# Canvas是基于状态绘制；

**先写状态语句，后写绘制语句：**

**//状态语句**

**cxt.fillStyle = "yellow";**

**cxt.strokeStyle = "black";**

**//绘制语句**

**cxt.fill();**

**cxt.stroke();**

# ClosePath()

# CanvasRenderingContext2D.closePath() 是 Canvas 2D API 将笔点返回到当前子路径起始点的方法。它尝试从当前点到起始点绘制一条直线。 如果图形已经是封闭的或者只有一个点，那么此方法不会做任何操作。

**beginPath()**

**CanvasRenderingContext2D.beginPath() 是 Canvas 2D API 通过清空子路径列表开始一个新路径的方法。 当你想创建一个新的路径时，调用此方法。**

**绘制需要描边的填充颜色的封闭多边形，要先填充颜色后描边：**

**cxt.fill();**

**cxt.stroke();**

**save表示保存save函数之前的状态，restore表示获取save保存的状态**

**开始绘制一个图形调用save()**

**绘制完成一个图形后应调用restore()**

**cxt.save();**

**cxt.restore();**

**save()应写在新的状态语句之前，保存新状态之前canvas的状态,**

**要回复之前的状态是调用restore()**

**save()和restore()应该成对出现**

const ctx = wx.createCanvasContext('canvas')

ctx.save() //保存初始状态

//新状态

ctx.setFillStyle('red')

ctx.scale(2, 2)

ctx.fillRect(10, 10, 150, 100) // 红色矩形一

//回复初始状态

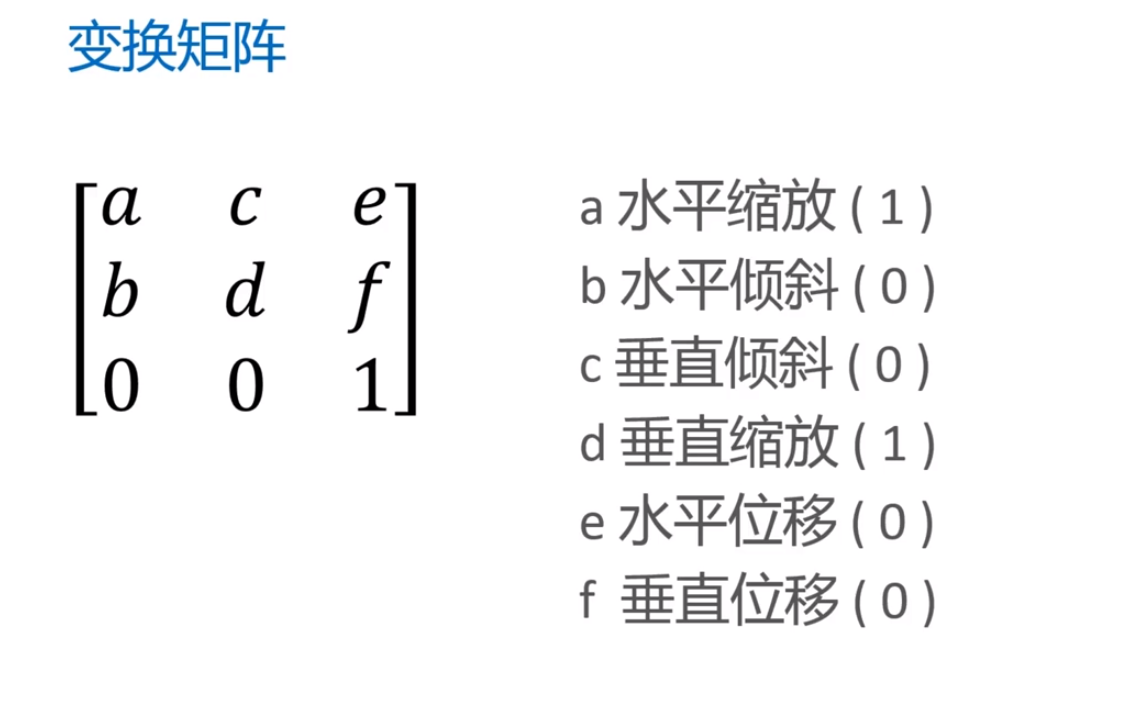
ctx.restore()

ctx.fillRect(50, 50, 150, 100) // 黑色矩形二仍为默认状态

ctx.draw()

**cxt.scale(sx,sy)会对图形的所有数值型数据全部放大或缩小：如左上角坐标，**

**边框宽度等**

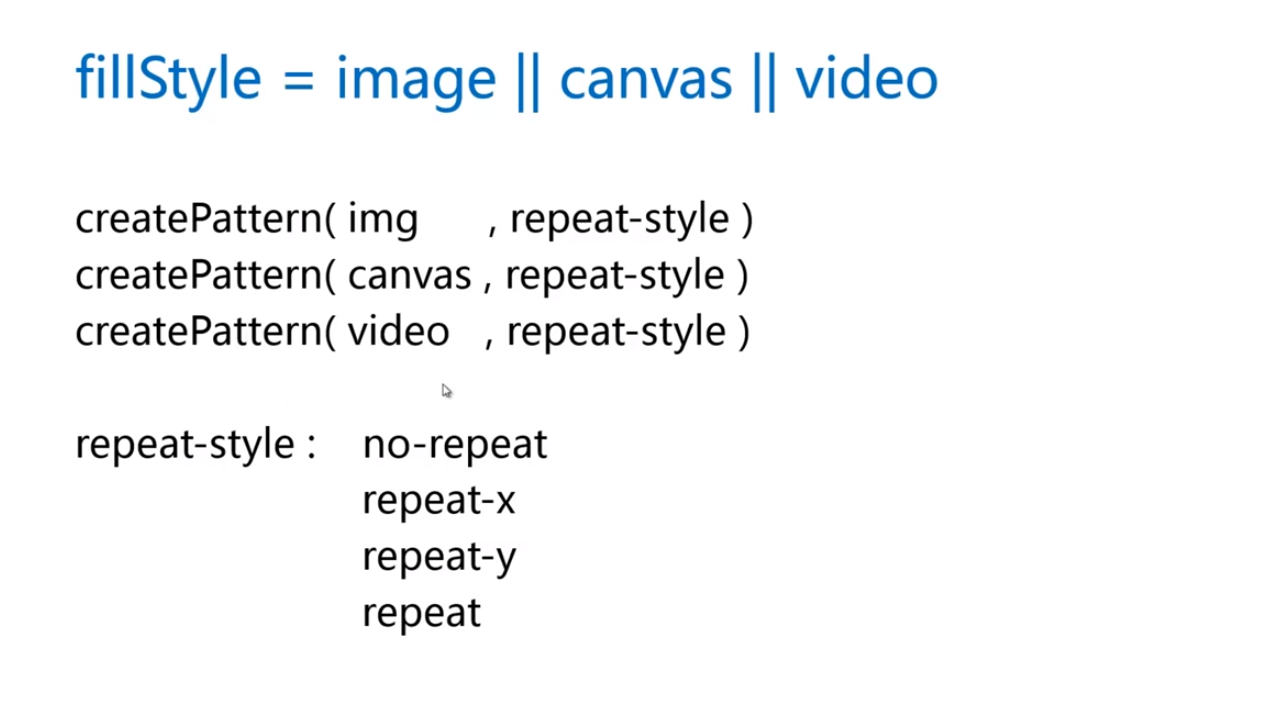


**transform(a,b,c,d,e,f)**

**setTransform(a,b,c,d,e,f)**

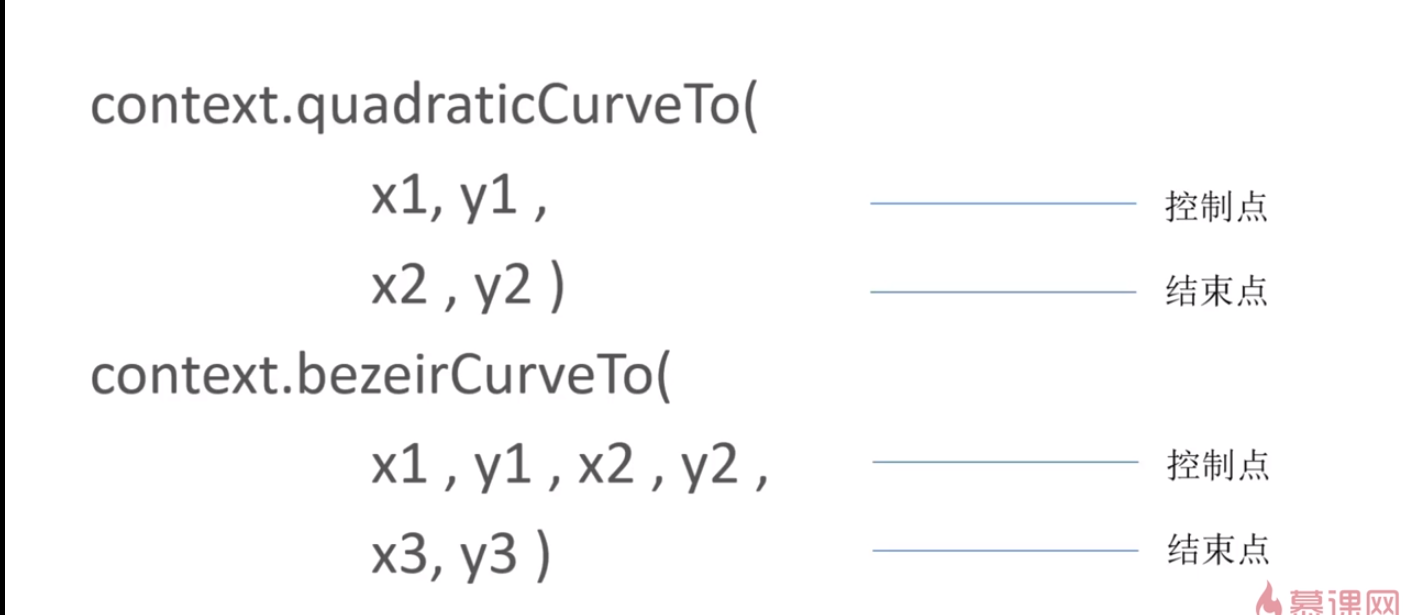
**setTransform会忽略之前所有的transform，以setTransform设置的值变换图形**

**createPattern()**



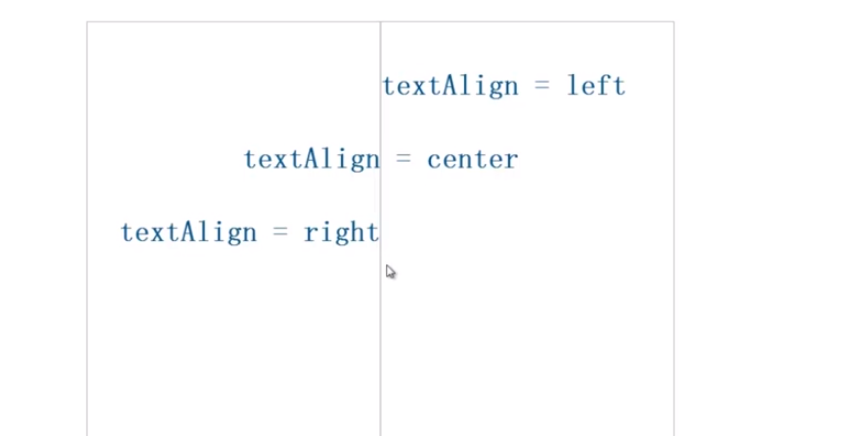
**Img为Img对象:var img = new Image();**

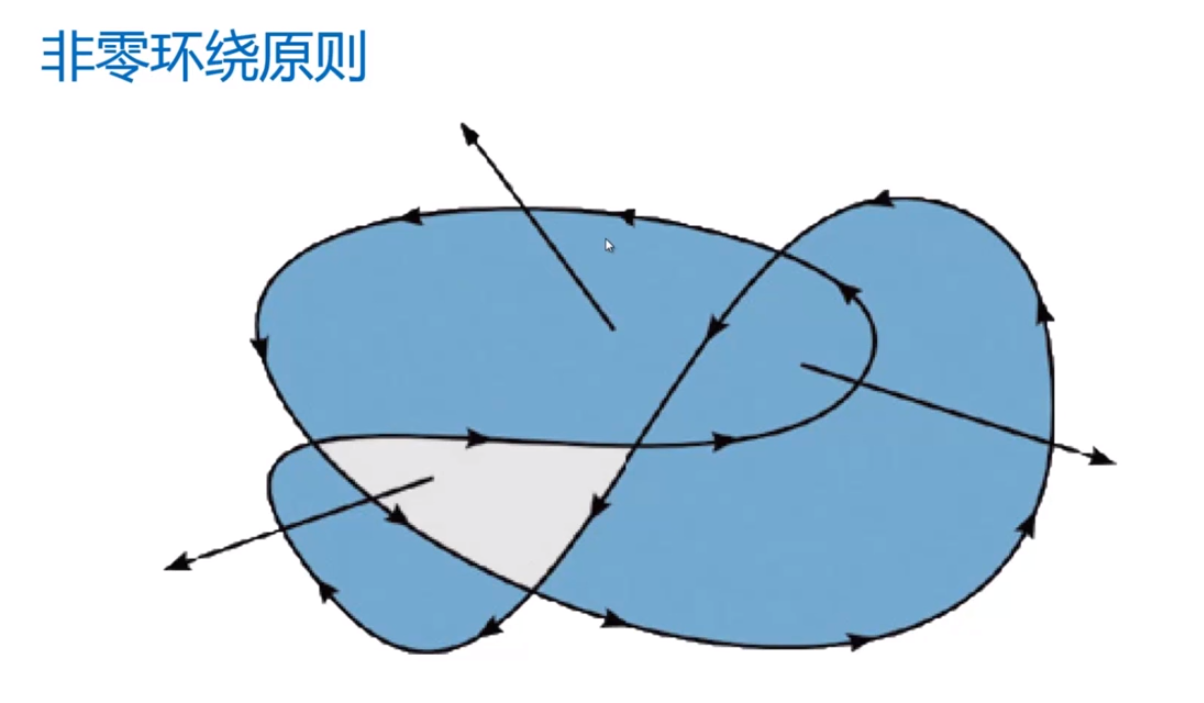
**arcTo(x1,y1,x2,y2):使用arcTo()时要先使用moveTo(x0,y0)定位到初始点位置**



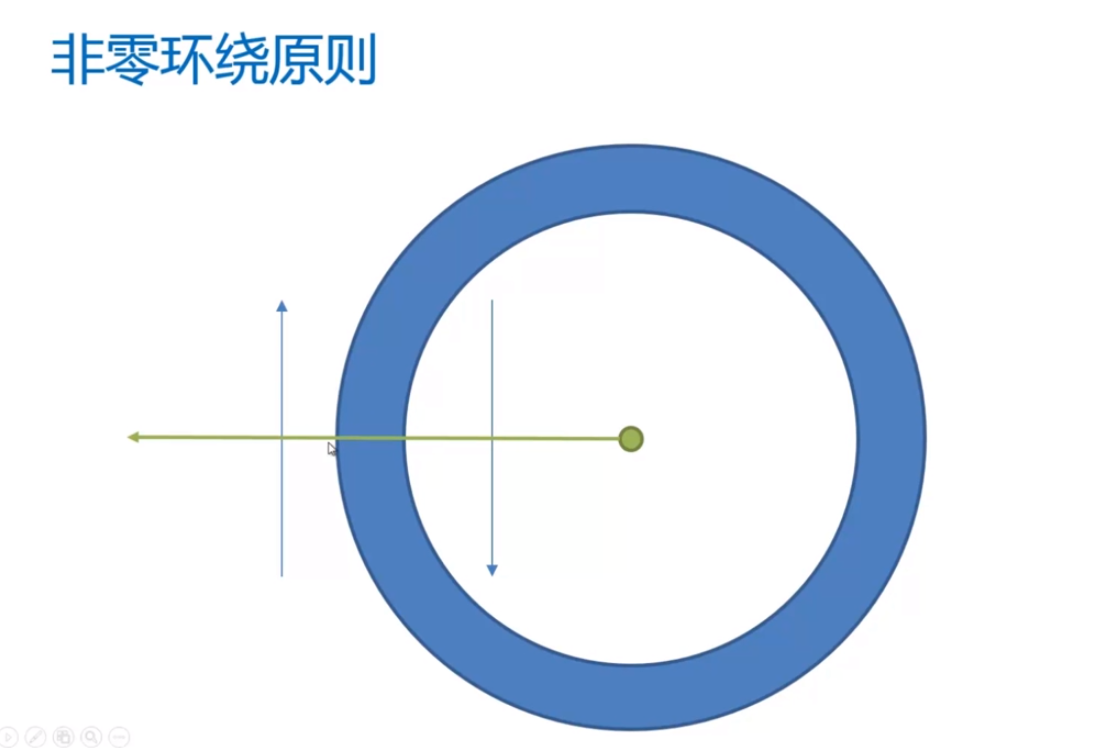
**贝塞尔二次曲线与贝塞尔三次曲线**

**textAlign**





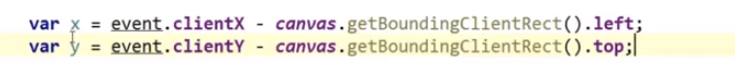
**规定一个方向为1另一个方向为负一，若存在一条射线，按上述规则进行加减运算，若结果为非0，则在图形里面，若为0则在外面**

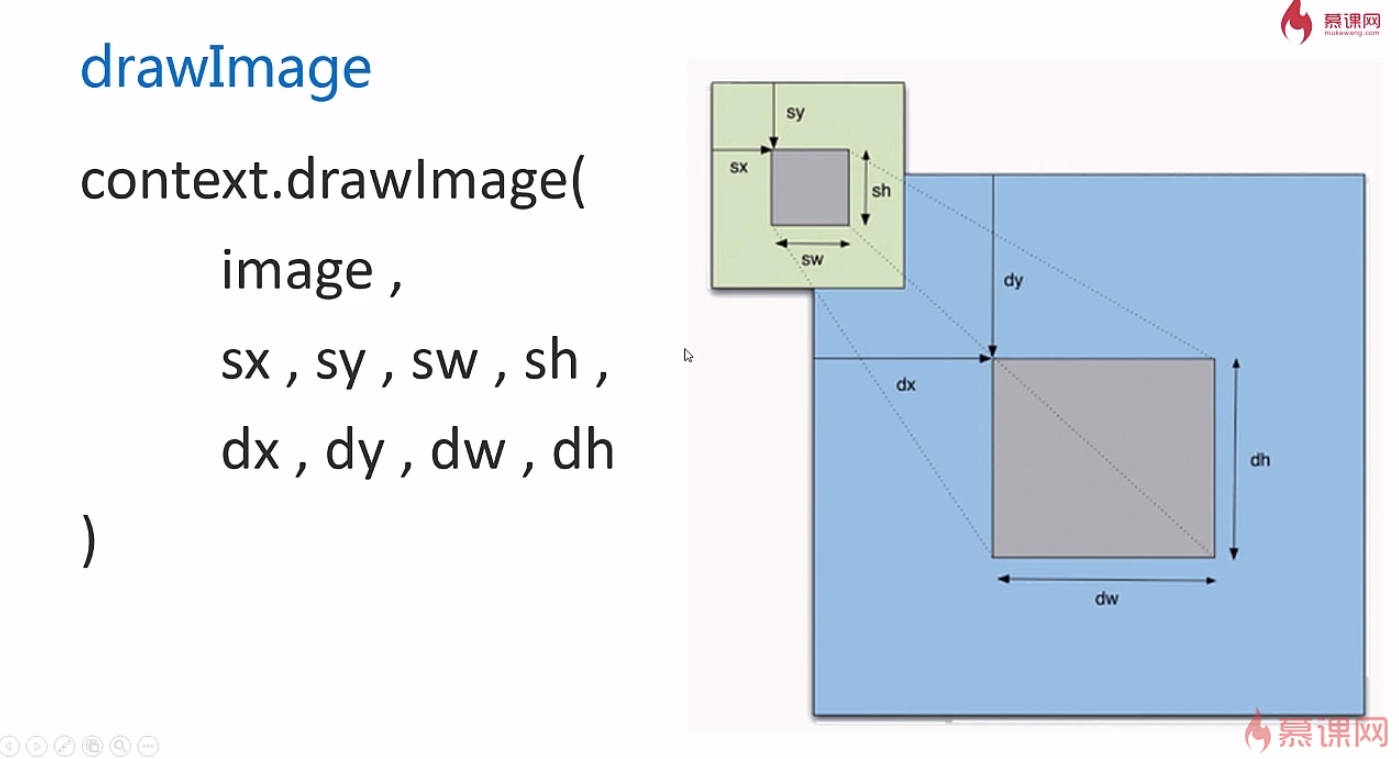


**Eg:圆弧**

**内部的射线计算值为0，则不染色，所以绘制两端圆，使其绘制方向不同即可绘制圆弧**

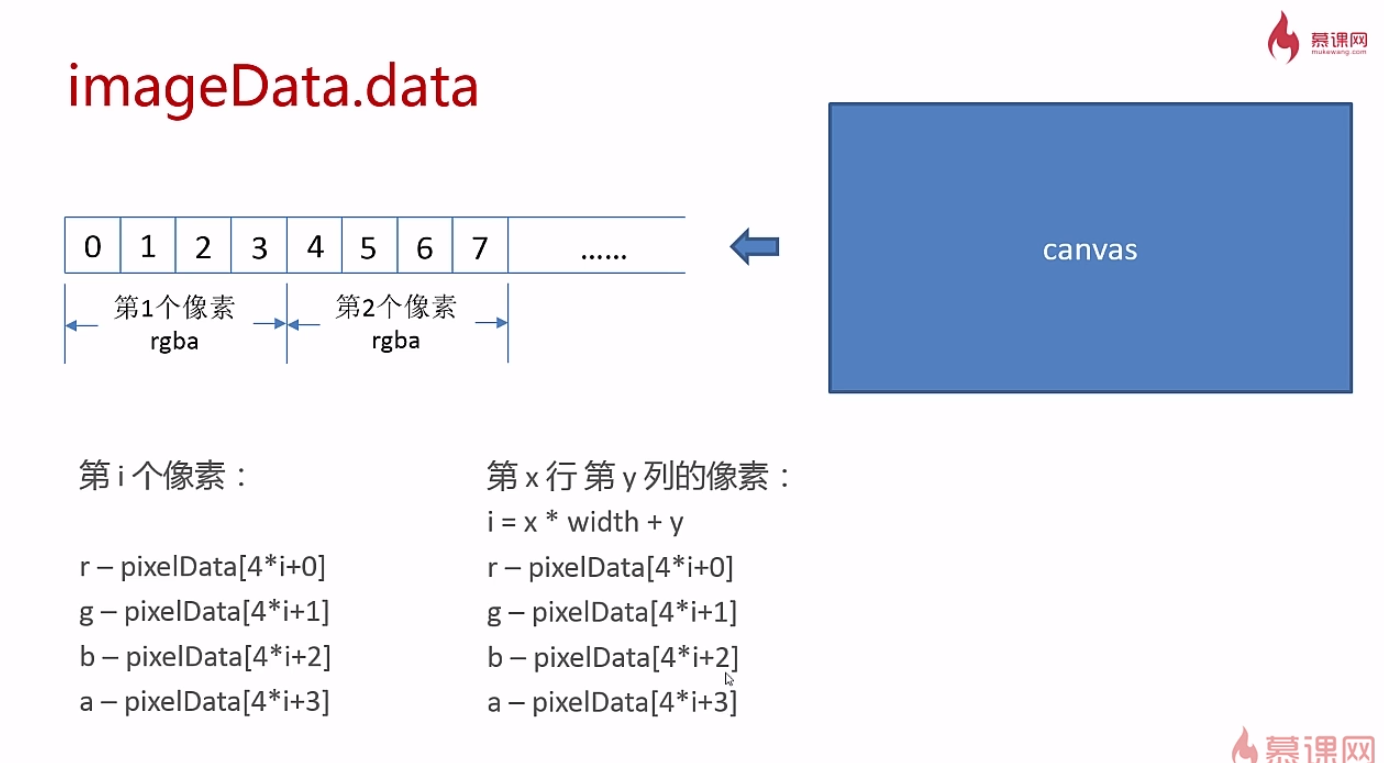
**在Canvas中获取鼠标点击位置**





**将原图像上（sx,sy）位置，sw\*sh大小的部分，绘制到画布的（dx,dy）位置，dw\*dh大小**

**离屏canvas:将一个canvas加载到另一个canvas上**



**imageData.data获取所有像素的一维数组，且每4个数据代表一个像素**

**绘制虚线：**

**context.setLineDash([a,b]);//a为虚线线段长度,b为虚线线段间隔长度**