Canvas的应用（一）

canvas是Html5中的一个新功能，它可以实现在一个矩形区域内绘制一些简单图形的功能，最近很是热衷于用它去实现一些交互动画，要说如何应用它，一言以蔽之，化繁为简吧。不考虑时间的维度，在我们眼前定格的每一帧画面，纵使再精美，也无外乎点线面的结合，所以能绘制基本的点、线、面也足以实现很多；要是考虑上时间这个维度，结合视觉暂留原理，就算没有绘画基础，也能实现简单的动画交互。

一、lineTo()的应用——签名

1.原理

原理其实很简单，就是用极限的思想去看待鼠标或者手指划出的所有线条，当线条无限短，我们就可以将它视为直线，短到什么程度呢，短到由两个像素构成，或水平或垂直或对角线，所有的可能也无外乎如此，所以仅仅是lineTo()就能够实现签名了，所以监控鼠标的mousemove事件，并获取当前事件对象坐标即可。

```
canvas.addEventListener('mousemove',function(e) {
    var x = e.clientX - this.offsetLeft;
    var y = e.clientY - this.offsetTop;
    Point.push([x,y]);
    var len = Point.length;
    if(flag) {
        ct.beginPath();
        ct.moveTo(Point[len-1][0],Point[len-1][1]);
        ct.lineTo(Point[len-2][0],Point[len-2][1]);
        ct.closePath();
        ct.stroke();
    }
});
```

将经过的的所有点放在一个数组中，在当前点和上一个点之间绘制路径，签名的基本功能就能够实现。

2.在手机上的实现

需要说明的一点是，在手机上event对象会有所不同，touches属性是一个数组，会独立记录每个触控点的信息，当event.touches.length为1的时候，说明是一个触控点，
event.touches.length为2的时候就是多点触控，签名自然是单点触控，所以要获取当前坐标就要从touches［0］取得。

```
canvas.addEventListener('touchmove',function(e) {
    event.preventDefault();
    if(e.touches.length == 1) {
        var x = e.touches[0].clientX - this.offsetLeft;
        var y = e.touches[0].clientY - this.offsetTop;
        Point.push([x,y]);
        var len = Point.length;
        if(flag) {
            ct.beginPath();
            ct.moveTo(Point[len-1][0],Point[len-1][1]);
            ct.lineTo(Point[len-2][0],Point[len-2][1]);
            ct.closePath();
            ct.stroke();
        }
    }
},false);
```

3.其他说明

如果仅仅是监控move事件，那么在pc端只要鼠标滑过就会绘制路径，所以得设置一个开始绘制路径的开关，即将mousedown事件所谓开始绘制的标志，mouseup则绘制结束。 当签名完成之后，调用canvas的toDataURL()可以获取画布内的绘制信息，并以图片的形式保存下来。

当然这里其实存在一些遗留问题，如何对签名进行识别，如果需要识别，那是否应该先对签名做一些平滑处理，这一系列问题留待后续解决。

二、bezierCurveTo()的应用

1.贝塞尔曲线

关于贝塞尔曲线具体是什么，这里不做赘述，随便一搜，各种百科说的很详细。这边主要要说的是贝塞尔曲线如何在canvas中加以应用。这边主要用到的是二次贝塞尔公式，即bezierCurveTo()的应用，它有6个参数，前两个是起始点的坐标，中间两个是控制点的坐标，最后两个是终点的坐标，知道起点终点和一个控制点，就能绘制我们想要的曲线了。

2.应用

仔细观察一些网站，很多地方都用到了直线变曲线、仿波浪形状，或者一些不规则图形，能够实现它们就是贝塞尔曲线的功劳，其中很著名的就是qq的消除气泡。

```
effect:function(){
    var canvas = this.canvas,
    offset = this.offset,
    ct = canvas.getContext("2d"),
    ui = this.ui,
    x1 = ui.originalPosition.left,
    y1 = ui.originalPosition.top,
    x = ui.offset.left,
    y = ui.offset.top;

    var base = Math.sqrt((x1-x)^2+(y1-y)^2),
    sin = Math.abs(x1-x)/base,
    cos = Math.abs(y1-y)/base;

    ct.clearRect(0,0,400,200);
    ct.beginPath();
    ct.arc(x1,y1,10,0,2*Math.PI,false);
    ct.fill();

    ct.beginPath();
    ct.moveTo(x1+cos,y1-sin);
    ct.lineTo(x1-cos,y1+sin);
    ct.bezierCurveTo(x1-cos,y1+sin,(x1+x)/2,(y1+y)/2,x-cos,y+sin);
    ct.lineTo(x+cos,y-sin);
    ct.bezierCurveTo(x+cos,y-sin,(x1+x)/2,(y1+y)/2,x1+cos,y1-sin);
    ct.fill();

    ct.beginPath();
    ct.arc(x,y,10,0,2*Math.PI,false);
    ct.fill();
},
```

这里贴出部分关键的代码，就是拖拽那个气泡出现的拉伸效果，如同拉扯一个有弹性的东西，被拉的越长就会越细，在bezierCurveTo()通过设置合理的控制点就能很好的实现这个效果。其他更多的应用，等有空把代码写出来再继续咯。

张玥

2016.7.4凌晨