基线：小写字母x的底

- **ctx.lineWidth**：设置线宽
  - 默认线宽为1px，但canvas画布中展现的却是2px，颜色也变淡，不是纯黑
  - 屏幕中展示内容最小单位是像素，所以最小展示也是1像素，但通过canvas绘制时，是以当前坐标为中心点，分别向两边扩展线宽的一半（**如果线宽为1px，则应该分别扩展0.5px，但因为最小单位为1px，所以只能分别扩展1px，将原本的1px展示为2px，并且使颜色变浅，但如果线宽为偶数，则没有影响**）

```
// 正常情况（整数坐标），绘制出来的 奇数宽度的线，最终会多1像素
// 比如：绘制一条 9像素宽的线，最终展示出来是：10像素

// 就想绘制一条真正1像素宽的线，该怎么绘制？？？
// 绘制1像素宽的线，从 0.5 开始绘制即可！！！

ctx.moveTo(100, 10.5);
ctx.lineTo(200, 10.5);
ctx.stroke();
```

画圆

弧度：radian

- **ctx.arc（x,y,半径,起始弧度startRadian，结束弧度，绘制方向）**：
  - 绘制方向 可省略不写，默认为顺时针/false

```
弧度和角度的关系：
数学中 1弧度 表示 180度
数学中用 π 来表示弧度

1π = 180°
```

```
// Math.sin() / Math.cos() 参数也是弧度

// 角度转弧度公式：（已知角度求弧度）
// 1 弧度 = 180 度
// 假设角度为：angle，弧度为：radian
// angle / 180 = radian / π
//
// radian = angle / 180 * π

// 弧度转角度公式：（已知弧度求角度）
// angle = radian / π * 180
```

```
// 将角度和弧度相互转化封装为函数：
var toRadian = function( angle ) {
  return angle / 180 * Math.PI;
};
var toAngle = function( radian ) {
  return radian / Math.PI * 180;
};
```

坐标为：向右是0度，向下是90度

```
// 该方法也是在描绘路径
ctx.arc(100, 100, 80, toRadian(0), toRadian(90));
ctx.stroke();
```

π：Math.PI

```
// 绘制整圆
ctx.arc(100, 100, 80, 0, Math.PI * 2);
// ctx.arc(100, 100, 80, toRadian(0), toRadian(360));
ctx.stroke();
```

动画的圆：

```
var toRadian = function(angle){
    return angle/180*Math.PI;
};
var toAngle = function(radian){
    return radian/Math.PI*180;
};
var cv = document.getElementById("cv");
var ctx = cv.getContext('2d');

var step = 3;
var startAngle=-90;
var timer = setInterval(function(){
     if(startAngle >= 270-step){
         clearInterval(timer);
     }
    if(startAngle >= 270){
        clearInterval(timer);
        return;
    }
    ctx.arc(100,100,100,toRadian(startAngle),toRadian(startAngle+=step));
    console.log(startAngle);
    ctx.stroke();
},50)
```

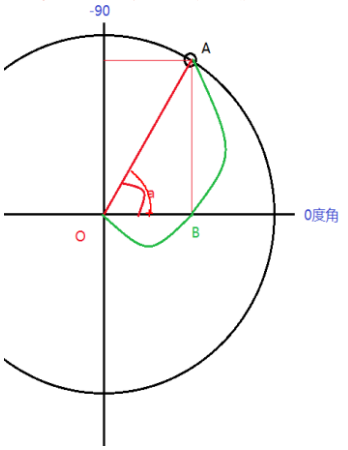
画扇形：

需要先moveTo，不然fill无法连出扇形的阴影

```
// 绘制扇形
// 需要先将起始点坐标移动到圆心点才行！
ctx.moveTo(100, 100);
// 描绘圆弧的路径
ctx.arc(100, 100, 80, toRadian(0), toRadian(90));
// ctx.stroke();
ctx.fill();
```

饼形图：数据可视化

cos和sin里面要的也是弧度



已知，O点坐标 (x0, y0)
求 A点坐标？？
(x0 + OB, y0 - AB)

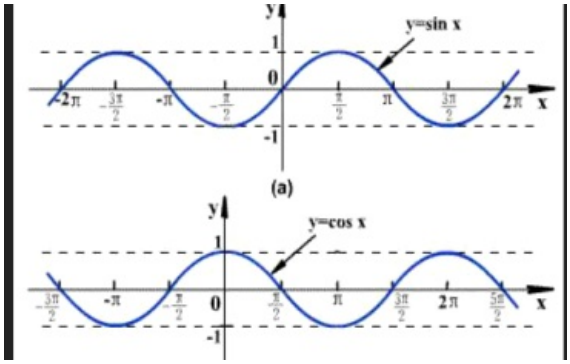
已知角度 a ，求 OB 和 AB
对边 / 斜边 = sin(a)
邻边 / 斜边 = cos(a)

对边 (AB) = r * sin(a)
邻边 (OB) = r * cos(a)

此时 A点坐标：
(x0 + r * cos(a), y0 - r * sin(a))

将 sin 和 cos 的正负考虑进来后，最终得到的公式为：

(x0 + r * cos(a), y0 + r *sin(a))



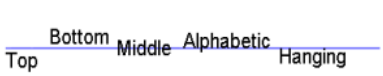
绘制文字：

注意：ctx.font一定是 字号写在前面，然后是字体

1. strokeText ( )
2. fillText()

- **获取文字的宽度**
  - **ctx.measureText('要计算的文字')**：有返回值，是对象，对象里有width属性。
- css中设置canvas的direction,可以改变文字的对齐方向
- **文字对齐方式**

- **垂直对齐：ctx.textBaseline**



- **水平对齐：ctx.textAlign = 'left/right' ---默认值是start**


```
// 文字对其方式
//
// 文字垂直对其方式
// ctx.textBaseline

// 文字的水平对齐方式
// ctx.textAlign 默认值为：start，即：默认为左对齐
ctx.fillText('水平对其方式', 100, 100);

// 让文字右对齐
ctx.textAlign = 'right';
ctx.fillText('测试文字', 100, 150);
```

```
// 有两个绘制文字的方法：
// 1 strokeText( txt )
// 2 fillText( txt )

ctx.font = '30px 微软雅黑';

// 绘制描边文字（空心）
// ctx.strokeText('要绘制的文字', 100, 100);

// 绘制填充文字（实心）
ctx.fillText('要绘制的文字', 100, 100);

//计算文字的宽度
var ret = ctx.measureText('要绘制的文字')
console.log( ret.width );
```

绘制图片：

一般情况，对于复杂图像，直接绘制图片效率更高！

ctx.drawImage（ ）-----参数只能选择写 3 / 5 / 9 个

3个：

```
// 第一个参数：表示一个图片对象
// 第二个参数：表示要将这个图片绘制到画布中的x坐标
// 第三个参数：表示要将这个图片绘制到画布中的y坐标
// ctx.drawImage(image, dx, dy)

// 绘制图片的步骤：
// 1 创建图片对象
// 2 设置图片路径
// 3 绑定load事件
// 4 在事件处理程序中绘制图片

// 怎么创建图片对象？？？
/*var img = document.createElement('img');
img.src = 'imgs/1.jpg';*/

var img = new Image();
img.src = 'imgs/3.jpg';

// 绘制图片的时候，需要等到 图片加载完毕以后，再绘制！
img.onload = function() {
  // load 事件的回调函数执行了，就表示图片加载完毕
  ctx.drawImage(img, 0, 0);
};
```

5个：

```
// 第一个参数：表示一个图片对象
// 第二个参数：表示要将这个图片绘制到画布中的x坐标
// 第三个参数：表示要将这个图片绘制到画布中的y坐标
// 第四个参数：表示绘制到画布中的宽度
// 第五个参数：表示绘制到画布中的高度
// ctx.drawImage(image, dx, dy, dW, dH)

var img = new Image();
img.src = 'imgs/2.jpg';

img.onload = function() {
  console.log( img.width )
  console.log( img.height )
  // 图片的宽度 / 图片的高度 = 绘制宽度 / 绘制高度
  // 图片的高度 / 图片的宽度 = 绘制高度/ 绘制宽度
  ctx.drawImage(img, 0, 0, 600, img.height/img.width*600);
```

等比例写法：图片宽高比 和 绘制的宽高比

9个：

```
// 第一个参数：表示一个图片对象
// 第二个参数：剪裁图片的起始点x坐标
// 第三个参数：剪裁图片的起始点y坐标
// 第四个参数：剪裁图片的宽度
// 第五个参数：剪裁图片的高度
// 第六个参数：表示要绘制到画布中的x坐标
// 第七个参数：表示要绘制到画布中的y坐标
// 第八个参数：表示要绘制到画布中的宽度
// 第九个参数：表示要绘制到画布中的高度
// ctx.drawImage(image, imgX, imgY, imgW, imgH, canvasX, canvasY,
canvasW, canvasH)

var img = new Image();
img.src = 'imgs/2.jpg';

img.onload = function() {
  ctx.drawImage(img, 710, 310, 150, 110, 100, 100, 150 * 2, 110 * 2);
};
```

```
ctx.drawImage(img, frameIndex++ * imgW
  2, imgH);

// 0 % 4 = 0
// 1 % 4 = 1
// 2 % 4 = 2
// 3 % 4 = 3
// 4 % 4 = 0
// frameIndex = frameIndex % 4;
frameIndex %= 4;

// frameIndex++;
/*if( frameIndex > 3 ) {
  frameIndex = 0;
}*/
```

优化
