

HTML5程序设计基础

第十六章 渐变与变形处理



主要内容

01

渐变效果

02

transform

03

transform-origin

04

多重变形

01

渐变效果



渐变效果

🎨 网页中的渐变效果包括渐变背景、渐变导航、渐变按钮等

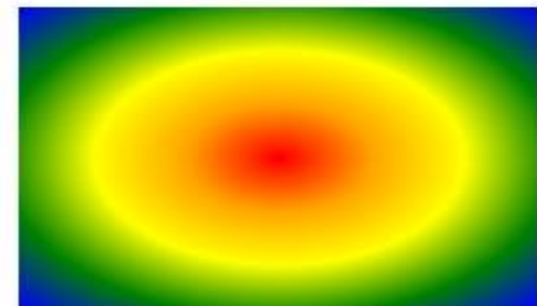
🎨 CSS3 渐变分类

➤ 线性渐变、径向渐变

🎨 CSS3 渐变优点

➤ 代替使用图像来实现效果，可以减少下载的时间和宽带的使用。

➤ 由浏览器生成，在放大时看起来效果更好



线性渐变

线性渐变

```
background: linear-gradient(direction, color-stop1, color-stop2, ...);
```

- 在一条直线上进行颜色渐变，渐变线由包含渐变图形的容器的中心点和一个角度来定义的。
- linear-gradient() 函数创建一个没有内在尺寸的，表示颜色线性渐变的 <image> 图像；它既不具有固有的或首选的尺寸，也不具有比率。它的具体尺寸将与其适用的元素尺寸匹配。

线性渐变

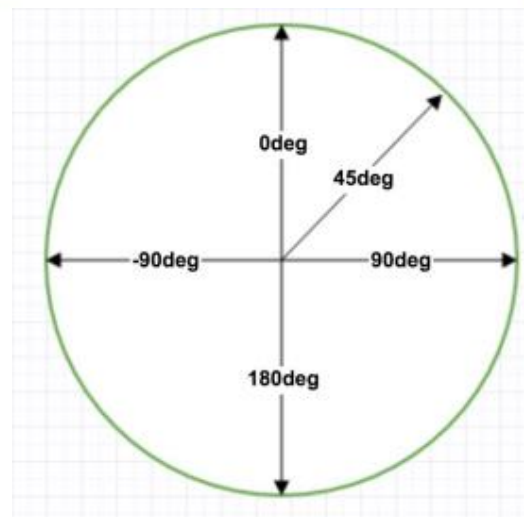
渐变方向，等价于180deg

linear-gradient(to bottom, #fff, #999)

渐变类型，
径向为radial

表示颜色的起始点和结束点，
可以有两至多个色值

角度	用英文	作用
0deg	to top	从下向上
90deg	to right	从左向右
180deg	to bottom	从上向下
270deg	to left	从右向左
	to top left	右下角到左上角
	to top right	左下角到右上角

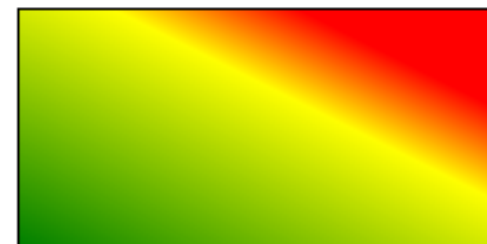


第一个参数：**指定渐变方向**，可以用“**角度**”或“**英文关键词**”来表示。
第一个参数省略时，默认为“180deg”，等同于“to bottom”。
第二个和第三个参数，表示颜色的起始点和结束点，可以有多个颜色值。

线性渐变

demo16-1.html

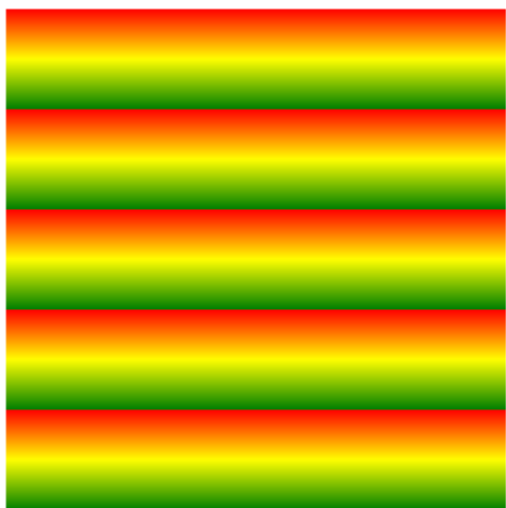
```
.div1 {  
  background: linear-gradient(red, yellow, green);  
}  
.div2 {  
  background: linear-gradient(to bottom left, red, yellow, green);  
}  
.div3 {  
  background: linear-gradient(to bottom left, red 20%, yellow 40%,  
green);  
}  
.div4 {  
  background: linear-gradient(60deg, red, yellow, green);  
}
```



重复的线性渐变

🎨 **repeating-linear-gradient()** 函数用于重复线性渐变：

```
background: repeating-linear-gradient(red, yellow 10%, green 20%);
```



[demo16-2.html](#)

练习

切角效果

- 把一个透明色标放在切角处，然后在相同的位置设置另一个色标



渐变按钮



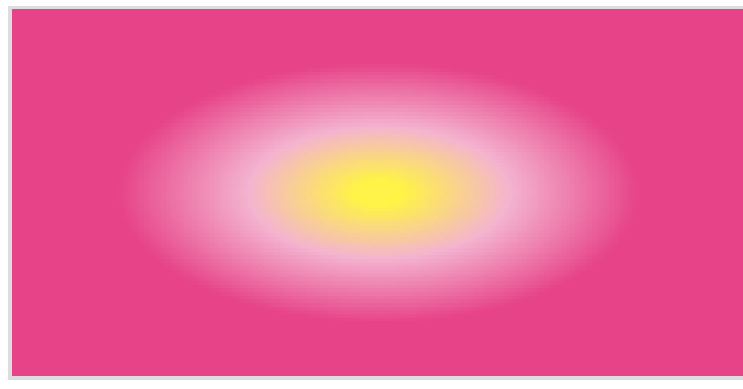
demo16-3.html

径向渐变

🎨 径向渐变

background: radial-gradient (shape at position, color1 stop, color2 stop,...)

- 是一种从起点到终点颜色从内到外进行圆形渐变（从中间向外拉，像圆一样）。



径向渐变

shape

➤ 关键字

- **circle** —— 定义径向渐变为 "圆形"
- **ellipse** —— 定义径向渐变为 "椭圆形"

position

➤ 关键字

➤ 长度值

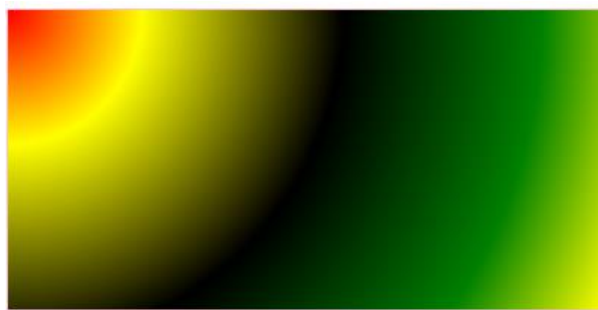
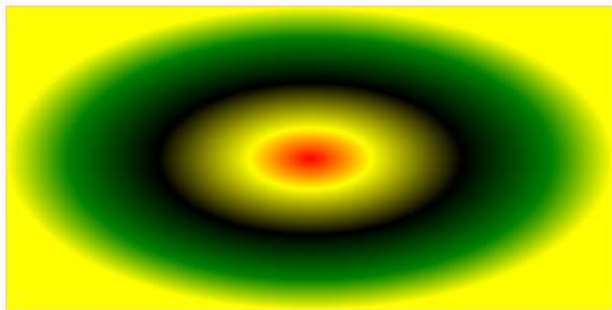
- 长度值 px(可负)
- 百分比 30%(可负)

属性值	说明
center	中部 (默认值)
top	顶部
right	右部
bottom	底部
left	左部
top left	左上
top center	靠上居中
top right	右上
left center	靠左居中
center center	正中
right center	靠右居中
bottom left	左下
bottom center	靠下居中
bottom right	右下

径向渐变

- 在径向渐变中没有设置位置时，其默认颜色为均匀间隔，设置了渐变位置就会按照渐变位置去渐变。

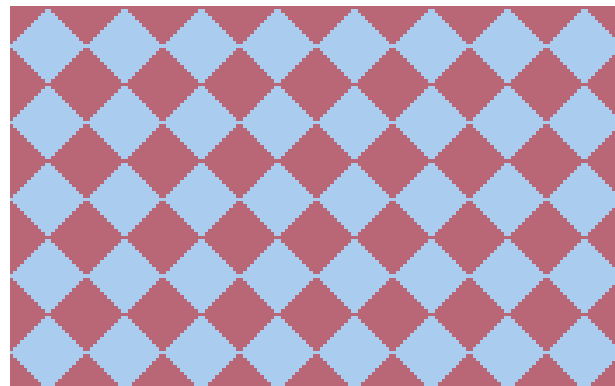
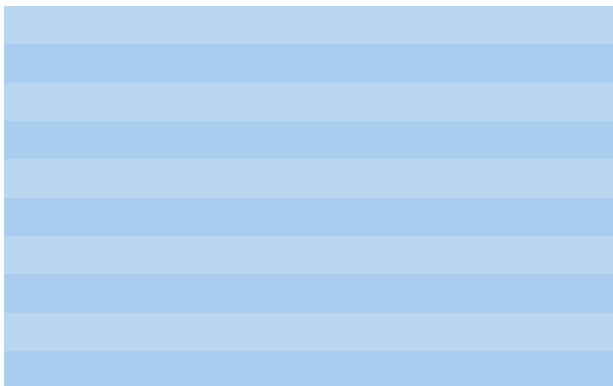
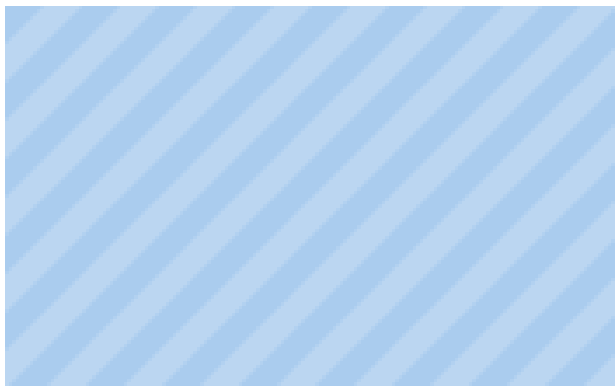
```
.div1{  
    background: radial-gradient( 50% 50%,red 0%, yellow  
    20%,black 50%,green 80%, yellow);}  
.div3{  
    background: radial-gradient(circle at top left,  
    red 0%, yellow 20%,black 50%,green 80%, yellow);}
```



[demo16-4.html](#)

渐变应用

制作背景



demo16-5.html





transform








CSS3变形

- ❖ 在 CSS3 中提供了 **transform** 和 **transform-origin** 两个用于实现 2D 变换的属性。
 - transform 属性用于实现平移、缩放、旋转和倾斜等 2D 变换。
 - transform-origin 属性则是用于设置变换的中心点的。

transform 属性

🎨 transform 属性向元素应用 2D 或 3D 转换。通过转换能够对元素进行旋转、缩放、移动、倾斜或拉伸。

🎨 浏览器支持

属性	浏览器支持				
transform					

- Chrome 和 Safari 需要前缀 **-webkit-**
- Internet Explorer 9 需要前缀 **-ms-**

transform 属性

语法 **transform:** none | transform-functions;

transform属性的属性值由值及函数组成

值	描述
none	定义不进行转换。
matrix(n,n,n,n,n,n)	定义 2D 转换，使用六个值的矩阵。
matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	定义 3D 转换，使用 16 个值的 4x4 矩阵。
translate(x,y)	定义 2D 转换。
translate3d(x,y,z)	定义 3D 转换。
translateX(x)	定义转换，只是用 X 轴的值。
translateY(y)	定义转换，只是用 Y 轴的值。
translateZ(z)	定义 3D 转换，只是用 Z 轴的值。
scale(x,y)	定义 2D 缩放转换。

<code>scale3d(x,y,z)</code>	定义 3D 缩放转换。
<code>scaleX(x)</code>	通过设置 X 轴的值来定义缩放转换。
<code>scaleY(y)</code>	通过设置 Y 轴的值来定义缩放转换。
<code>scaleZ(z)</code>	通过设置 Z 轴的值来定义 3D 缩放转换。
<code>rotate(angle)</code>	定义 2D 旋转，在参数中规定角度。
<code>rotate3d(x,y,z,angle)</code>	定义 3D 旋转。
<code>rotateX(angle)</code>	定义沿着 X 轴的 3D 旋转。
<code>rotateY(angle)</code>	定义沿着 Y 轴的 3D 旋转。
<code>rotateZ(angle)</code>	定义沿着 Z 轴的 3D 旋转。
<code>skew(x-angle,y-angle)</code>	定义沿着 X 和 Y 轴的 2D 倾斜转换。
<code>skewX(angle)</code>	定义沿着 X 轴的 2D 倾斜转换。
<code>skewY(angle)</code>	定义沿着 Y 轴的 2D 倾斜转换。
<code>perspective(n)</code>	为 3D 转换元素定义透视视图。

移动—translate()方法

🎨 translate() 方法能够重新定位元素的坐标。

- translate**X**(x)：元素仅在水平方向移动（X轴移动）；
- translate**Y**(y)：元素仅在垂直方向移动（Y轴移动）；
- translate(x , y)：元素在水平方向和垂直方向同时移动；

说明：

在实际开发中需要根据情况添加各浏览器厂商的前缀。

Firefox浏览器添加-moz-前缀；IE浏览器添加-ms-前缀；

Opera浏览器添加-o-前缀；Chrome浏览器添加-webkit-前缀。

移动—translate()方法

```
div{  
    width:200px;  
    height:150px;  
    background-color:yellow;  
    border:1px solid black;  
}  
#div2{  
    transform: translate(50px,50px);  
}
```

你好。这是一个 div 元素。

示例文字

demo16-6.html

translate()方法实例

给导航菜单添加定位功能，使导航菜单更富动感。



demo16-7.html

旋转—rotate() 方法

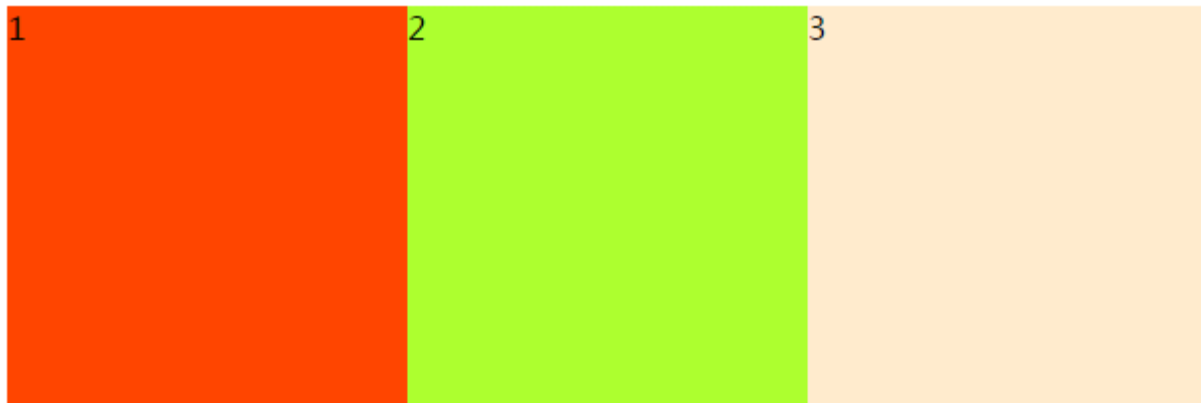
🎨 rotate() 方法能够相对中心原点**旋转指定的**元素。

- transform: **rotate(angle)**
- 正角度为顺时针旋转元素
- 负角度为逆时针旋转元素



demo16-8.html

rotate()旋转



demo16-9.html

```
div:hover{  
    transform: rotate(-50deg);  
}
```

缩放—scale() 方法

🎨 scale() 方法能够实现文字或图像根据中心原点进行**缩放**。

- scale**X**(x)：元素仅水平方向缩放（X轴缩放）；
- scale**Y**(y)：元素仅垂直方向缩放（Y轴缩放）；
- scale(x, y)：元素水平方向和垂直方向同时缩放；
- 参数 x, y 为自然数数值（可以为正、负、小数）

说明：

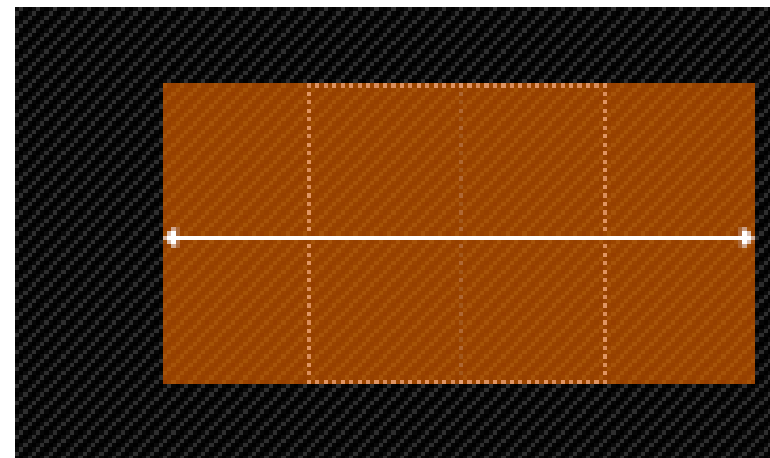
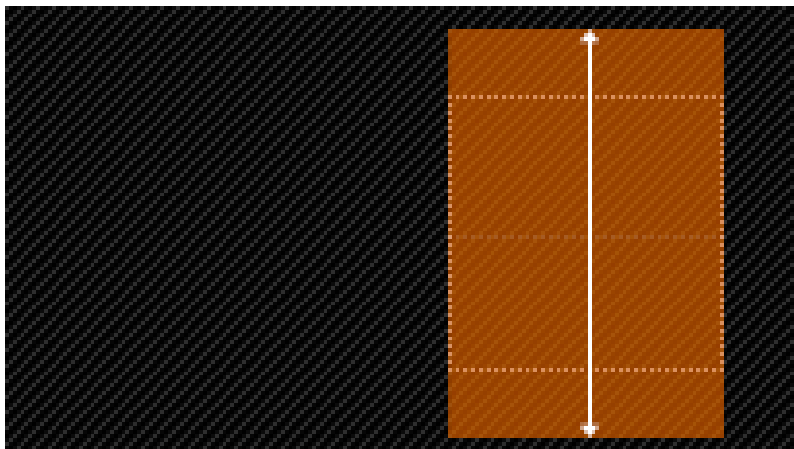
绝对值大于1，代表放大；绝对值小于1，代表缩小；

当参数值为1时，表示不进行缩放；

当值为负数时，对象反转。

scale()缩放

transform:scaleX(2)



transform:scaleY(2)

demo16-10.html

scale()缩放

transform:scaleX(-2)

HTML+CSS

220+JMT H

注意：当使用scaleX(num)或scaleY(num)函数时，实现的是**非等比例**缩放

demo16-11.html

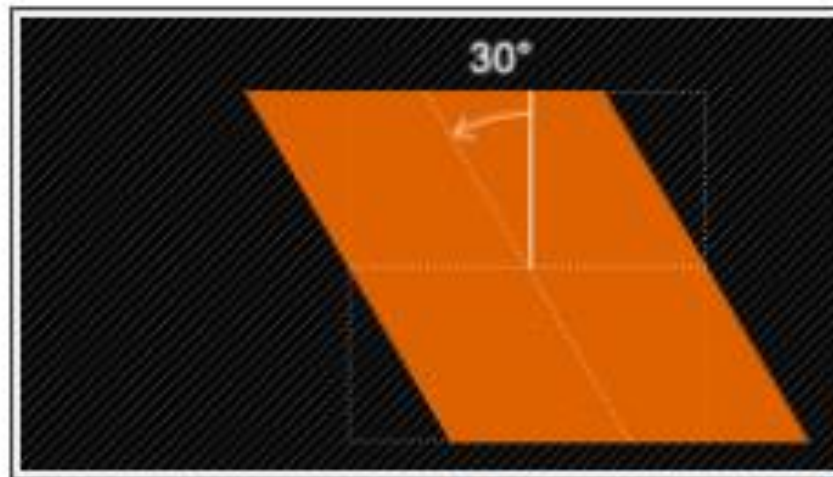
倾斜—skew()方法

🎨 skew() 方法能够倾斜指定的元素。

- skewX(x)：元素仅水平方向倾斜（X轴倾斜）；
- skewY(y)：元素仅垂直方向倾斜（Y轴倾斜）；
- skew(x, y)：元素水平方向和垂直方向同时倾斜；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。

倾斜—skew()

transform: **skewX(30deg)**;



transform: **skewY(10deg)**;

demo16-12.html

3D变形功能

CSS3 允许使用 3D 转换来实现元素在X轴、Y轴、Z轴方向上的变形处理

`rotateX(angle)`

`rotateY(angle)`

`rotateZ(angle)`

- 元素围绕其 X 轴以给定的度数进行旋转。
- 元素围绕其 Y 轴以给定的度数进行旋转。
- 元素围绕其 Z 轴以给定的度数进行旋转。

3D旋转

3D旋转变形实例：

页面中显示一个div元素以及一个“绕X轴旋转”按钮、一个“绕Y轴旋转”按钮及一个“绕Z轴旋转”按钮。用户单击各按钮时脚本程序通过修改div元素的transform属性值中rotateX、rotateY、rotateZ方法的参数值使div元素分别围绕X、Y、Z轴旋转180°。

示例文字

绕X轴旋转

绕Y轴旋转

绕Z轴旋转

demo16-13.html



transform-origin



transform-origin

🎨 transform-origin 属性更改变换的基点位置。

```
transform-origin: x-axis y-axis;
```

- 默认情况下，元素基点位置为元素的中心点，即X 轴和 Y 轴的 50% 处。
- CSS3 变形进行的位移、缩放、旋转、倾斜都是以元素的基点进行变形。

transform-origin属性值

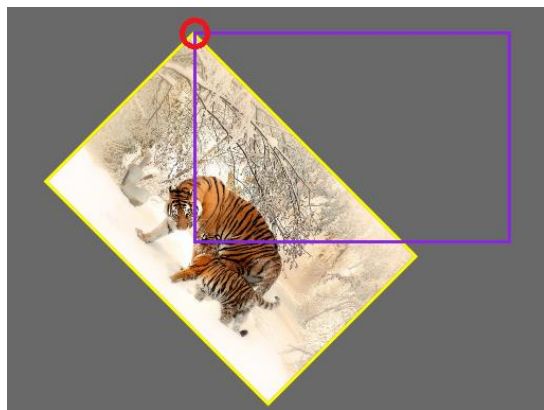
🎨 **transform-origin**: x-axis y-axis;

值	描述
x-axis	指定原点被置于 X 轴的何处。可能的值： left center right length %
y-axis	指定原点被置于 Y 轴的何处。可能的值： top center bottom length %

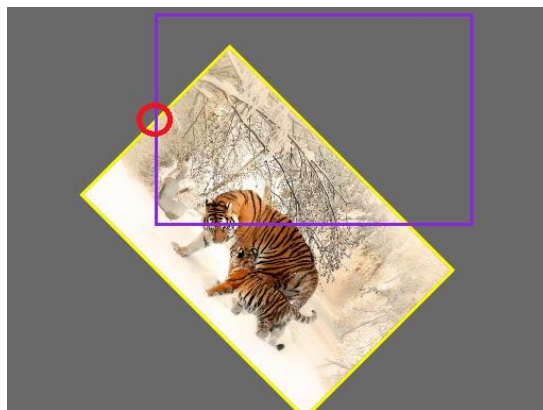
🎨 **默认值**为 center center ，等价于 50% 50%

变换基点

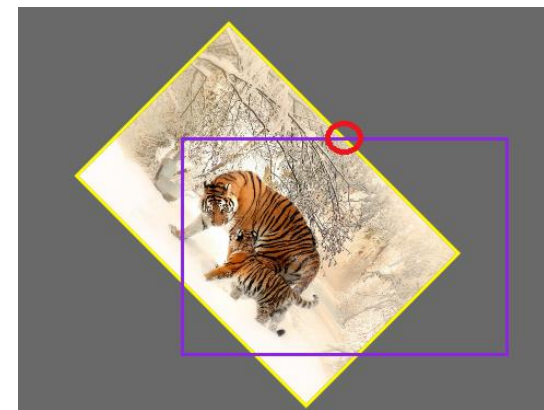
left-top



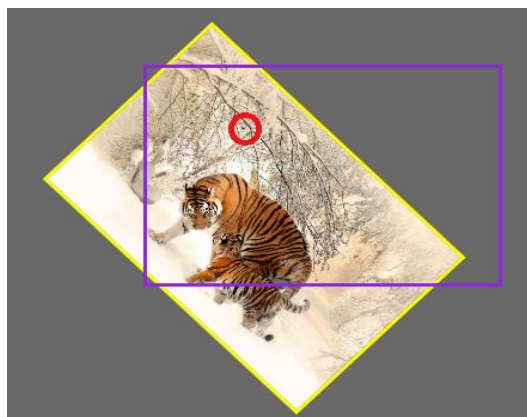
left-center



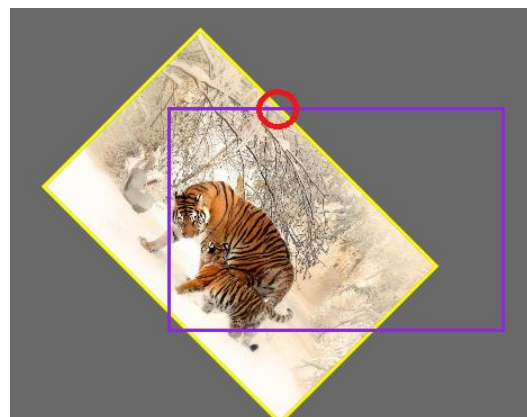
center-top



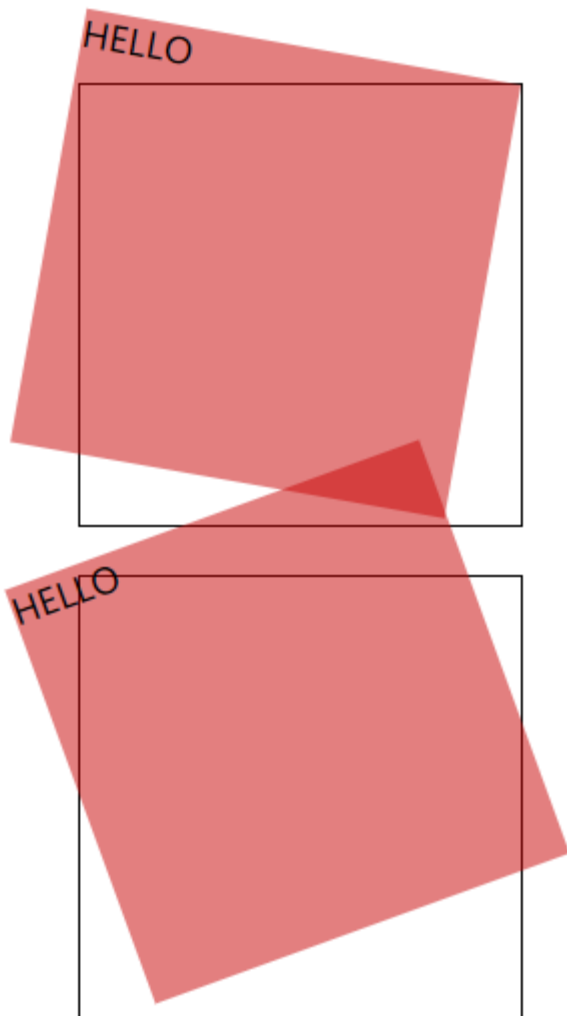
25% 25%



100px 0



变换基点



```
#div1-1{
  width: 200px;
  height: 200px;
  position: absolute;
  background-color: rgba(200,0,0,0.5);
  transform: rotate(10deg);
  transform-origin:top right;
}
#div1-2{
  width: 200px;
  height: 200px;
  position: absolute;
  background-color: rgba(200,0,0,0.5);
  transform: rotate(-20deg);
  transform-origin:0 100px;
}
```

04

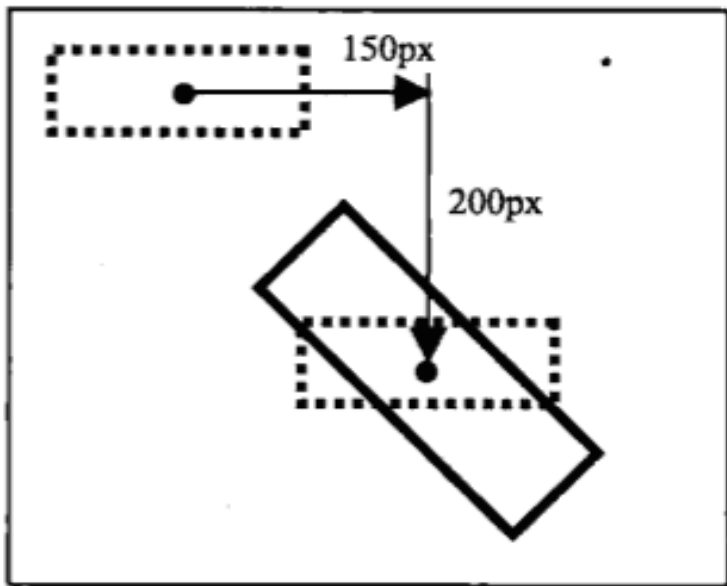
多重变形



多重变形

对同一元素可添加多种变形效果。

```
transform: translate(150px,200px) rotate(120deg) scale(1.5,1.5);
```



元素经过移动后旋转并放大

demo16-15.html

案例实战

- ❖ 不使用JS，综合运用CSS阴影、透明效果、变形动画，设计涂鸦墙。
使用移动、旋转、缩放等函数控制元素创建丰富、轻量级的界面应用。
- ❖ 默认状态下图片被随意显示在墙面上，鼠标经过图片时会竖直摆放，并被放大显示。



THANKYOU

