*只设置一个值的语法\*/

transform-origin: x-offset

transform-origin:offset-keyword

/\*设置两个值的语法\*/

transform-origin：x-offset y-offset

transform-origin：y-offset x-offset-keyword

transform-origin：x-offset-keyword y-offset

transform-origin：x-offset-keyword y-offset-keyword

transform-origin：y-offset-keyword x-offset-keyword

/\*设置三个值的语法\*/

transform-origin：x-offset y-offset z-offset

transform-origin：y-offset x-offset-keyword z-offset

transform-origin：x-offset-keyword y-offset z-offset

transform-origin：x-offset-keyword y-offset-keyword z-offset

transform-origin：y-offset-keyword x-offset-keyword z-offset

transform-origin`属性值可以是百分比、em、px等具体的值，也可以是top、right、bottom、left和center这样的关键词。

2D的变形中的transform-origin属性可以是一个参数值，也可以是两个参数值。如果是两个参数值时，第一值设置水平方向X轴的位置，第二个值是用来设置垂直方向Ｙ轴的位置。3D的变形中的transform-origin属性还包括了Ｚ轴的第三个值。其各个值的取值简单说明如下：

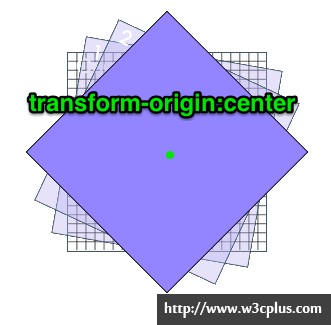
* x-offset：用来设置transform-origin水平方向Ｘ轴的偏移量，可以使用和值，同时也可以是正值（从中心点沿水平方向Ｘ轴向右偏移量），也可以是负值（从中心点沿水平方向Ｘ轴向左偏移量）。
* offset-keyword：是top、right、bottom、left或center中的一个关键词，可以用来设置transform-origin的偏移量。
* y-offset：用来设置transform-origin属性在垂直方向Ｙ轴的偏移量，可以使用和值，同时可以是正值（从中心点沿垂直方向Ｙ轴向下的偏移量），也可以是负值（从中心点沿垂直方向Ｙ轴向上的偏移量）。
* x-offset-keyword：是left、right或center中的一个关键词，可以用来设置transform-origin属性值在水平Ｘ轴的偏移量。
* y-offset-keyword：是top、bottom或center中的一个关键词，可以用来设置transform-origin属性值在垂直方向Ｙ轴的偏移量。
* z-offset：用来设置3D变形中transform-origin远离用户眼睛视点的距离，默认值z=0，其取值可以，不过在这里将无效。

看上去transform-origin取值与background-position取值类似。为了方便记忆，可以把关键词和百分比值对比起来记：

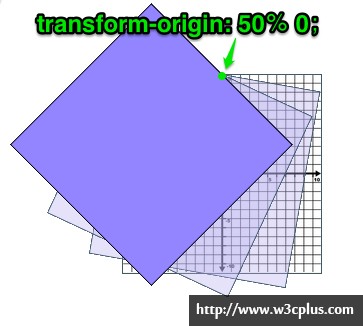
* top = top center = center top = 50% 0
* right = right center = center right = 100%或(100% 50%)
* bottom = bottom center = center bottom = 50% 100%
* left = left center = center left = 0或(0 50%)
* center = center center = 50%或（50% 50%）
* top left = left top = 0 0
* right top = top right = 100% 0
* bottom right = right bottom = 100% 100%
* bottom left = left bottom = 0 100%

为了让大家能一眼看明白，下面截图以transform中的旋转rotate()为例，并transform-origin取值不一样时的效果：

transform-origin取值为center（或center center或50% 或50% 50%）：



transform-origin取值为top（或top center或center top或50％ 0）



注：CSS3变形中旋转、缩放、倾斜都可以通过transform-origin属性重置元素的原点，但其中的位移translate()始终以元素中心点进行位移。

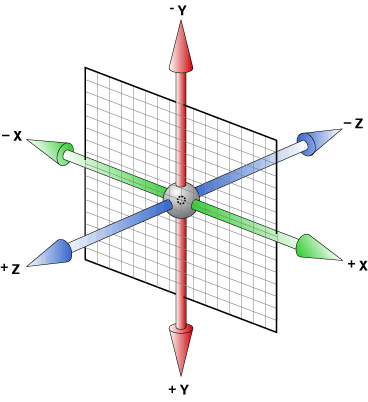
代码：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8">  <title>疯狂的按键</title>  <style>  /\*基本样式\*/  body, ul {  margin: 0;  padding: 0;  }  ul, li {  list-style: none;  }  h1 {  background: #eee;  width: 100%;  height: 50px;  line-height: 50px;  text-align: center;  font-size: 20px;  margin: 60px 0;  }  h1:nth-child(1) {  margin-top: 20px;  }  ul {  width: 77%;  margin: 0 auto;  height: 150px;  }  li {  width: auto;  height: auto;  float: left;  margin-right: 50px;  }  ul li:last-child {  margin-right: 0;  }  /\*设置过渡效果\*/  .share1 img,.share2 img{  transition: all 0.8s ease-in-out;  }  /\*设置变形\*/  .share1 img:hover {  transform: rotate(360deg) scale(1.5);  }  .share2 img:hover {  transform: rotate(-360deg) scale(0.7);  }  </style>  </head>  <body>  <h1>顺时针旋转360°放大1.2倍</h1>  <ul class="share1">  <li><a href="#"><img src="images/share1.png" alt=""></a></li>  <li><a href="#"><img src="images/share2.png" alt=""></a></li>  <li><a href="#"><img src="images/share3.png" alt=""></a></li>  <li><a href="#"><img src="images/share4.png" alt=""></a></li>  <li><a href="#"><img src="images/share5.png" alt=""></a></li>  <li><a href="#"><img src="images/share6.png" alt=""></a></li>  </ul>  <h1>逆时针旋转360°缩小0.7倍</h1>  <ul class="share2">  <li><a href="#"><img src="images/share7.png" alt=""></a></li>  <li><a href="#"><img src="images/share8.png" alt=""></a></li>  <li><a href="#"><img src="images/share9.png" alt=""></a></li>  <li><a href="#"><img src="images/share10.png" alt=""></a></li>  <li><a href="#"><img src="images/share11.png" alt=""></a></li>  <li><a href="#"><img src="images/share12.png" alt=""></a></li>  </ul>  </body>  </html> |



2. 3D变形

#### 2.1 3D的概念



认识3D的突破口：3D transform有下面这三个方法:rotateX, rotateY, rotateZ

* rotateX( angle )
* rotateY( angle )
* rotateZ( angle )

rotate旋转的意思，rotateX旋转X轴，rotateY旋转Y轴，rotateZ旋转Z轴…

什么X轴/Y轴/Z轴，这几个词从我嘴里一出来，别说你们，我自己都晕了~~

邹凯的体操单杠运动是rotateX；



蔡依林姐姐的钢管舞是rotateY；



旋转飞刀的特技表演是rotateZ。



#### 2.2 perspective属性

如果不设置透视，元素的3D变形效果将无法实现。应用在非inline元素。

​ perspective属性的存在与否决定了你所看到的是2次元的还是3次元的，也就是是2D transform还是3D transform. 这不难理解，没有透视，不成3D。

要深入了解透视，需要了解观察者、被透视元素和变形元素这几个概念。

首先变形元素，顾名思义，就是进行transform3D变形的元素，主要进行transform、transform-origin、backface-visibility等属性的设置。

　　观察者是浏览器模拟出来的用来观察被透视元素的一个没有尺寸的点，观察者发出视线，类似于一个点光源发出光线。

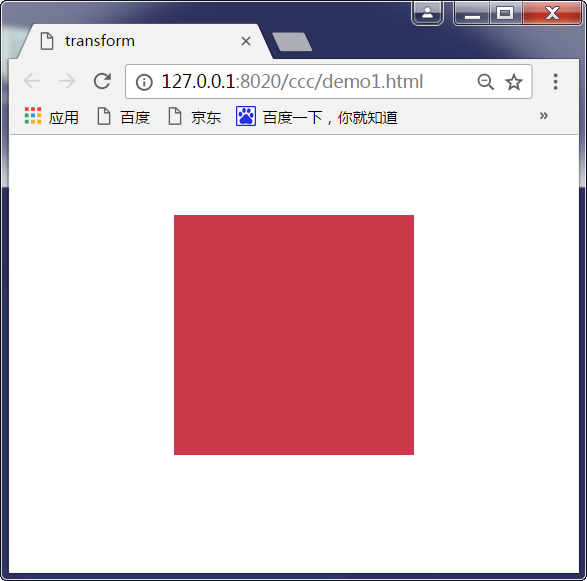
　　被透视元素也就是被观察者观察的元素，根据属性设置的不同，它有可能是变形元素本身，也可能是它的父级或祖先其元素(后面会详细介绍)，主要进行perspective、perspective-origin等属性的设置。

**透视距离：**我们可以简单的理解为视距，用来设置用户和元素3D空间Z平面之间的距离。而其效应由他的值来决定，值越小，用户与3D空间Z平面距离越近，视觉效果更令人印象深刻；反之，值越大，用户与3D空间Z平面距离越远，视觉效果就很小。

代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>transform</title>  <style type="text/css">  body{  perspective: 600px;  }  .box{  width: 200px;  height: 200px;  background: #c9394a;  margin: 100px auto;  transform: translateZ(500px);  }  </style>  </head>  <body>  <div class="box"></div>  </body>  </html> |

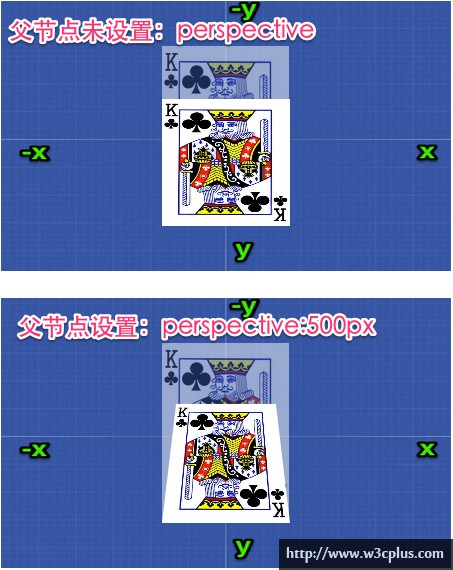
显示效果:



代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  div {  width: 500px;  height: 300px;  margin: 30px auto;  position: relative;  background: url(img/bg.jpg) no-repeat center center;  background-size: 100% 100%;  }    div img {  position: absolute;  top: 50%;  left: 50%;  margin-left: -71px;  margin-top: -100px;  transform-origin: bottom;  }    div img:nth-child(1) {  opacity: .5;  z-index: 1;  }    div img:nth-child(2) {  z-index: 2;  transform: rotateX(45deg);  }    div:nth-of-type(2) {  perspective: 500px;  }  </style>  </head>  <body>  <div>  <img src="img/timg (1).jpg" alt="" width="142" height="200" />  <img src="img/timg (1).jpg" alt="" width="142" height="200" />  </div>  <div>  <img src="img/timg (1).jpg" alt="" width="142" height="200" />  <img src="img/timg (1).jpg" alt="" width="142" height="200" />  </div>  </body>  </html> |

显示效果：



再来看看perspective的使用语法：

perspective：none | <length>

perspective属性包括两个属性：none和具有单位的长度值。其中perspective属性的默认值为none，表示无限的角度来看3D物体，但看上去是平的。另一个值接受一个长度单位大于0的值。而且其单位不能为百分比值。值越大，角度出现的越远，从而创建一个相当低的强度和非常小的3D空间变化。反之，此值越小，角度出现的越近，从而创建一个高强度的角度和一个大型3D变化。

注:

* 一般的时候perspective加在根元素上，一般的时候加body上。也可以加在子元素的transform上。
* 透视perspective不可为0和负数透视perspective不可取百分比。

【1】一般地，物体离得越远，显得越小。反映在perspective属性上，就是该属性值越大，元素的3d效果越不明显。(就像离一个人很近，甚至可以看到他的毛孔；如果离一个人很远，可能只看到一个轮廓)

【2】设置透视perspective属性的元素就是被透视元素。一般地，该属性只能设置在变形元素的父级或祖先级。因为浏览器会为其子级的变形产生透视效果，但并不会为其自身产生透视效果。

#### 2.3 perspective-origin

透视原点perspective-origin是指观察者的位置.应用在非inline元素。

perspective-origin属性是3D变形中另一个重要属性，主要用来决定perspective属性的源点角度。它实际上设置了X轴和Y轴位置，在该位置观看者好像在观看该元素的子元素。

perspective-origin属性的使用语法也很简单：

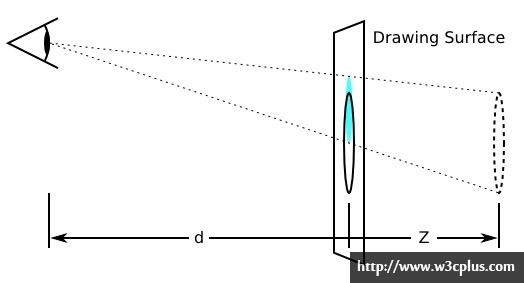
perspective-origin：[<percentage> | <length> | left | center | right | top | bottom] | [[<percentage> | <length> | left | center | right] && [<percentage> | <length> | top | center | bottom]]

该属性的默认值为“50% 50%”(也就是center)，其也可以设置为一个值，也可以设置为两个长度值：

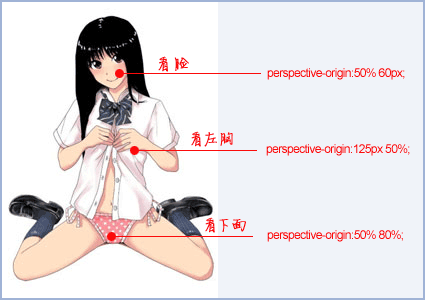
* 第一个长度值指定相对于元素的包含框的X轴上的位置。它可以是长度值（以受支持的长度单位表示）、百分比或以下三个关键词之一：left（表示在包含框的X轴方向长度的0%），center（表示中间点）,或right（表示长度的100%）。
* 第二个长度值指定相对于元素的包含框的Y轴上的位置。它可以是长度值、百分比或以下三个关键词之一：top（表示在包含框的Y轴方向长度的0%），center（表示中间点），或bottom（表示长度的100%）。

注意，为了指转换子元素变形的深度，perspective-origin属性必须定义父元素上。通常perspective-origin属性本身不做任何事情，它必须被定义在设置了perspective属性的元素上。换句话说，perspective-origin属性需要与perspective属性结合起来使用，以便将视点移至元素的中心以外位置。

往往我们看一样东西不可能一直都在中心位置看，想换个角度，换个位置一看究竟，这个时候就离不开perspective-origin这个属性，下面来自于W3C官网的图可以简单阐述这一观点。



这个属性超级好理解，表示你那双色迷迷的眼睛看的位置。默认就是所看舞台或元素的中心。有时候，我们对中心的位置是不感兴趣的，希望视线放在其他一些地方.



#### 2.4 transform-style属性

transform-style属性是3D空间一个重要属性，指定嵌套元素如何在3D空间中呈现。他主要有两个属性值：flat和preserve-3d。

transform-style属性的使用语法非常简单：

transform-style: flat | preserve-3d

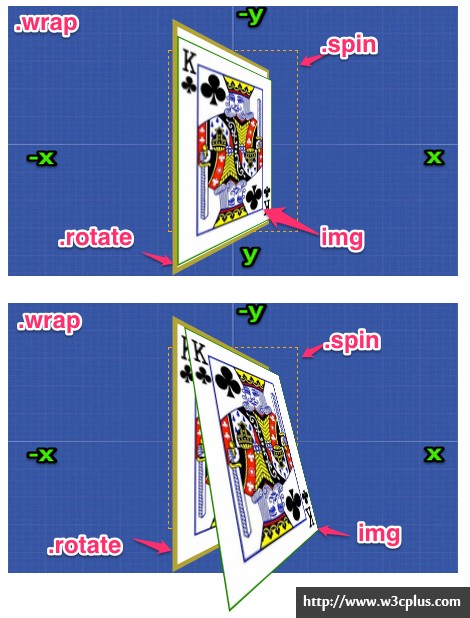
其中flat值为默认值，表示所有子元素在2D平面呈现。preserve-3d表示所有子元素在3D空间中呈现。

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。

transform-style属性需要设置在父元素中，并且高于任何嵌套的变形元素。最后，我们运用一个翻转的例子，来加深一下对transform-style属性的印象：

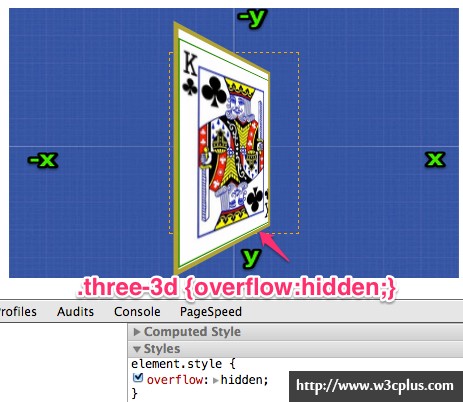
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  .wrap {  width: 500px;  height: 300px;  margin: 30px auto;  position: relative;  background: url(images/bg-grid.jpg) no-repeat center center;  background-size: 100% 100%;  }  /\*设置动画\*/    @keyframes spin {  0% {  transform: rotateY(0deg)  }  100% {  transform: rotateY(360deg)  }  }  .spin {  width: 142px;  height: 200px;  position: absolute;  top: 50%;  left: 50%;  margin-left: -72px;  margin-top: -101px;  border: 1px dashed orange;  cursor: pointer;  transform-style: preserve-3d;  }  /\*调用动画\*/  .spin:hover {  animation: spin 5s linear infinite;  }  .rotate {  background: url(images/cardKingClub.png) no-repeat center;  background-size: 100% 100%;  border: 5px solid hsla(50, 50%, 50%, .9);  transform: perspective(200px) rotateY(45deg);  }  .rotate img {  border: 1px solid green;  transform: rotateX(15deg) translateZ(10px);  transform-origin: 0 0 40px;  }  /\*改变transform-style的默认值\*/  .three-d {  transform-style: preserve-3d;  }  </style>  </head>  <body>  <div class="wrap">  <div class="spin">  <div class="rotate">  <img src="img/timg (1).jpg" alt="" width="142" height="200" />  </div>  </div>  </div>  <div class="wrap">  <div class="spin">  <div class="rotate three-d">  <img src="img/timg (1).jpg" alt="" width="142" height="200" />  </div>  </div>  </div>  </body>  </html> |

显示效果



正如您所看到的，当元素设置.rotate设置了flat值时，其子元素img不会根据translateZ()值摊开，而在同一平面旋转，如上图上部分所示；当元素.rotate设置了preserve-3d值时，其子元素img会根据translateZ()值摊开，不再会堆叠在一起，如上图下部分所示。

有一点需要特别提醒大家，如果你的元素设置了transform-style值为preserve-3d，就不能为了防止子元素溢出容器而设置overflow值为hidden，如果设置了overflow:hidden同样可以迫死子元素出现在同一平面（和元素设置了transform-style为flat一样的效果），如下图所示：



注:transform-style加在对应的舞台元素上，给父元素设置舞台。

#### 2.4 backface-visibility属性

backface-visibility属性决定元素旋转背面是否可见。对于未旋转的元素，该元素的正面面向观看者。当其Y轴旋转约180度时会导致元素的背面面对观众。

backface-visibility属性使用语法很简单：

backface-visibility: visible | hidden

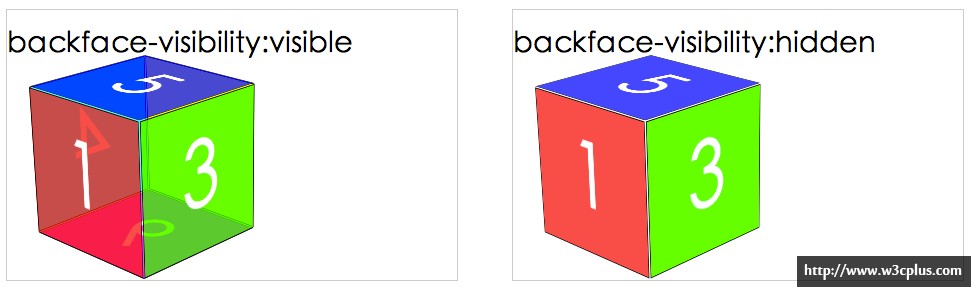
该属性被设置为以下两个关键词之一：

* visible：为backface-visibility的默认值，表示反面可见
* hidden：表示反面不可见
* 一个元素的可见性与“backface-visibility:hidden”决定如下：
* 元素在3D环境下渲染上下文，将根据3D变形矩阵来计算，反之元素不在3D环境下渲染上下文，将根据2D变形矩阵来计算。
* 如果组件的矩阵在第3行、3列是负值，那么元素反面是隐藏，反之是可见的。

简单点来说，backface-visibility属性可用于隐藏内容的背面。默认情况下，背面可见，这意味着即使在翻转后，旋转的内容仍然可见。但当backface-visibility设置为hidden时，旋转后内容将隐藏，因为旋转后正面将不再可见。该功能可帮助您模拟多面的对象，例如下例中使用的纸牌。通过将backface-visibility设置为hidden，您可以轻松确保只有正面可见。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  .container {  width: 500px;  height: 300px;  float: left;  position: relative;  margin: 30px;  border: 1px solid #CCC;  perspective: 1200px;  }  .cube {  width: 100%;  height: 100%;  position: absolute;  animation: spinCube 8s infinite ease-in-out;  transform-style: preserve-3d;  transform: translateZ( -100px);  }  @keyframes spinCube {  0% {  transform: translateZ( -100px) rotateX( 0deg) rotateY( 0deg);  }  100% {  transform: translateZ( -100px) rotateX( 360deg) rotateY( 360deg);  }  }  .side {  display: block;  position: absolute;  width: 196px;  height: 196px;  border: 2px solid black;  line-height: 196px;  font-size: 120px;  font-weight: bold;  color: white;  text-align: center;  }  .cube.backface-visibility-visible .side {  backface-visibility: visible;  }    .cube.backface-visibility-hidden .side {  backface-visibility: hidden;  }  .cube .front {  background: hsla( 0, 100%, 50%, 0.7);  }  .cube .back {  background: hsla( 60, 100%, 50%, 0.7);  }  .cube .right {  background: hsla( 120, 100%, 50%, 0.7);  }  .cube .left {  background: hsla( 180, 100%, 50%, 0.7);  }  .cube .top {  background: hsla( 240, 100%, 50%, 0.7);  }  .cube .bottom {  background: hsla( 300, 100%, 50%, 0.7);  }  .cube .front {  transform: translateZ( 100px);  }  .cube .back {  transform: rotateX( -180deg) translateZ( 100px);  }  .cube .right {  transform: rotateY( 90deg) translateZ( 100px);  }  .cube .left {  transform: rotateY( -90deg) translateZ( 100px);  }  .cube .top {  transform: rotateX( 90deg) translateZ( 100px);  }  .cube .bottom {  transform: rotateX( -90deg) translateZ( 100px);  }  </style>  </head>  <body>  <div class="container">  <h1>backface-visibility:visible</h1>  <div class="cube backface-visibility-visible">  <div class="side front">1</div>  <div class="side back">2</div>  <div class="side right">3</div>  <div class="side left">4</div>  <div class="side top">5</div>  <div class="side bottom">6</div>  </div>  </div>  <div class="container">  <h1>backface-visibility:hidden</h1>  <div class="cube backface-visibility-hidden">  <div class="side front">1</div>  <div class="side back">2</div>  <div class="side right">3</div>  <div class="side left">4</div>  <div class="side top">5</div>  <div class="side bottom">6</div>  </div>  </div>  </body>  </html> |

显示效果



#### 2.5 3D平移

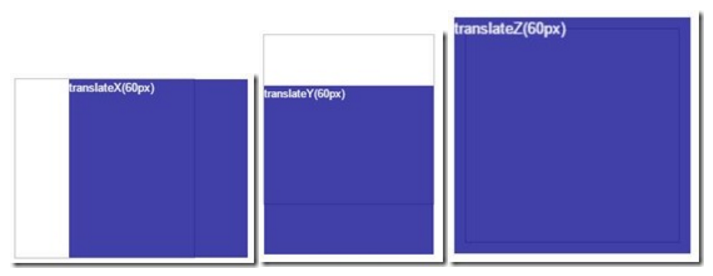
在CSS3中3D位移主要包括两种函数translateZ()和translate3d()。translate3d()函数使一个元素在三维空间移动。这种变形的特点是，使用三维向量的坐标定义元素在每个方向移动多少。

随着px的增加，直观效果上：

X：从左向右移动

Y：从上向下移动

Z：以方框中心为原点，变大



从上图的效果可以看出，当z轴值越大时，元素也离观看者更近，从视觉上元素就变得更大；反之其值越小时，元素也离观看者更远，从视觉上元素就变得更小。

3d位移函数主要包括traslateZ()和translate3d()

translate3d(x,y,z)

　　[注意]其中，x和y可以是长度值，也可以是百分比，百分比是相对于其本身元素水平方向的宽度和垂直方向的高度和；z只能设置长度值

traslateZ(z)相当于translate3d(0,0,z)。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  .stage {  width: 300px;  height: 300px;  float: left;  margin: 15px;  position: relative;  background: url(img/bg.jpg) repeat center center;  perspective: 1200px;  }    .container {  position: absolute;  top: 50%;  left: 50%;  transform-style: preserve-3d;  }    .container img {  position: absolute;  margin-left: -35px;  margin-top: -50px;  }    .container img:nth-child(1) {  z-index: 1;  opacity: .6;  }    .s1 img:nth-child(2) {  z-index: 2;  transform: translate3d(30px, 30px, 200px);  }    .s2 img:nth-child(2) {  z-index: 2;  transform: translate3d(30px, 30px, -200px);  }  </style>  </head>  <body>  <div class="stage s1">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  </div>  </div>  <div class="stage s2">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  </div>  </div>  </body>  </html> |

#### 2013112812244616.jpg

#### 2.6 3D旋转

3d旋转函数主要包括rotateX()、rotateY()、rotateZ()、rotate3d()

rotate3d(x,y,z,Ndeg)

x、y、z分别用来描述围绕x、y、z轴旋转的矢量值。最终变形元素沿着由(0,0,0)和(x,y,z)这两个点构成的直线为轴，进行旋转。当N为正数时，元素进行顺时针旋转；当N为负数时，元素进行逆时针旋转.

* x：是一个0到１之间的数值，主要用来描述元素围绕X轴旋转的矢量值；
* y：是一个０到１之间的数值，主要用来描述元素围绕Y轴旋转的矢量值；
* z：是一个０到１之间的数值，主要用来描述元素围绕Z轴旋转的矢量值；
* a：是一个角度值，主要用来指定元素在3D空间旋转的角度，如果其值为正值，元素顺时针旋转，反之元素逆时针旋转。
* rotateX(Ndeg)相当于rotate3d(1,0,0,Ndeg)
* rotateY(Ndeg)相当于rotate3d(0,1,0,Ndeg)
* rotateZ(Ndeg)相当于rotate3d(0,0,1,Ndeg)

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  .stage {  width: 300px;  height: 300px;  float: left;  margin: 15px;  position: relative;  background: url(img/bg.jpg) repeat center center;  perspective: 1200px;  }    .container {  position: absolute;  top: 50%;  left: 50%;  transform-style: preserve-3d;  }    .container img {  position: absolute;  margin-left: -70px;  margin-top: -100px;  }    .container img:nth-child(1) {  z-index: 1;  opacity: .6;  }    .s1 img:nth-child(2) {  z-index: 2;  transform: rotateX(45deg);  }    .s2 img:nth-child(2) {  z-index: 2;  transform: rotateY(45deg);  }    .s3 img:nth-child(2) {  z-index: 2;  transform: rotateZ(45deg);  }    .s4 img:nth-child(2) {  z-index: 2;  transform: rotate3d(.6, 1, .6, 45deg);  }  </style>  </head>  <body>  <div class="stage s1">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  </div>  </div>  <div class="stage s2">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  </div>  </div>  <div class="stage s3">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  </div>  </div>  <div class="stage s4">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  <img src="img/timg (1).jpg" alt="" width="140" height="200" />  </div>  </div>  </body>  </html> |

#### 2013112812247472.jpg

#### 2.7 3D缩放

#### 3d缩放函数主要包括scaleZ()和scale3d()

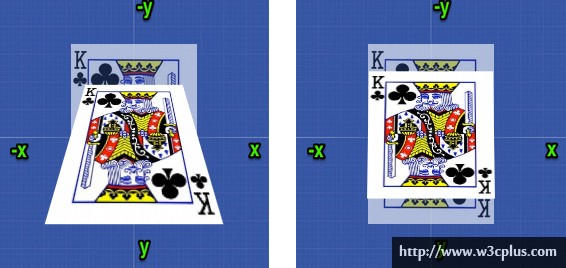
scale3d(x,y,z)

　　默认值为scale3d(1,1,1)，当参数为负值时，先翻转再缩放

scaleZ(z)相当于scale3d(1,1,z)

[注意]scaleZ()和scale3d()单独使用时没有任何效果，需要配合其他的变形函数一起使用才会有效果。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  .stage {  width: 300px;  height: 300px;  float: left;  margin: 15px;  position: relative;  background: url(img/bg.jpg) repeat center center;  perspective: 1200px;  }    .container {  position: absolute;  top: 50%;  left: 50%;  transform-style: preserve-3d;  }    .container img {  position: absolute;  margin-left: -35px;  margin-top: -50px;  }    .container img:nth-child(1) {  z-index: 1;  opacity: .6;  }    .s1 img:nth-child(2) {  z-index: 2;  transform: scaleZ(5) rotateX(45deg);  }    .s2 img:nth-child(2) {  z-index: 2;  transform: scaleZ(.25) rotateX(45deg);  }  </style>  </head>  <body>  <div class="stage s1">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  </div>  </div>  <div class="stage s2">  <div class="container">  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  <img src="img/timg (1).jpg" alt="" width="70" height="100" />  </div>  </div>  </body>  </html> |



2.8 CSS3 3D倾斜

倾斜是二维变形，不能在三维空间变形。元素可能会在X轴和Y轴倾斜，然后转化为三维，但它们不能在Z轴倾斜。

注：

我们一般的时候设置body的透视距离，设置父元素的3D舞台效果，高置后所有的子元素都会变成立体的。

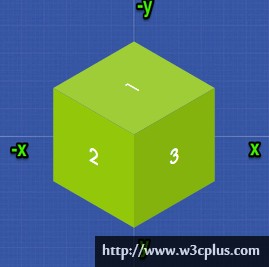
一般和过渡及动画一起连用。

2.9多重变形

在CSS3中，不管是2D变形还是3D变形，我们都可以使用多重变形，他们之间使用空格分隔。

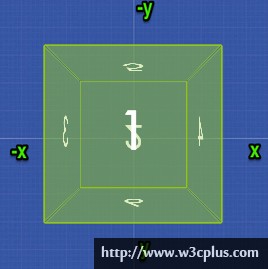
案例一：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">    @keyframes spin {  0% {  transform: rotateY(0deg)  }  100% {  transform: rotateY(360deg)  }  }    .stage {  width: 300px;  height: 300px;  float: left;  margin: 15px;  position: relative;  background: url(img/bg.jpg) repeat center center;  perspective: 1200px;  }    .container {  position: relative;  height: 230px;  width: 100px;  top: 50%;  left: 50%;  margin: -100px 0 0 -50px;  transform-style: preserve-3d;  }    .container:hover {  animation: spin 5s linear infinite;  }    .side {  font-size: 20px;  font-weight: bold;  height: 100px;  line-height: 100px;  color: #fff;  position: absolute;  text-align: center;  text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.2);  text-transform: uppercase;  width: 100px;  }    .top {  background: #9acc53;  transform: rotate(-45deg) skew(15deg, 15deg);  }    .left {  background: #8ec63f;  transform: rotate(15deg) skew(15deg, 15deg) translate(-50%, 100%);  }    .right {  background: #80b239;  transform: rotate(-15deg) skew(-15deg, -15deg) translate(50%, 100%);  }  </style>  </head>  <body>  <div class="stage s1">  <div class="container">  <div class="side top">1</div>  <div class="side left">2</div>  <div class="side right">3</div>  </div>  </div>  </body>  </html> |



案例二：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  @keyframes spin {  0% {  transform: rotateY(0deg)  }  100% {  transform: rotateY(360deg)  }  }  .stage {  width: 300px;  height: 300px;  margin: 15px auto;  position: relative;  background: url(img/bg.jpg) repeat center center;  perspective: 300px;  }  .container {  top: 50%;  left: 50%;  margin: -100px 0 0 -100px;  position: absolute;  transform: translateZ(-100px);  transform-style: preserve-3d;  }  .container:hover {  animation: spin 5s linear infinite;  }  .side {  background: rgba(142, 198, 63, 0.3);  border: 2px solid #8ec63f;  font-size: 60px;  font-weight: bold;  color: #fff;  height: 196px;  line-height: 196px;  position: absolute;  text-align: center;  text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.2);  text-transform: uppercase;  width: 196px;  }  .front {  transform: translateZ(100px);  }  .back {  transform: rotateX(180deg) translateZ(100px);  }  .left {  transform: rotateY(-90deg) translateZ(100px);  }  .right {  transform: rotateY(90deg) translateZ(100px);  }  .top {    transform: rotateX(90deg) translateZ(100px);  }  .bottom {    transform: rotateX(-90deg) translateZ(100px);  }  </style>  </head>  <body>  <div class="stage">  <div class="container">  <div class="side front">1</div>  <div class="side back">2</div>  <div class="side left">3</div>  <div class="side right">4</div>  <div class="side top">5</div>  <div class="side bottom">6</div>  </div>  </div>  </body>  </html> |



在三维旋转中，我们常的一种结构：

舞台=>div.wrapper 容器=>div.item 产品图片＝>img 产品信息=>span.information

实际案例

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style type="text/css">  .wrapper {  display: inline-block;  width: 310px;  height: 100px;  vertical-align: top;  margin: 1em 1.5em 2em 0;  cursor: pointer;  position: relative;  font-family: Tahoma, Arial;  perspective: 4000px;  }    .item {  height: 100px;  transform-style: preserve-3d;  transition: transform .6s;  }    .item img {  display: block;  position: absolute;  width: 310px;  top: 0;  border-radius: 3px;  box-shadow: 0px 3px 8px rgba(0, 0, 0, 0.3);  transform: translateZ(50px);  transition: all .6s;  }    .item .information {  display: block;  position: absolute;  top: 0;  height: 80px;  width: 290px;  text-align: left;  border-radius: 15px;  padding: 10px;  font-size: 12px;  text-shadow: 1px 1px 1px rgba(255, 255, 255, 0.5);  box-shadow: none;  background: rgb(236, 241, 244);  background: -webkit-linear-gradient(to bottom, rgba(236, 241, 244, 1) 0%, rgba(190, 202, 217, 1) 100%);  background: -ms-linear-gradient(to bottom, rgba(236, 241, 244, 1) 0%, rgba(190, 202, 217, 1) 100%);  background: linear-gradient(to bottom, rgba(236, 241, 244, 1) 0%, rgba(190, 202, 217, 1) 100%);  transform: rotateX(-90deg) translateZ(50px);  transition: all .6s;  }    .item:hover {  transform: translateZ(-50px) rotateX(95deg);  }    .item:hover img {  box-shadow: none;  border-radius: 15px;  }    .item:hover .information {  box-shadow: 0px 3px 8px rgba(0, 0, 0, 0.3);  border-radius: 3px;  }  </style>  </head>  <body>  <div class="wrapper">  <div class="item">  <img src="img/mm.png" />  <span class="information">  <strong>Contact Form</strong> The easiest way to add a contact form to your shop. </span>  </div>  </div>  </body>  </html> |



注意：