# 1 transition过渡

过渡是CSS3中具有颠覆性的特征之一，可以实现元素不同状态间的平滑过渡（补间动画），经常用来制作动画效果。

补间动画：自动完成从起始状态到终止状态的的过渡，不用管中间的状态

帧动画：扑克牌切换，通过一帧帧的画面按照顺序和速度播放，如电影

最进本的过渡代码：

<style>  
 div {  
 height: 100px;  
 width: 100px;  
 background-color: #5cb85c;  
 }  
 div:hover {  
 height: 500px;  
 width: 500px;  
 background-color: #2aabd2;

/\*动画完成时间：可以不指定过渡对象，或指定其中某项/多项\*/  
 transition: height 2s, width 3s,background-color 5s;   
 }  
</style>

常用属性（注意属性可以连写）：

transition-property 要运动的样式 （all || [attr] || none）

transition-duration 运动时间

transition-delay 延迟时间

transition-timing-function 运动形式

ease：（逐渐变慢）默认值

linear：（匀速）

ease-in：(加速)

ease-out：（减速）

ease-in-out：（先加速后减速）

cubic-bezier 贝塞尔曲线（ x1, y1, x2, y2 ） <http://matthewlein.com/ceaser/>

transition:hover 鼠标移入时动画

transition的end事件：

Webkit内核： obj.addEventListener('webkitTransitionEnd',function(){},false);

firefox内核： obj.addEventListener('transitionend',function(){},false);

每改变一次样式，都会触发一次end事件（如果绑定了该事件）。

# 2 transform2D转换

转换是CSS3中具有颠覆性的特征之一，可以实现元素的位移、旋转、变形、缩放，甚至支持矩阵方式，配合过渡和动画，可以取代大量之前只能靠Flash才可以实现的效果。在css3 当中，通过transform(变形) 来实现2d 或者3d 转换,其中2d 有：缩放、移动、旋转。

transform-origin的意思是 变换的中心在哪里。

## 2.1 旋转

rotate(deg) 可以对元素进行旋转，正值为顺时针，负值为逆时针；

基本使用：

body:hover div {  
 -webkit-transform: rotate(30deg); /\* 顺时针旋转 \*/  
}

注意：如果要逆时针旋转，为 -90deg

## 2.2 斜切

-webkit-transform: skewX(30deg);

-webkit-transform: skewY(30deg);

-webkit-transform: skew(30deg，30deg);

XY控制斜切的方向

## 2.3 缩放

scale(x, y) 可以对元素进行水平和垂直方向的缩放，x、y的取值可为小数，不可为负值；

transform: scale(1.5,1.5)

## 2.4 移动

translate(x, y) 可以改变元素的位置，x、y可为负值；

x 相对于自身在水平方向移动。

y 相对于自身在垂直方向移动。

位置还原：transform: translate(0,0) rotate(0)

## 2.5 transform知识点

### 2.5.1 中心点

动画的变化会有一个中心点，比如旋转、缩放，即围绕着哪个点进行旋转。

transform-origin:right bottom 围绕右下原点旋转

当然也可以直接书写位置：transform-origin:0 0 （左上角为中心）

第一个参数是 X轴位置，第二个参数是Y轴位置

### 2.5.2 执行顺序

transform 执行顺序问题 — 后写先执行

## 2.6 矩阵

### 2.6.1 矩阵用法

2D转换的效果其实都是通过矩阵来实现的。很多效果IE不支持，但是IE下也可以使用矩阵，我们可以通过矩阵函数来兼容IE678。

在标准浏览器中的矩阵函数

matrix(1,0,0,1,0,0); 参数为初始化的值。

在IE下的矩阵函数（没有标准下的后面2个参数，且与标准顺序不一致：  
且第二个参数、第三个参数位置与标准浏览器位置相反）：

filter(“progid:DXImageTransform.Microsoft.Matrix( M11= 1, M12= 0, M21= 0 , M22=1,SizingMethod='auto expand'”);

参数1：X轴缩放值

参数2：Y轴的倾斜度，计算方式为：Math.tan(xDeg/180\*Math.PI)

（IE6789下为参数3）

参数3：X轴的倾斜度，计算方式为：Math.tan(xDeg/180\*Math.PI)

（IE6789下为参数2）

参数4：Y轴缩放值

参数5：X位移

参数6：Y位移

实例代码：

var box = document.getElementById('box');  
box.onclick = function () {  
 *//X轴缩放0.5,Y轴缩放0.2,X轴斜切0.3,Y轴斜切0.4* box.style.transform = 'matrix(0.5,0.4,0.3,0.2,0,0)'; *//IE10及以上以及大部分高版本浏览器支持*box.style.filter = "progid:DXImageTransform.Microsoft.Matrix( M11= 0.5, M12= 0.3, M21= 0.4 , M22= 0.2, SizingMethod='auto expand')";  
 *// box.style.WebkitTransform = 'matrix(0.5,0,0,0.2,0,0)'; //低版本chrome，以下同理  
 // box.style.MozTransform = 'matrix(0.5,0\*0.2,0,1\*0.2,0,0)';*}

### 2.6.2 矩阵旋转

通过矩阵实现旋转--注意IE下顺序

参数1=Math.cos(deg/180\*Math.PI);

参数2=Math.sin(deg/180\*Math.PI);

参数3=-Math.sin(deg/180\*Math.PI);

参数4=Math.cos(deg/180\*Math.PI);

但是在IE下旋转并不是围绕中心点来展开，需要做处理，案例如下：

<!DOCTYPE HTML>  
<html>  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
 <title>无标题文档</title>  
 <style>  
 .box{width:100px;height:100px;margin:30px auto; position:relative;border:1px solid #000;}  
 #box{width:100px;height:100px;background:red; position:absolute;left:0;top:0;}  
 </style>  
</head>  
<body>  
<div class="box">  
 <div id="box"></div>  
</div>  
<script>  
 var oBox=document.getElementById("box");  
 var iDeg=0;  
 setInterval(function(){  
 iDeg++;  
 toRotate(oBox,iDeg);  
 },30);  
 function toRotate(obj,iDeg)  
 {  
 var a=Math.round(Math.cos(iDeg/180\*Math.PI)\*100)/100;  
 var b=Math.round(Math.sin(iDeg/180\*Math.PI)\*100)/100;  
 var c=-b;  
 var d=a;  
 obj.style.WebkitTransform="matrix("+a+","+b+","+c+","+d+",0,0)";  
 obj.style.MozTransform="matrix("+a+","+b+","+c+","+d+",0,0)";  
 obj.style.transform="matrix("+a+","+b+","+c+","+d+",0,0)";  
 obj.style.filter="progid:DXImageTransform.Microsoft.Matrix( M11="+a+", M12= "+c+", M21= "+b+" , M22="+d+",SizingMethod='auto expand')";  
 obj.style.left=(obj.parentNode.offsetWidth-obj.offsetWidth)/2+"px";  
 obj.style.top=(obj.parentNode.offsetHeight-obj.offsetHeight)/2+"px";  
 }  
</script>  
</body>  
</html>

# 3 transform3D转换

## 1 初识3D

<style>  
 #box {  
 width: 400px;  
 height: 400px;  
 margin: 100px auto;  
 border: 1px solid #2aabd2;  
 transform-style: preserve-3d; /\* 建立3D空间 \*/  
 perspective: 150px /\* 景深基点--站在多远观看\*/  
 }  
 #mini {  
 margin: 100px auto;  
 width: 200px;  
 height: 200px;  
 background-color: #5cb85c;  
 transition: 1s;  
 }  
 #box:hover #mini{  
 transform: rotateX(180deg);  
 }  
 </style>  
  
<div id="box">  
 <div id="mini">666</div>  
</div>

常见属性：

transform-style: preserve-3d; 建立3D空间

perspective: 150px 景深基点--站在多远观看

backface-visibility:hidden 设置盒子背面是否隐藏

新增的一些函数：

rotateX()

rotateY()

rotateZ()

translateZ() 必须配合透视（即景深基点）使用

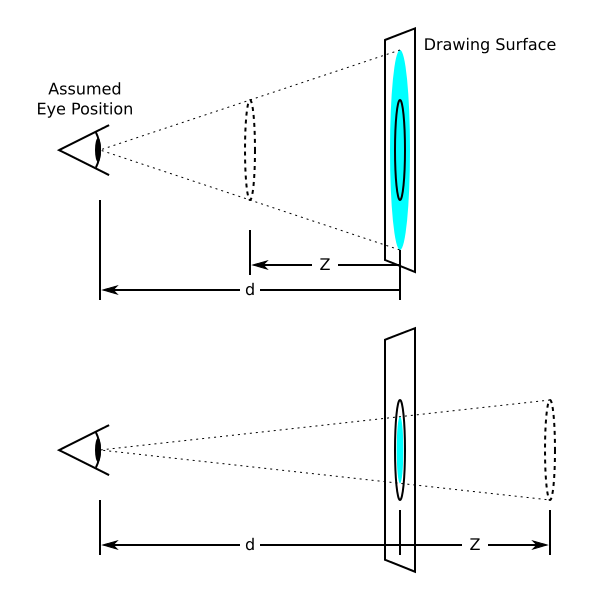
scaleZ()

## 2 CSS中的3D坐标系

CSS3中的3D坐标系与上述的3D坐标系是有一定区别的，相当于其绕着X轴旋转了180度，如下图



电脑显示屏是一个2D平面，图像之所以具有立体感（3D效果），其实只是一种视觉呈现，通过透视可以实现此目的。透视可以将一个2D平面，在转换的过程当中，呈现3D效果。透视会产生“近大远小”的效果。



透视属性perspective可以设置离用户眼睛的距离，距离越远，立体效果越小。透视有两种写法：作为一个属性，设置给父元素，作用于所有3D转换的子元素；作为transform属性的一个值，做用于元素自身。

# 4 动画animation

动画是CSS3中具有颠覆性的特征之一，可通过设置多个节点来精确控制一个或一组动画，常用来实现复杂的动画效果。

代码示例：

div {  
 width: 200px;  
 height: 200px;  
 background-color: #5cb85c;  
 animation: 2S move; /\* 指定动画名称和时间 \*/  
}  
@keyframes move {  
 50%{  
 width: 1000px;  
 }  
}

动画问题：动画执行完毕后，盒子会回到原始状态，如果我们不想其回到原始状态，一般可以给其设定一个额外的class。

<style>  
 .div {

width: 100px;  
 height: 100px;  
 background-color: #5cb85c;  
 }  
 .move {  
 animation: 2S move; /\* 指定动画名称和时间 \*/  
 width: 500px;  
 }  
 @keyframes move {  
 0%{  
 width: 100px;  
 }  
 100%{  
 width: 500px;  
 }  
 }  
 </style>  
<div class="div move"></div>

注意：参数forward 可以让动画保持在最后一帧

keyFrames：即关键帧。只需指明两个状态，之间的过程由计算机自动计算

格式1

@keyframes 动画名称

{

动画状态

}

格式2

@keyframes 动画名称

{

from { background:red; }

to { background:green; }

}

可以只有to

animation：定义动画的执行时间和执行行为是谁（名字）。当然也有一些可选参数如：animation: 2S move ease-in;

常见属性与对应值：

animation-name 必要属性-动画名称（关键帧名称）

animation-duration 必要属性-动画持续时间

例如：

-webkit-animation-name: ‘miaov';

-webkit-animation-duration: 4s;

animation-timing-function 动画运动形式

linear 匀速

ease 缓冲

ease-in 由慢到快

ease-out 由快到慢

ease-in-out 由慢到快再到慢

animation-play-state 播放状态（ running 播放 和paused 暂停 ）

animation-delay 动画延迟（只是第一次）

animation-iteration-count 重复次数（infinite为无限次）

animation-direction 动画是否重置后再开始播放

alternate 动画直接从上一次停止的位置开始执行

normal 动画第二次直接跳到0%的状态开始执行

cubic-bezier(number, number, number, number)：

特定贝塞尔曲线类型，4个数值需在[0, 1]区间内

end函数：

obj.addEventListener('webkitAnimationEnd', function (){}, false);