第六章 canvas 矩阵转换

6.1 状态的保存和恢复

canvas是可以进行变形的，但是变形的不是画布，而是ctx，ctx就是整个画布的渲染区域，整个画布在变形。我们需要在画布变形前，进行保存和恢复

- **save():保存画布 (canvas) 的所有状态**
- **restore():恢复 canvas 的状态**
- save 和 restore 方法是用来保存和恢复 canvas 状态的，都没有参数。Canvas 的状态就是当前画面应用的所有样式和变形的一个快照。

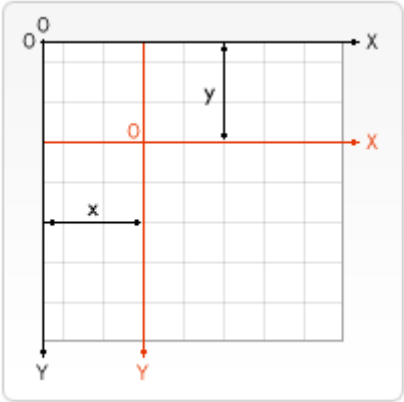
可保存的状态有：

- 当前应用的变形（即移动，旋转和缩放，见下）

  以及下面这些属性：strokeStyle, fillStyle, globalAlpha, lineWidth, lineCap, lineJoin, miterLimit, lineDashOffset, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor, globalCompositeOperation, font, textAlign, textBaseline, direction, imageSmoothingEnabled

6.2 移动 Translating

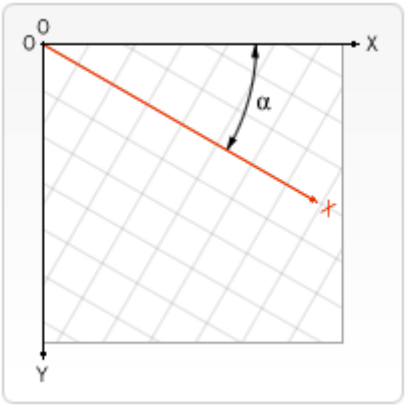
用来移动 canvas 和它的原点到一个不同的位置。



- `ctx.translate(x,y)` 方法重新映射画布上的 (0,0) 位置
- 参数说明：
  - `x`： 添加到水平坐标（x）上的值
  - `y`： 添加到垂直坐标（y）上的值
- 发生位移后，相当于把画布的 0,0 坐标 更换到新的 x,y 的位置，所有绘制的新元素都被影响。
- 位移画布一般配合缩放和旋转等。

6.2 旋转 Rotating

用于以原点为中心旋转 canvas。



- `context.rotate(angle);` 方法旋转当前的绘图

  这个方法只接受一个参数：旋转的角度 (angle)，它是顺时针方向的，以弧度为单位的值。旋转的中心点始终是 canvas 的原点，如果要改变它，我们需要用到 translate方法。

- 注意参数是弧度（PI）
- 如需将角度转换为弧度，请使用 degrees*Math.PI/180 公式进行计算。

6.3 缩放 Scaling

用来增减图形在 canvas 中的像素数目，对形状，位图进行缩小或者放大。

- `scale()` 方法缩放当前绘图，更大或更小
- 语法：

```
context.scale(scalewidth,scaleheight)
```

  - `scalewidth` : 缩放当前绘图的宽度 (1=100%, 0.5=50%, 2=200%, 依次类推)
  - `scaleheight` : 缩放当前绘图的高度 (1=100%, 0.5=50%, 2=200%, etc.) +注意：缩放的是整个画布，缩放后，继续绘制的图形会被放大或缩小。

6.4 变形 Transforms

- transform(a, b, c, d, e, f)

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

- 这个函数的参数各自代表如下：
  - a　水平方向的缩放
  - b　竖直方向的倾斜偏移
  - c　水平方向的倾斜偏移
  - d　竖直方向的缩放
  - e　水平方向的移动
  - f　竖直方向的移动

6.5 动画

canvas的主要功能是用来绘图，本身并具备动画能力，但是我们可以利用一些别的手段来实现动画，因为动画的本质是图像不断快速切换，让人的视觉产生滞留的效果而产生的，所以我们现在只要需要不断的切换图像就可以达到我们想要的效果。

不断切换的效果我们自然而然想到了定时器，但是其实一般不推荐在做动画时使用定时器来做，因为定时器其实是不精确的，所以浏览器给我们提供了一个精确儿平滑实现动画的方法requestAnimationFrame，该方法需要传入一个函数，然后形成递归调用来实现动画。

- 语法：

```
window.requestAnimationFrame(callBack);
```

requestAnimationFrame方法的回调函数会自动在浏览器更新UI的每一帧自动调用，所以我们只需要在里面对图形的状态更新就可以了。比如我们做一个方形的横移动画，我们在每帧让方形向右移动1像素。

```
var x = 0;
function move(){
    ctx.fillRect(x+=1,100,100,100);
    requestAnimationFrame(move);
}
requestAnimationFrame(move);
```

值得注意的是，requestAinimationFrame需要递归调用才会不断更新图像。

但是出乎意料的是我们的位移动画好像不是位移，而是看起来像是方形在变长，原因是上一顿画好了图像会留在在 anvas 上，不会自动消失，所以我们需要先把上一的图像先清空。清空画布使用 clearRect 方法

```
var x = 0;
function move(){
    ctx.clearRect(0,0,canvas.width,canvas.height);//清空画布
    ctx.fillRect(x+=1,100,100,100);
    requestAnimationFrame(move);
}
requestAnimationFrame(move);
```

案例：环绕动画

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        canvas{
            background-color: rgb(48, 45, 45);
            display: block;
            margin: 0 auto;
            color: rgb(231, 227, 10);
        }
    </style>
</head>
<body>
```

```
    <canvas width="800" height="600"></canvas>
    <script>
        var canvas = document.querySelector('canvas');
        var ctx=canvas.getContext('2d');

        //定义一个地球旋转角度的变量
        var a = 0;
        //定义一个月球旋转的角度
        var b = 0;
        function loop(){
            ctx.save();//保存之前的状态
            //清除之前在画布上的内容
            ctx.clearRect(0,0,canvas.width,canvas.height);
            //绘制太阳
            ctx.beginPath();
            ctx.fillStyle='rgb(231, 227, 10)';
            ctx.shadowColor="rgb(231, 227, 10)";
            ctx.shadowBlur=10;
            ctx.arc(400,300,100,0,2*Math.PI);
            ctx.fill();
            ctx.closePath();


            //地球
            ctx.beginPath();
            //先把canvas的原点移动到正中心
            ctx.translate(400,300);
            //让画笔旋转一个角度
            // ctx.rotate((a+=1)*Math.PI/180);
            ctx.rotate((a+=0.5)*Math.PI/180);
            ctx.fillStyle='blue';
            ctx.shadowBlur=0;
            // ctx.arc(200,0,20,0,Math.PI*2);
            //为了后面画月亮的动画，要把原点变成和地球中心一致
            ctx.translate(200,0);
            ctx.arc(0,0,20,0,Math.PI*2);
            ctx.fill();
            ctx.closePath();

            //月亮
            ctx.beginPath();
            //旋转
            ctx.rotate((b+=1)*Math.PI/180);
            ctx.arc(40,0,6,0,Math.PI*2);
            ctx.fillStyle='#fff';
            ctx.fill();
            ctx.closePath();

            //恢复状态
            ctx.restore();
            requestAnimationFrame(loop);//递归调用
        }

        //使用requestAnimationFrame调用函数
        requestAnimationFrame(loop);
    </script>
</body>
</html>
```