day15

过渡【9：03】

1.过渡概念

过渡可以为一个元素在“不同状态之间”切换，在切换时定义不同的过渡效果。如果改变状态，使用伪类。

过渡属性,`transition`这是一个集合写法，过渡，过渡需要的时间，过渡的方式和过渡延迟的时间。

2.过渡属性的拆分

含义	属性	值
过渡名称	transition-property	过渡的属性名，默认all
过渡时间	transition-duration	必要属性，时间单位s
延迟时间	transition-delay	时间单位s
过渡样式	transition-timing-function	ease 默认 慢-快-慢 ease-in 慢-快 ease-out 快-慢 ease-in-out 慢-匀速-慢 linear 匀速

3.过渡的简写

（1）简写语法

```
transition:过渡名称 过渡时间 延迟时间 过渡方式;
如
transition: box-shadow 1s 1s linear;
```

注意：除了过渡时间一定要写在延迟时间之前，其他的顺序都可以改。

（2）最简写法

```
原本语法：
            all   0s[必要]   0s     ease
transition:过渡名称 过渡时间 延迟时间 过渡方式;
最简写法：
transition:过渡时间;
```

（3）多组过渡

重要的是，下一组的延迟时间等于之前组的执行时间之和

```
transition: 1s background-color,1s 1s border-radius,1s 2s box-shadow;
```

练习

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <style type="text/css">
            .box1 {
                width: 100px;
                height: 100px;
                background-color: red;
                /* 过渡的简写 */
                /*         过渡属性  时间 延迟 过渡方式*/
                transition: box-shadow 1s 1s linear;
            }
            .box1:hover {
                box-shadow: 5px 5px 5px 0 #000;
            }
            /* 进度条 */
            /* 1.渲染外层区域 */
            .box2 {
                width: 600px;
                height: 30px;
                border: 3px solid #000000;
            }
            /* 2.可动进度要先初始化渲染 */
            .bar {
                width: 0%;
                height: 30px;
                background-color: red;
                /* 过渡 */
                transition: 1s;
            }
            /* 3.鼠标悬停的位置其实只能是能看到的box2 */
            .box2:hover .bar{
                width: 100%;
            }
        </style>
        <title></title>
    </head>
    <body>
        <!-- 红方块，鼠标移入阴影 -->
        <div class="box1"></div>

        <!-- 进度条 -->
        <div class="box2">
            <div class="bar"></div>
        </div>
    </body>
</html>
```

变换

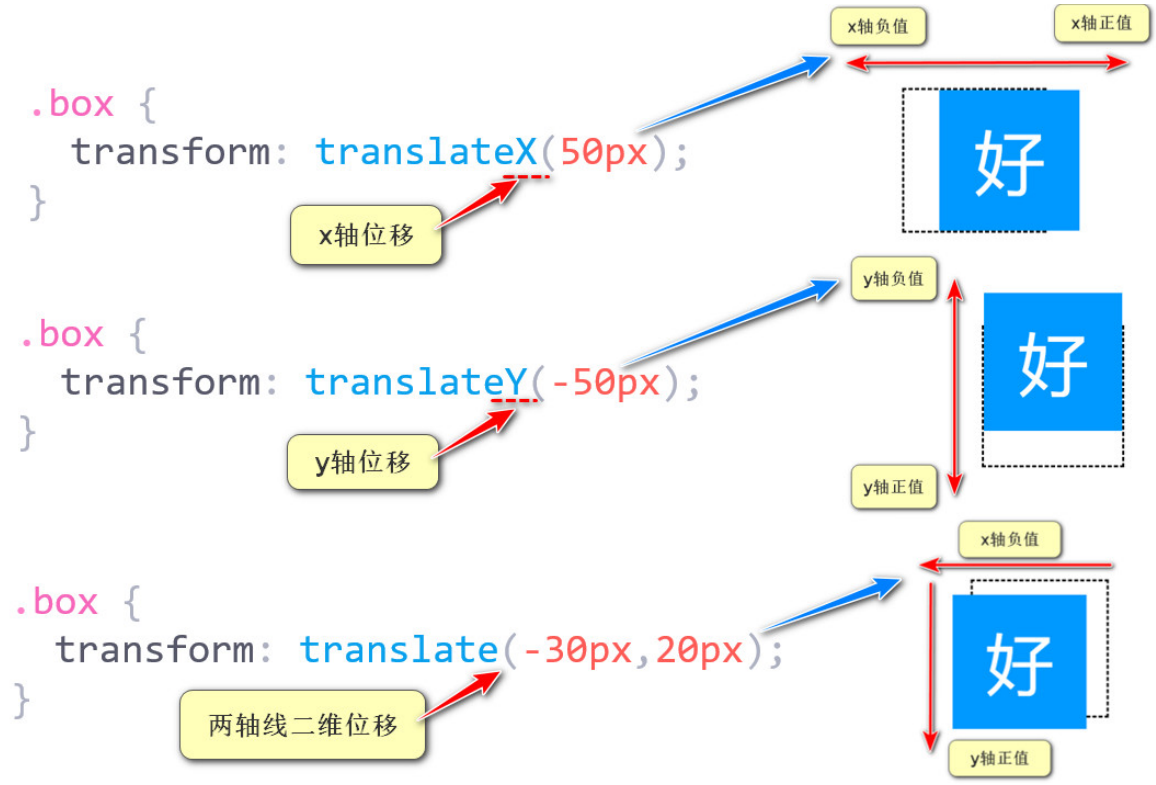
1. 变换的概念【10:46】

`transform`属性是变换属性，允许元素旋转，位移，扭曲，缩放。变换函数是变换属性的值，在函数的()中写具体的数据，执行顺序，是从左到右执行函数。

- 元素的实际尺寸和"原位置"不会发生改变
- 使用变换属性还会创建层叠上下文，简单说会出现元素堆叠
- 对内联元素，变换属性无效（可替代内容的内联元素除外img）
- 不同顺序会造成不同效果，限于多种变换函数同时使用的时候，有顺序造成不同结果
- 出现锯齿化或虚化，如果是高清屏浏览器就不会出现
- 使用变换属性，变换的元素默认参照的基础点为中心点，而不再是左上角。

2.位移函数【11：23】

`translate()`指位移函数,可以单独指定x轴或者y的位移，也可以同时指定，值需要用逗号分开，前面是x轴，后面是y轴。




```
/* x轴上的位移，正值向右，负值向左 */
transform: translateX(100px);
/* y轴上的位移，正值向下，负值向上 */
transform: translateY(100px);
/* 参数1：x轴方向，参数2：y轴方向 */
transform: translate(200px,60px);
```

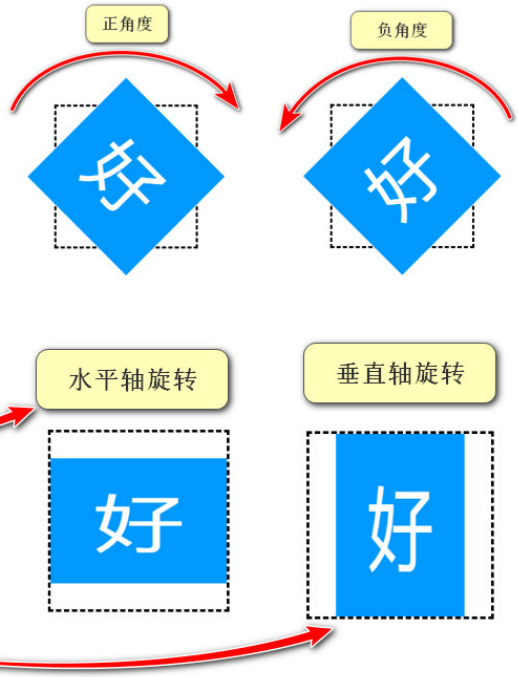
3.旋转函数【11:41】

`rotate()`函数定义了将元素围绕一个点旋转，指定的角度就是旋转的角度，若为正值，顺时针旋转。负值逆时针旋转。角度的单位都是deg。

围绕的默认参照基础点，是元素的中心点，这个点也叫"基点"，是可以改变的。

```css
.box {
    transform: rotate(45deg);
  }
.box {
  transform: rotate(-45deg);
}

.box {
  transform: rotateX(45deg);
}
.box {
  transform: rotateY(-45deg);
}
```




```
transform: rotate(60deg); z轴顺时针角度
transform: rotate(-60deg);z轴逆时针角度
transform: rotateX(60deg);x轴旋转
transform: rotateY(60deg);y轴旋转
```

练习【14:10】


```css
.wrap {
  width: 205px;
  height: 289px;
  margin: 100px auto;
  border: 1px solid blue;
  position: relative;
}
img {
  display: block;
  position: absolute;
  transition: 12s;
  /* 背面不可见 */
  backface-visibility: hidden;
}
.one {
  z-index: 1;
}
```

```html
<div class="wrap">
  <img src="../img/puke/扑克a.jpg" class="one">
  <img src="../img/puke/扑克b.jpg" class="tow">
</div>
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <style type="text/css">
            .wrap {
```

```
                width: 206px;
                height: 289px;
                border: 1px solid red;
                margin: 200px auto;
                position: relative;
            }
            .wrap img {
                display: block;
                position: absolute;
                /* 背面不可见 */
                backface-visibility: hidden;
                transition: 1s;
            }
            .one {
                z-index: 1;
            }
            .two {
                transform: rotateY(180deg);
            }
            .wrap:hover .one {
                transform: rotateY(180deg);
            }
            .wrap:hover .two {
                transform: rotateY(0deg);
            }
        </style>
        <title></title>
    </head>
    <body>
        <div class="wrap">
            <img src="./img/puke/扑克a.jpg" class="one">
            <img src="./img/puke/扑克b.jpg" class="two">
        </div>
    </body>
</html>
```

4.缩放函数【14:24】

`scale()` 允许元素使用缩放比例，比例为数值，无需单位。

（1）放大缩小效果

1默认值不缩放 大于1放大，小于1大于0就是缩小

```
/* x轴的缩放  */
transform: scaleX(0.2);
/* y轴的缩放 值同上 */
transform: scaleY(2);
/* 一个值代表两个方向 */
transform: scale(2);
/* 两个值，第一个x轴，第二个y轴 */
transform: scale(0.5,1.5);
/* 0值 缩小到没有*/
transform: scale(0);
```

（2）翻转效果

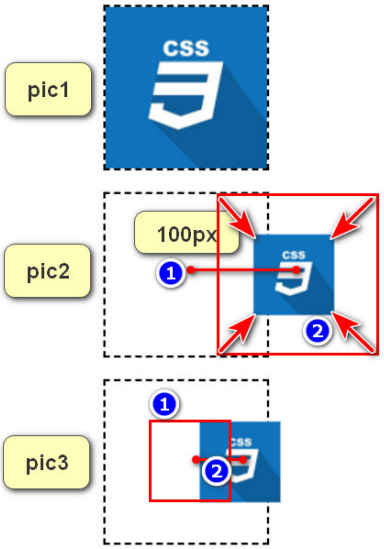
```
/* 负号指的翻转，数字管放大缩小 */
transform: scaleX(-2); x轴翻转
transform: scaleY(-1); y轴翻转
transform: scale(-1); xy轴都翻转
```

5.扭曲【14:44】

`skew()` 函数定义了一个元素在二维平面上的倾斜转换。值为角度，表示在某个方向的倾斜量。

```
transform: skewX(60deg); x轴扭曲
transform: skewY(-30deg); y轴扭曲
transform: skew(40deg); x轴扭曲
```

6.变换属性顺序【16:40】




```
<div class="box">
  <img src="logo.png" class="pic1">
</div>
<div class="box">
  <img src="logo.png" class="pic2">
</div>
<div class="box">
  <img src="logo.png" class="pic3">
</div>
```

先位移（100px），再缩小到（75px）

```
.pic2 {
  transform: translateX(100px) scale(0.5);
}
```

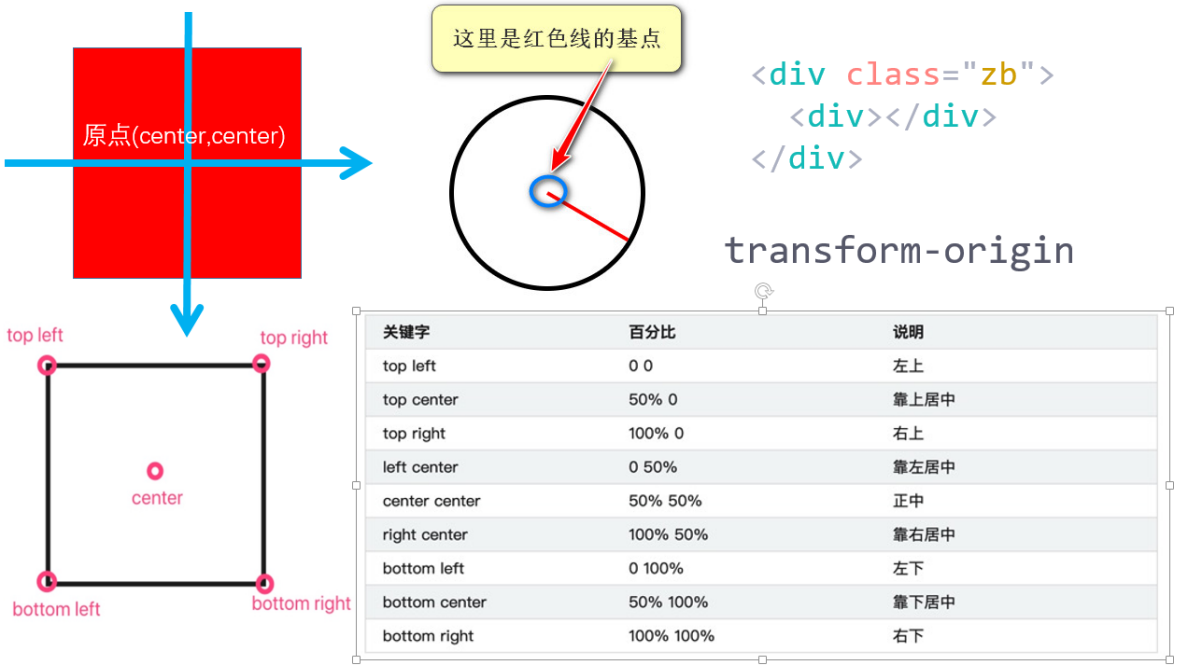
先缩小一倍，之后位移的100px的参照体系发生改变，原来大小的100px变成现在大小的50px

```
.pic3 {
  transform: scale(0.5) translateX(100px);
}
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <style type="text/css">
            div {
                width: 150px;
                height: 150px;
                border: 4px dashed #000;
                margin: 10px 0;
            }
            img {
                width: 100%;
                display: block;
            }
            .pic2 {
                transform: translateX(100px) scale(0.5);
            }
```

```
            .pic3 {
                transform: scale(0.5) translateX(100px) ;
            }
            .pic2 {
                transform: translateX(100px) rotate(60deg);
            }
            .pic3 {
                transform: rotate(60deg) translateX(100px) ;
            }
        </style>
        <title></title>
    </head>
    <body>
        <div>
            <img src="./img/csslogo/logo.png" class="pic1">
        </div>
        <div>
            <img src="./img/csslogo/logo.png" class="pic2">
        </div>
        <div>
            <img src="./img/csslogo/logo.png" class="pic3">
        </div>
    </body>
</html>
```

7.基点【16:55】



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

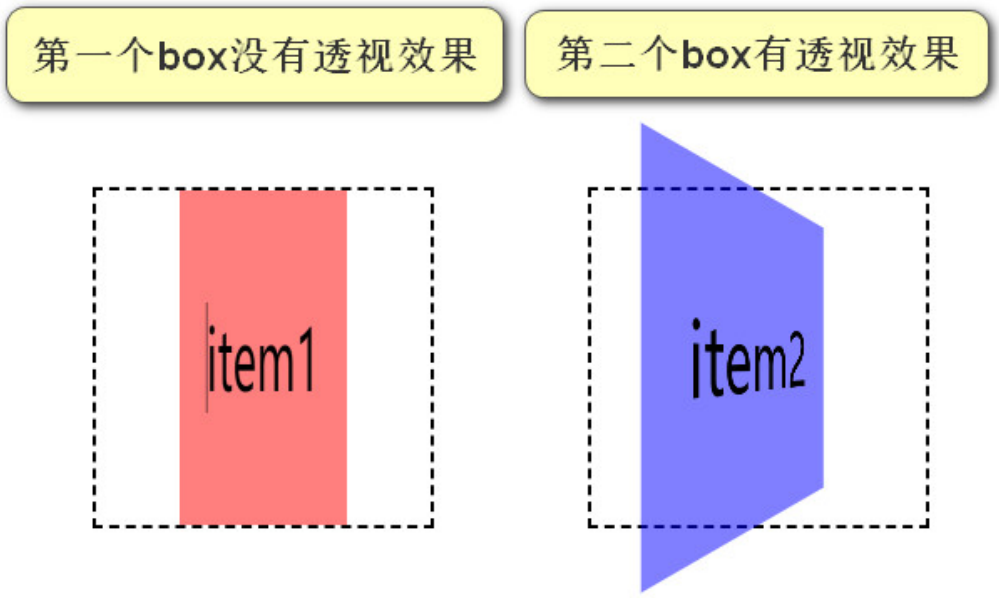
8.透视

`perspective`叫做透视，也叫作“视距”。表示视点距离屏幕的长短，视点模拟人眼的位置。

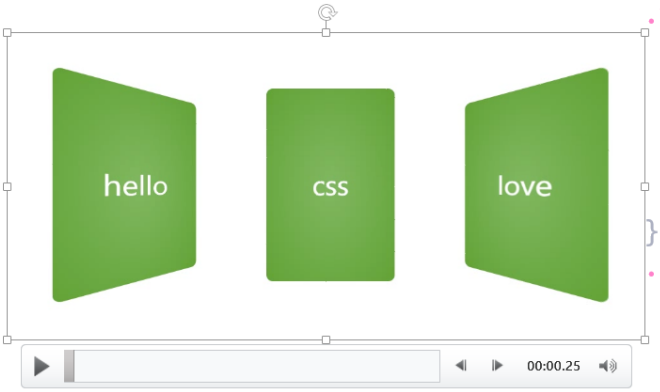
简单说就是假装眼睛和元素的距离。

透视尺寸设置的越大，相当于眼睛离元素越远（远小），反之视距设置的越小，相当于眼睛离元素越近（近大）

透视属性一定要给父级元素设置



练习【17:44】




```
<div class="wrap">
  <div class="item">hello</div>
  <div class="item">css</div>
  <div class="item">love</div>
</div>
```

```
.wrap {
  width: 800px;
  margin: 100px auto;
  display: flex;
  justify-content: space-between;
  这里要加什么属性?
}
.item {
  width: 200px;
  height: 300px;
  border-radius: 10px;
  font-size: 40px;
  color: #fff;
  text-align: center;
  line-height: 300px;
  background-image:
radial-gradient(#81B861,#62A336);
}
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <style type="text/css">
            .wrap {
                width: 800px;
                margin: 100px auto;
                display: flex;
                justify-content: space-between;
                /* 准备写透视 */
                perspective: 500px;
            }
            .item {
                width: 200px;
                height: 300px;
                border-radius: 10px;
                background-color: red;
```

```
                color: #fff;
                font-size: 50px;
                text-align: center;
                line-height: 300px;
                transition: 1s;
            }
            .item:first-child{
                transform: rotateY(60deg);
            }
            .item:last-child{
                transform: rotateY(-60deg);
            }
            .item:hover {
                transform: rotateY(0) scale(1.2);
                box-shadow: 0 0 5px 6px #666;
            }
        </style>
        <title></title>
    </head>
    <body>
        <div class="wrap">
            <div class="item"></div>
            <div class="item"></div>
            <div class="item"></div>
        </div>
    </body>
</html>
```