# CSS3 转换

* 通过 CSS3 转换，我们能够对元素进行

1. 移动 translate() ; transform ; translate() 方法，元素从其当前位置移动，根据给定的 left（x 坐标） 和 top（y 坐标） 位置参数：

translateY(n) 定义 2D 转换，沿着 X 轴移动元素。

translateX(n) 定义 2D 转换，沿着 Y 轴移动元素。

\*{

transform:translate(50px,100px);

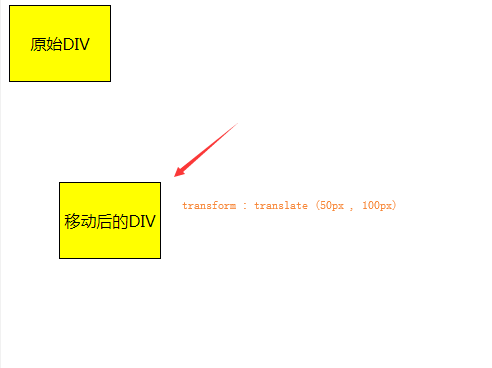
-ms-transform:translate(50px,100px); /\* IE 9 \*/

-moz-transform:translate(50px,100px); /\* Firefox \*/

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

-o-transform:translate(50px,100px); /\* Opera \*/

}



2. 旋转rotate() 元素顺时针旋转给定的角度。允许负值，元素将逆时针旋转。

注意rotate的单位是 deg

\*{

transform:rotate(30deg);

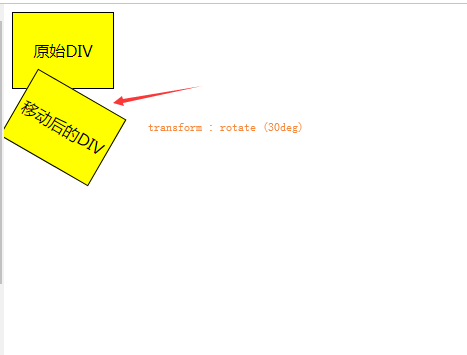
-ms-transform:rotate(30deg); /\* IE 9 \*/

-moz-transform:rotate(30deg); /\* Firefox \*/

-webkit-transform:rotate(30deg); /\* Safari and Chrome \*/

-o-transform:rotate(30deg); /\* Opera \*/

}



3. 缩放 scale() 元素的尺寸会 增加或减少，根据给定的宽度（X 轴）和高度（Y 轴）参数

参数可以为 负数 ， 小数

scaleX(n) 定义 2D 缩放转换，改变元素的宽度。

scaleY(n) 定义 2D 缩放转换，改变元素的高度。

.div1{

transform: scale(2,4);

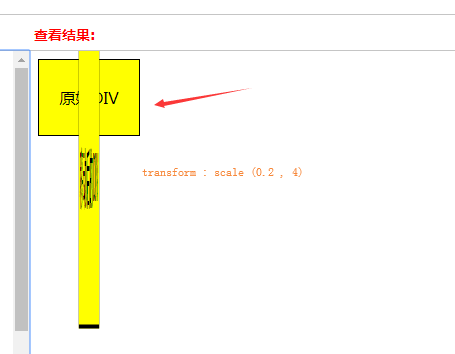
-ms-transform: scale(2,4);  /\* IE 9 \*/

-webkit-transform: scale(2,4);  /\* Safari 和 Chrome \*/

-o-transform: scale(2,4);   /\* Opera \*/

-moz-transform: scale(2,4); /\* Firefox \*/





4. 拉长或拉伸 skew()/matrix()

skewX(angle) 定义 2D 倾斜转换，沿着 X 轴。

skewY(angle) 定义 2D 倾斜转换，沿着 Y 轴。

拉长skew()

transform : skew(<angle> [,<angle>])

包含两个参数值，分别表示X轴和Y轴倾斜的角度，如果第二个参数为空，则默认为0，参数为负表示向相反方向倾斜。

skew(30deg,20deg) 元素在X轴和Y轴上倾斜20度30度。

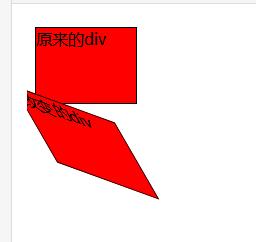
\*{

    transform:skew(30deg,20deg);

    -ms-transform:skew(30deg,20deg); /\* IE 9 \*/

    -webkit-transform:skew(30deg,20deg); /\* Safari and Chrome \*/

}



拉伸matrix()

matrix()方法和2D变换方法合并成一个

matrix 方法有六个参数，包含旋转，缩放，移动（平移）和倾斜功能。

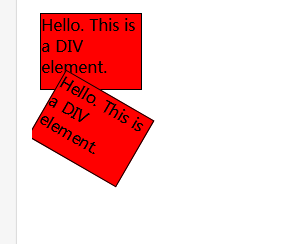
\*{

transform:matrix(0.866,0.5,-0.5,0.866,0,0);

-ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /\* IE 9 \*/

-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /\* Safari and Chrome \*/

}



matrix()与各个效果的对应关系

Matrix()与translate()关系：translate(x,y)对应Matrix(1,0,0,1,x,y);

Matrix()与scale()关系：scale(x,y)对应Matrix(x,0,0,y,0,0);

Matrix()与rotate()关系：rotate(x)对应Matrix(cos(x),sin(x),-sin(x),cos(x),0,0)

Matrix()与skew()关系：skew(x,y)对应Matrix(1,tan(x),tan(y),1,0,0)

# CSS3 过渡

CSS3 过渡是元素从一种样式逐渐改变为另一种的效果

要实现这一点，必须规定两项内容：

指定要添加效果的CSS名称 指定效果的持续时间

transition

要添加多个样式的变换效果，添加的属性由逗号分隔：

div {

width: 100px;

height: 100px;

background: red;

transition: width 20s, height 2s, transform 2s;

}

div:hover {

width: 200px;

height: 200px;

transform: rotate(180deg);

}

animation动画效果

注意要有 @keyframes 后面跟上动画的 名字(**animation-name**) 指定要绑定到选择器的关键帧的名称

**animation-duration**定义动画完成一个周期需要多少秒或毫秒

**animation-timing-function**：指定动画将如何完成一个周期。

基础版

div

{

    width:100px;

    height:100px;

    background:red;

    position:relative;

    animation:mymove 5s infinite ;

}

@keyframes mymove

{

    0% {left:0px;}

    100% {left:200px;}

}

| **值** | **说明** |
| --- | --- |
| linear | 动画从头到尾的速度是相同的。 |
| ease | 默认。动画以低速开始，然后加快，在结束前变慢。 |
| ease-in | 动画以低速开始。 |
| ease-out | 动画以低速结束。 |
| ease-in-out | 动画以低速开始和结束。 |
| cubic-bezier(n,n,n,n) | 在 cubic-bezier 函数中自己的值。可能的值是从 0 到 1 的数值。 |
| step-start | 在变化过程中，都是以下一帧的显示效果来填充间隔动画 |
| step-end | 在变化过程中，都是以上一帧的显示效果来填充间隔动画 |
| steps() | 可以传入两个参数，第一个是一个大于0的整数，他是将间隔动画等分成指定数目的小间隔动画，然后根据第二个  参数来决定显示效果。第二个参数设置后其实和step-start，step-end同义，在分成的小间隔动画中判断显示效果。 |

cubic-bezier(n,n,n,n)称为三次贝塞尔曲线，速度曲线。

cubic-bezier(n,n,n,n)曲线定制网站地址：<http://cubic-bezier.com/#.23,.66,.76,.52>

**animation-delay**：属性定义动画什么时候开始。

1 单位可以是秒（s）或毫秒（ms）。

2 单位可以是负值，-2s表示动画立马开始，但跳过 2 秒进入动画，即前2秒的动画不执行，直接跳过前2秒进入动画。

加载条 demo地址 : <http://codepen.io/aSuncat/pen/ALLNpE>

.spinner{

width:60px;

height:50px;

margin:100px auto;

text-align:center;

}

.spinner>div{

display:inline-block;

width:6px;

margin-left:3px;

margin-right:3px;

height:100%;

background:green;

animation:strechdelay 1.2s infinite ease-in-out;

-webkit-animation:strechdelay 1.2s infinite ease-in-out;

}

.spinner .line2{

animation-delay:-1.1s;

-webkit-animation-delay:-1.1s;

}

.spinner .line3{

animation-delay:-1.0s;

-webkit-animation-delay:-1.0s;

}

.spinner .line4{

-webkit-animation-delay:-0.9s;

}

.spinner .line5{

animation-delay:-0.8s;

-webkit-animation-delay:-0.8s;

}

@keyframes strechdelay {

0%,40%,100%{

transform:scaleY(.4);

-webkit-transform:scaleY(.4);

}

20%{

transform:scaleY(1);

-webkit-transform:scaleY(1);

}

}

@-webkit-keyframes strechdelay {

0%,40%,100%{

-webkit-transform:scaleY(.4);

}

20%{

-webkit-transform:scaleY(1);

}

}

京东全民寻宝文字 demo地址：<http://codepen.io/aSuncat/pen/WGoZGm>

@-webkit-keyframes title2{

24.24%{

background-position:0 0

}

to{

background-position:0 -416px

}

}

@keyframes title2{

24.24%{

background-position:0 0

}

to{

background-position:0 -416px

}

}

.icon\_title\_2{

background:#000 url(//wq.360buyimg.com/fd/promote/201609/iphone7/images/title2-s685478f868\_3b31377c.png);

background-position:0 0;

background-size:246px 416px;

height:52px;

width:246px;

animation:title2 2.48s steps(8) infinite both;

-webkit-animation:title2 2.48s steps(8) infinite both;

}

**animation-iteration-count**  定义动画应该播放多少次。

| **值** | **说明** |
| --- | --- |
| n | 一个数字，定义应该播放多少次动画 |
| infinite | 指定动画应该播放无限次（永远） |

**animation-direction**：定义是否循环交替反向播放动画

| **值** | **说明** |
| --- | --- |
| normal | 默认值。动画按正常播放。 |
| reverse | 动画反向播放。 |
| alternate | 动画在奇数次（1、3、5…）正向播放，在偶数次（2、4、6…）反向播放。 |
| alternate-reverse | 动画在奇数次（1、3、5…）反向播放，在偶数次（2、4、6…）正向播放。 |
| initial | 设置该属性为它的默认值。 |
| inherit | 从父元素继承该属性。 |

**animation-fill-mode**：规定当动画不播放时（当动画完成时，或当动画有一个延迟未开始播放时），要应用到元素的样式。

| **值** | **说明** |
| --- | --- |
| none | 默认值。动画在动画执行之前和之后不会应用任何样式到目标元素。 |
| forwards | 在动画结束后（由 animation-iteration-count 决定），动画将应用该属性值。 |
| backwards | 动画将应用在 animation-delay 定义期间启动动画的第一次迭代的关键帧中定义的属性值。这些都是 from 关键帧中的值（当 animation-direction 为 “normal” 或 “alternate” 时）或 to 关键帧中的值（当 animation-direction 为 “reverse” 或 “alternate-reverse” 时）。 |
| both | 动画遵循 forwards 和 backwards 的规则。也就是说，动画会在两个方向上扩展动画属性。 |
| initial | 设置该属性为它的默认值。 |
| inherit | 从父元素继承该属性。 |

@keyframes

| **值** | **说明** |
| --- | --- |
| animationname | 必需的。定义animation的名称。 |
| keyframes-selector | 必需的。动画持续时间的百分比。 |
| css-styles | 必需的。一个或多个合法的CSS样式属性 |

指定的变化时发生时使用％，或关键字”from”和”to”，这是和0％到100％相同。以下两段代码效果相同。

其实 from就等于 0% ； to就等于 100%

@keyframes myAnim{

from { background: #f00; }

50% { background: #0f0; }

to { background: yellowgreen; }

}

@keyframes myAnim{

0% { background: #f00; }

50% { background: #0f0; }

100% { background: yellowgreen; }

}

可以多个状态一起写

@keyframes myAnim{

0%，100% { background: #f00; }

50% { background: #0f0;

}