# canvas

1. 标签canvas
2. **直接使用属性** width=” 600 ” height=” 600 ” 设置画布宽高（注意不带单位），**不要使用css**（包括style），避免拉伸
3. 若浏览器不支持canvas，会显示canvas标签包裹的文本
4. Js操作
5. 获取canvas
6. 获取canvas的上下文 var ctx=画布节点**.**getContext( ‘2d’ ); //3D需填入的参数为 ’webg1’
7. 绘制图形

* 绘制直线 i. 获取画笔的起点 ctx.moveTo( x , y )

ii. 画一条直线路径到终点，并将终点更改为下一步的起点 ctx.lineTo(x,y)

iii. 闭合路径 ctx.closePath() 连接终点于最开始的店

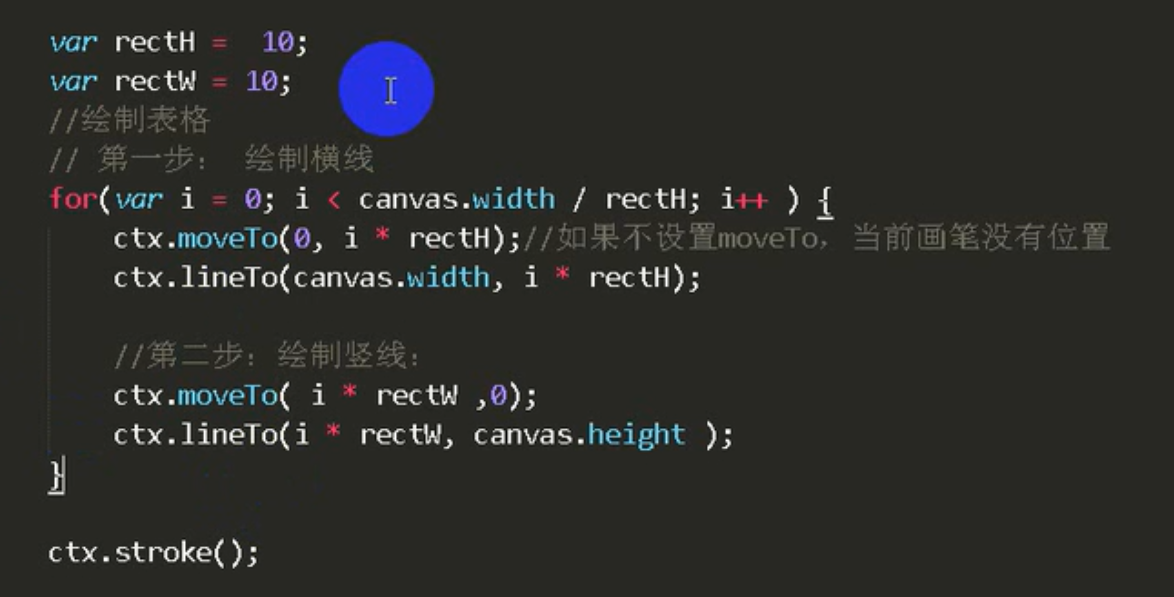
iv. 填充出实线 ctx.stroke()

v. 实线颜色 ctx.strokestyle = ‘ 颜色值 ’ 必须放在stroke之前

vi. 线宽 ctx.lineWidth = 一半为内线宽，一半为外线宽

* 填充 i. 填充闭合曲线 ctx.fill()

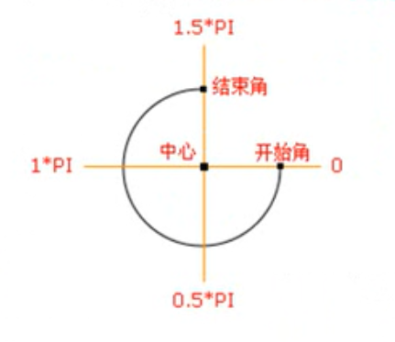
ii. 填充颜色 ctx.fillstyle = ‘ ’ 写在fill前

* 绘制网格 
* **ctx.beginPath();** 开启一个新的状态，该状态会**继承上一个状态**的样式，也可以自己设置样式，并且使得**接下来设置的样式不会影响之前的样式**。
* 绘制矩形 i. ctx.rect( **x , y , width , height** );

ii. ctx.strokeRect( x , y , w , h ) 绘制矩形并描边

iii．ctx.fillRect(x,y,w,h) 绘制矩形并填充

iv. ctx.clearRect(x,y,w,h) 清除矩形范围内的内容

* 绘制圆形 i. ctx.**arc** ( x , y , startAngle , endAngle , counterclockwise )

绘制弧度，xy为圆心坐标

**或者-0.5\*PI**

Angle均使用弧度制

弧度角度公式 **角度\*Math.PI/180**

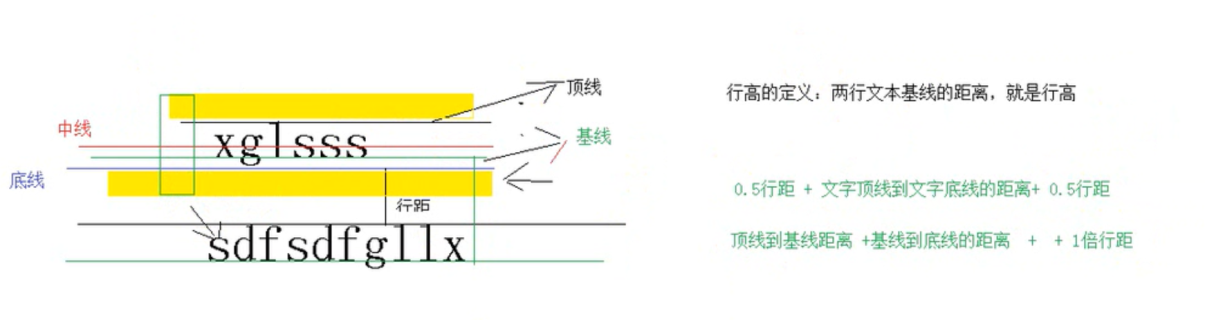
**Math.sin(角度\*Math.PI/180) 0 此处的值被自动设置为负数**

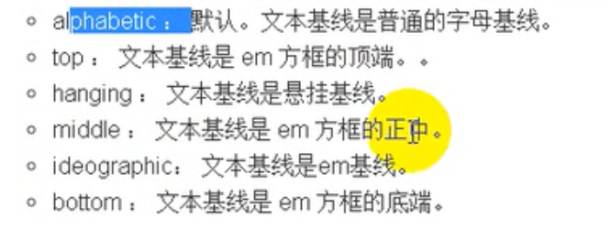
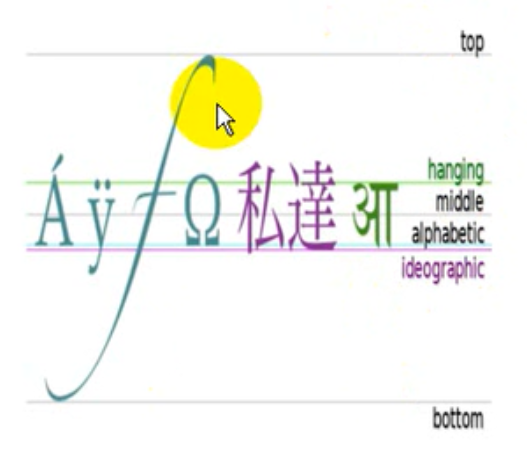
counterclockwise为绘制方向

默认false顺时针，true为逆时针

* 绘制文字 i. ctx.font= ’ 大小px “字体” ’

ii. ctx.textBaseline = “ bottom top middle等” 设置字体底线 于绘制基线的对齐方式





iii. ctx.textAlign = ” left right start end center等 ” 设置字体与基线的对齐方式

iv. ctx.strokeText ( “文字内容” , x ,y ) 描边字体

v. ctx.fillText( “文字内容” , x , y ) 填充字体

* 绘制图片

1. 创建图片dom对象 var img = new Image ( ) ;

或者 var img=document.getElementBy…

1. 设置图片的相对路径属性 img.src= “ ”
2. 图片加载完成事件 img.onload = function(){
3. 将图片绘制到canvas上 **ctx.drawImage( img对象 , x , y , width , height );** width height可省略，默认为图片原始宽高 x y为绘制位置的左上角坐标
4. 图片裁切并绘制 ctx.drawImage( img , sx , sy , swidth ,sheight , x , y ,width , height ) sx sy 为截取部分的左上角相对于图片的坐标 swidth sheight为截取部分的宽高

* 制作帧动画 setInterval( function(){

**ctx.clearRect(0,0,canvas.width,canvas.height) 清除残留帧**

**canvas.width=canvas.width 重设画布宽高，亦会清空画布**

ctx.draw( img , frameIndex\*40 , …… )

frameIndex++

frameIndex%=总个数 实现循环播放

} , 1000 / 10 )

**1秒 10帧**

1. }