

# HTML5程序设计基础

## 第十一章 画布（二）



# 主要内容

---

01 线条的样式

02 画布转换和状态保存

03 文字的渲染

04 阴影

01

# 线条的样式



# 线条样式

---

🎨 **lineWidth** 属性 —— 设置当前线条的宽度，以像素计

🎨 **lineCap** 属性 —— 设置线条末端线帽的样式

```
context.lineCap="butt |round |square";
```

值	描述
butt	默认。向线条的每个末端添加平直的边缘。
round	向线条的每个末端添加圆形线帽。
square	向线条的每个末端添加正方形线帽。

# lineCap

---



```
context.lineCap = "butt";
```

```
context.lineCap = "round";
```

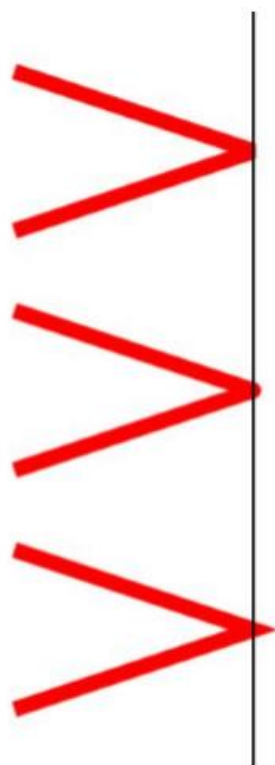
```
context.lineCap = "square";
```

"round" 和 "square" 会使线条略微变长

[demo11-1.html](#)

# lineJoin

🎨 lineJoin 属性 —— 设置当两条线交汇时所创建边角的类型



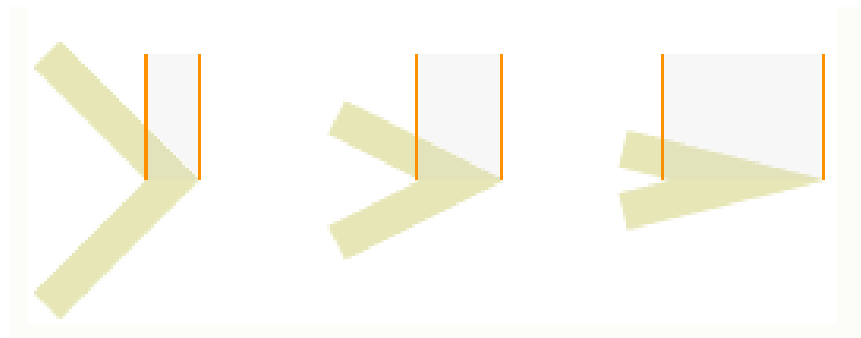
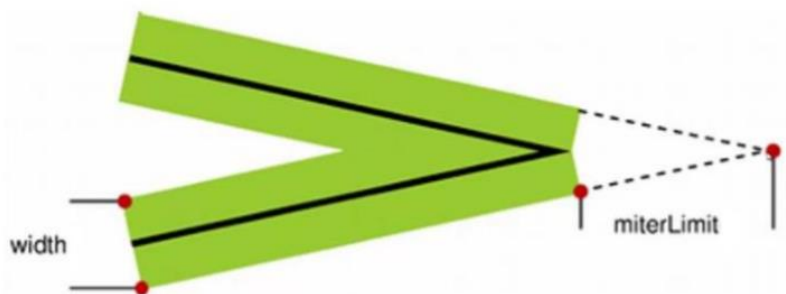
```
context.lineJoin="bevel |round |miter";
```

值	描述
bevel	创建斜角。
round	创建圆角。
<b>miter</b>	默认。创建尖角。

# miterLimit

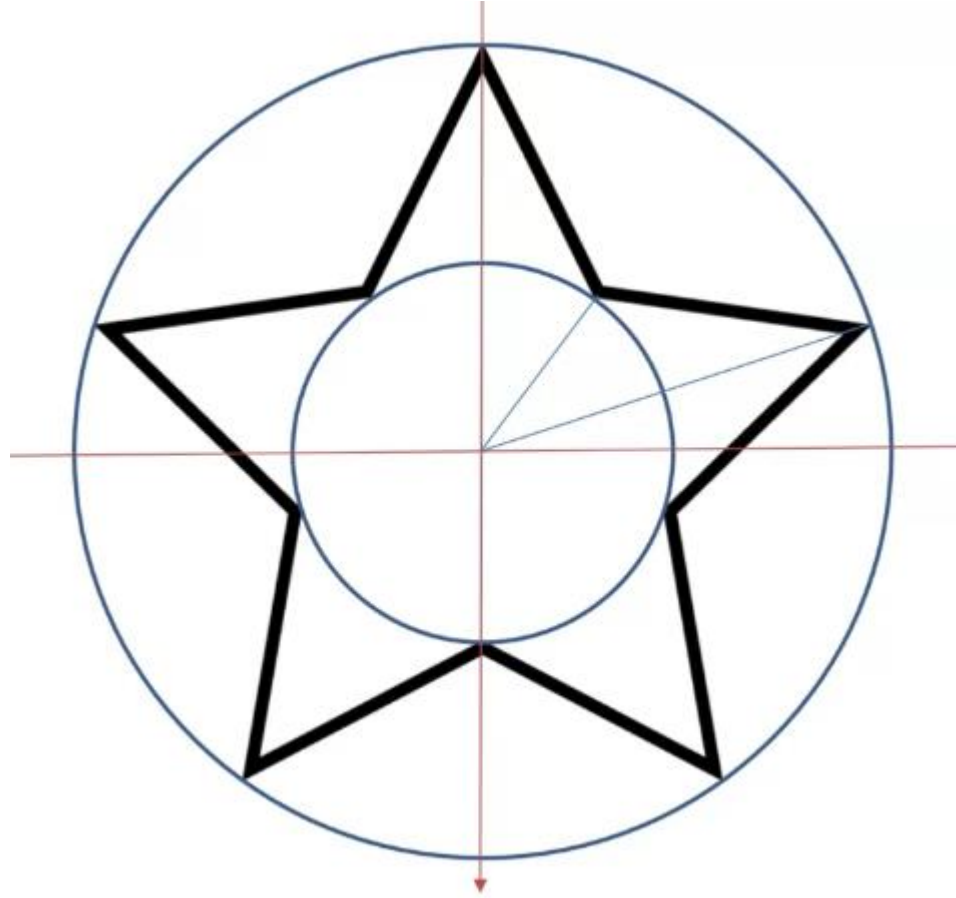
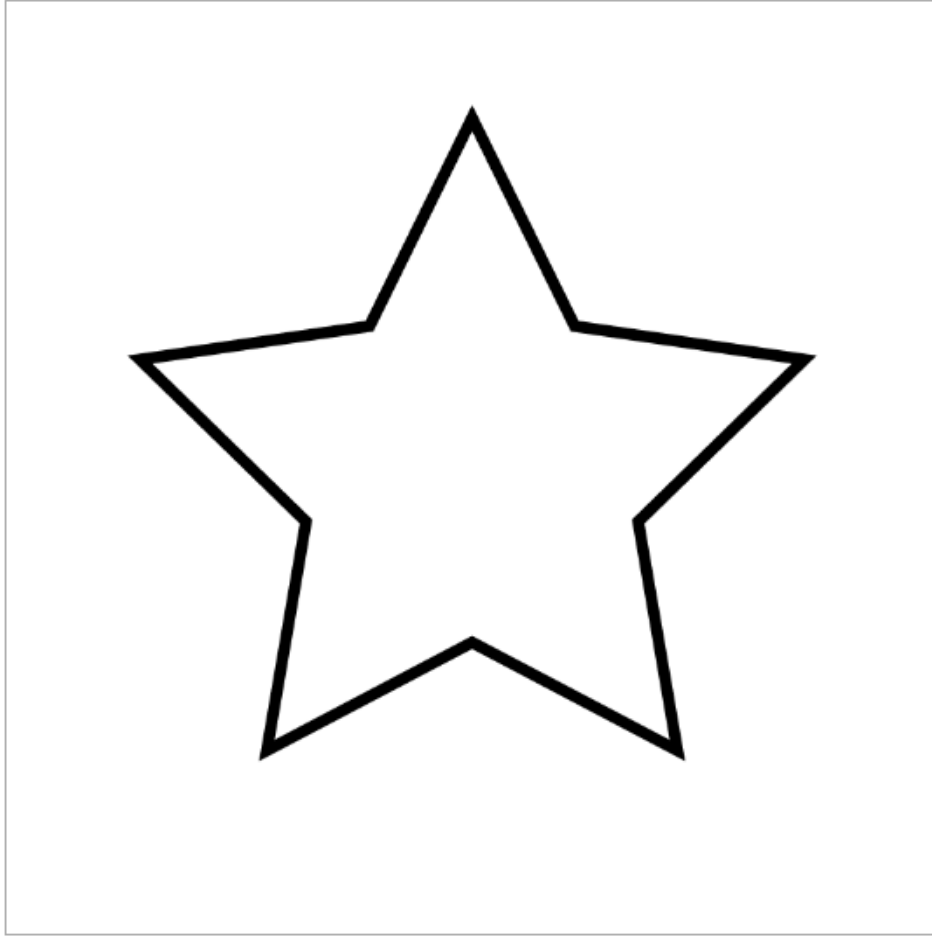
🎨 **miterLimit** 属性 —— 设置最大斜接长度，默认值为10

- 斜接长度指的是在两条线交汇处内角和外角之间的距离。
- 只有当 `lineJoin` 属性为 "miter" 时，`miterLimit` 才有效。
- 边角的角度越小，斜接长度就会越大。
- 如果斜接长度超过 `miterLimit` 的值，边角会以 `lineJoin` 的 "bevel" 类型来显示。



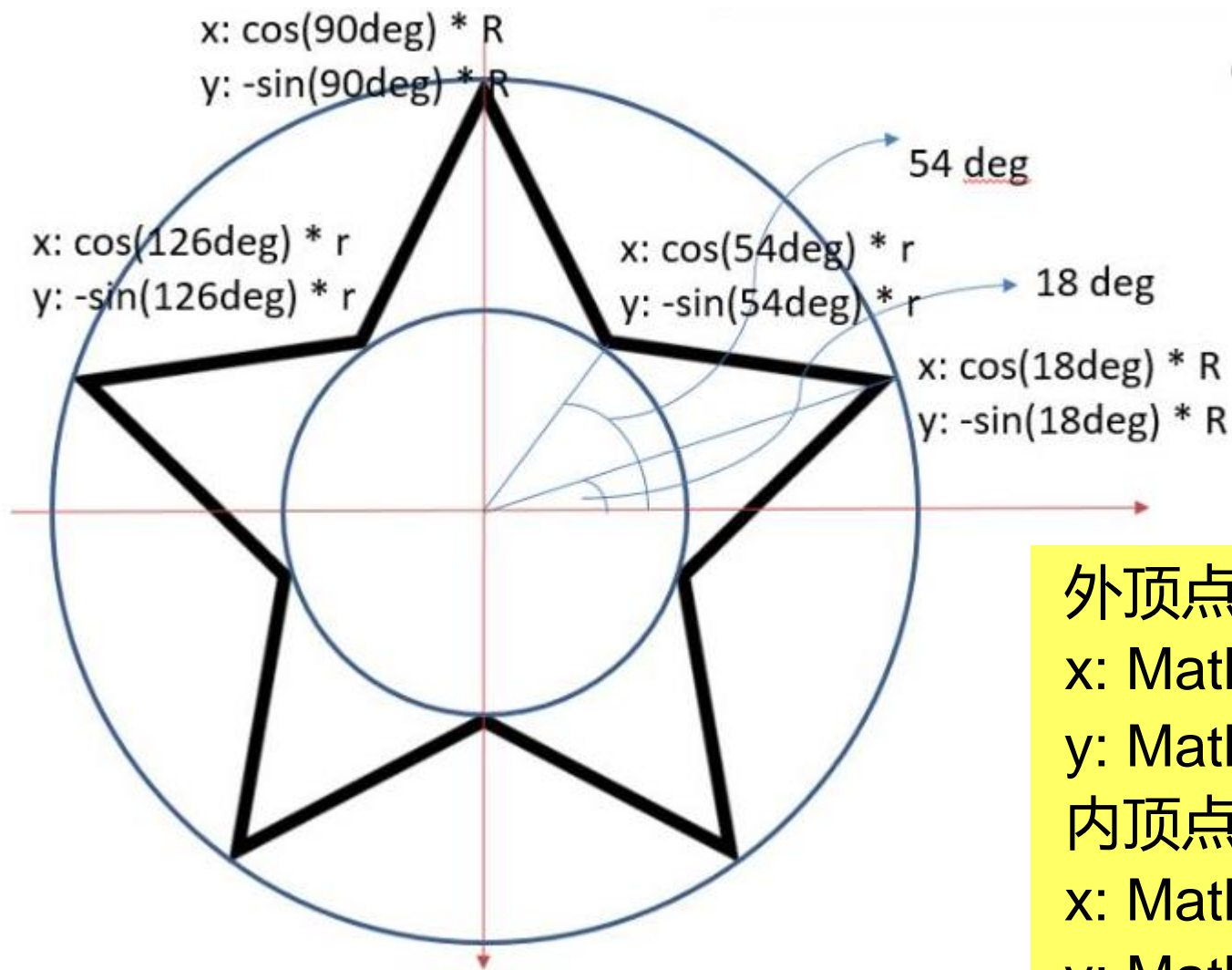
# 五角星

---





# 五角星



demo11-4.html

外顶点坐标

$x: \text{Math.cos}((18+72*i)/180*\text{Math.PI}) * R$

$y: \text{Math.sin}((18+72*i)/180*\text{Math.PI}) * R$

内顶点坐标

$x: \text{Math.cos}((54+72*i)/180*\text{Math.PI}) * r$

$y: \text{Math.sin}((54+72*i)/180*\text{Math.PI}) * r$

# 绘制一片星空

---

## ①设置五角星的填充色，绘制

```
cxt.fillStyle = "#fd3";  
cxt.strokeStyle = "#fd5";  
cxt.lineWidth = 3;  
cxt.lineJoin = "round";  
cxt.fill();  
cxt.stroke();
```

## ②绘制背景

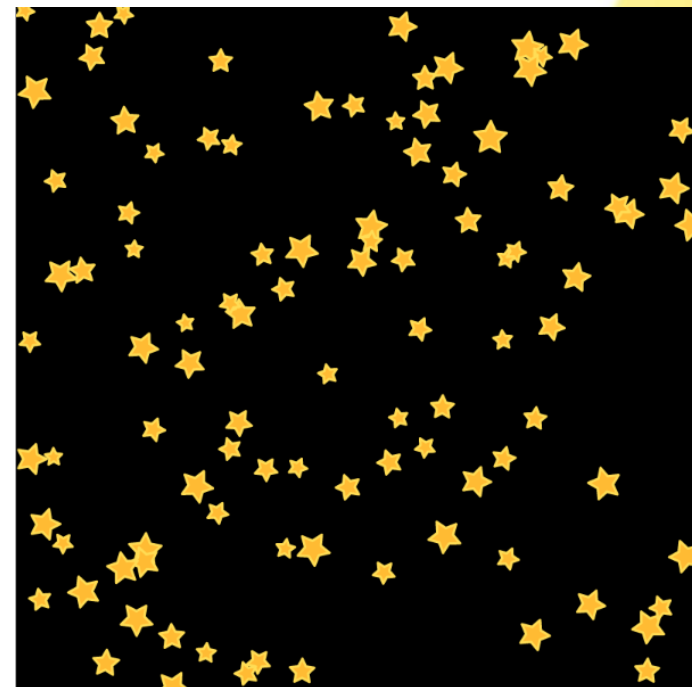
```
context1.fillStyle = "skyblue";  
context1.fillRect(0,0,canvas1.width,canvas1.height);
```

# 绘制一片星空

---

③循环设置多个五角星。使大小随机，位置随机，旋转角度随机

```
for(var i=0; i<100; i++){  
    var r = Math.random()*10+10;  
    var x = Math.random()*canvas1.width;  
    var y = Math.random()*canvas1.height;  
    var a = Math.random()*360;  
    drawStar(context1, r/2, r, x, y, a);  
}
```



02

## 画布转换和状态保存

# 画布转换

---

🎨 画布转换指的是转换画布的坐标系。

- 平移 —— `translate(x, y)`
- 旋转 —— `rotate(deg)`
- 缩放 —— `scale(sx, sy)`

# 平移

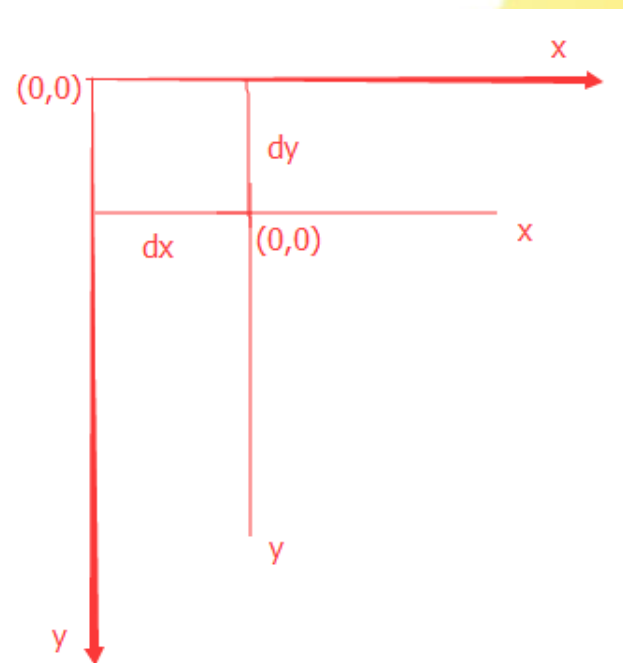
## 🎨 `translate(dx, dy)`

- 平移画布的用户坐标系统，即重新映射画布上的 (0,0) 位置。
- 参数 `dx` —— 坐标原点沿水平方向的偏移量
- 参数 `dy` —— 坐标原点沿垂直方向的偏移量

## 🎨 `translate()` 平移，坐标系会累加平移。

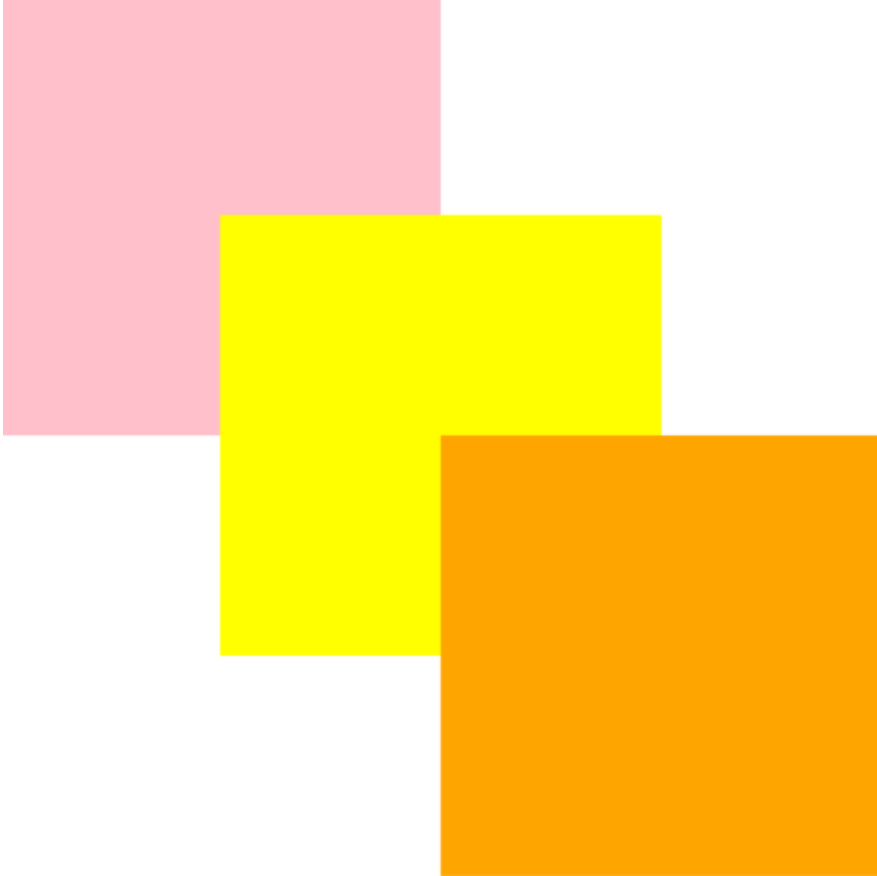
```
context.translate(100,100);
```

```
context.translate(100,100);
```



# 平移

---



demo11-6.html

```
context.fillStyle = "pink";  
context.fillRect(0,0,200,200);
```

```
context.fillStyle = "yellow";  
context.translate(100,100);  
context.fillRect(0,0,200,200);
```

```
context.fillStyle = "orange";  
context.translate(100,100);  
context.fillRect(0,0,200,200);
```

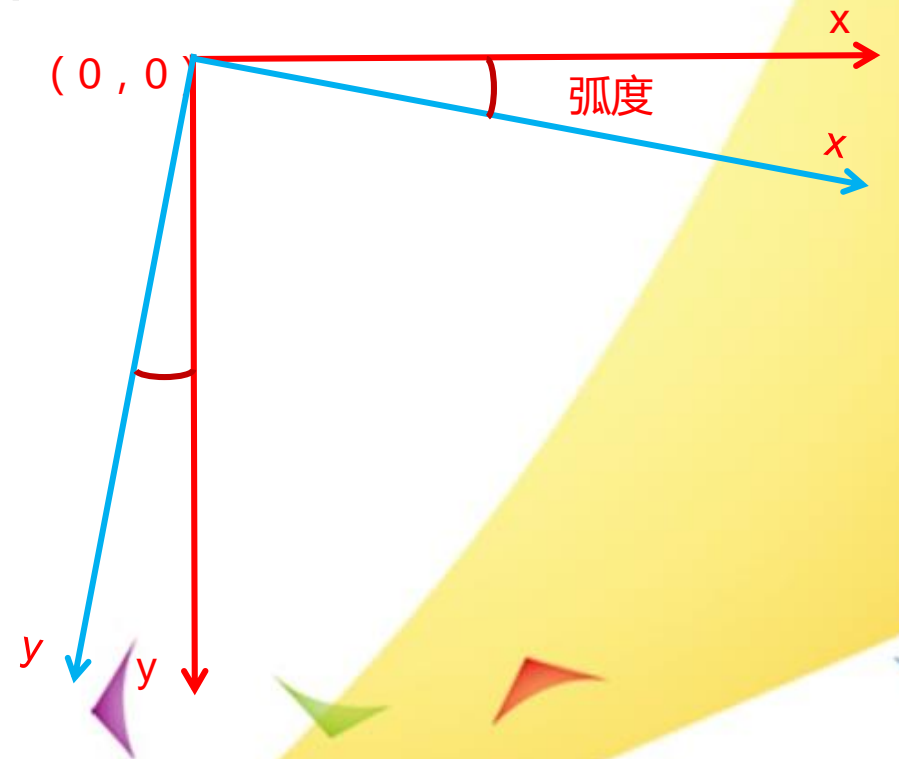
# 旋转

## 🎨 rotate(angle)

- 旋转画布的用户坐标系，即改变坐标系 x 与 y 轴的指向。
- 参数 angle —— 旋转角度，以弧度计。
  - 正值表示顺时针方向旋转
  - 负值表示逆时针方向旋转
  - 将角度转换为弧度公式

$$\text{angle} = \text{degrees}/180 * \text{Math.PI}$$

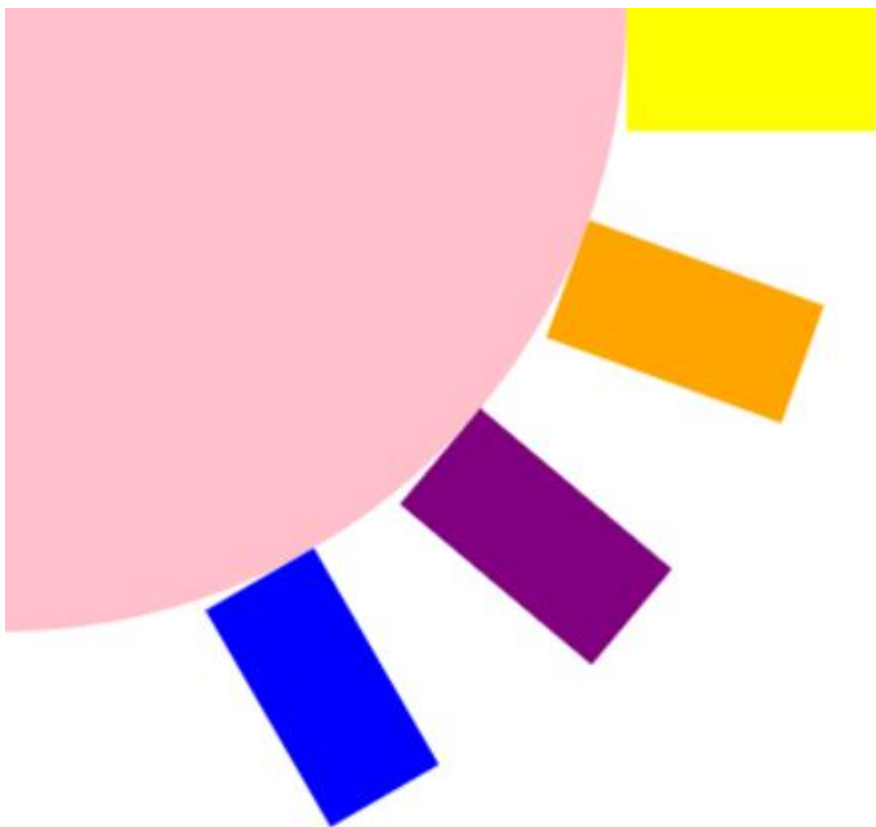
## 🎨 rotate() 旋转，坐标系会累加旋转。





# 旋转

---



demo11-7.html

```
context.beginPath();  
context.rotate(20/ 180 * Math.PI );  
context.fillStyle="orange";  
context.fillRect(250, 0, 100, 50);  
  
context.beginPath();  
context.rotate(20 / 180 * Math.PI);  
context.fillStyle="purple";  
context.fillRect(250, 0, 100, 50);  
...
```

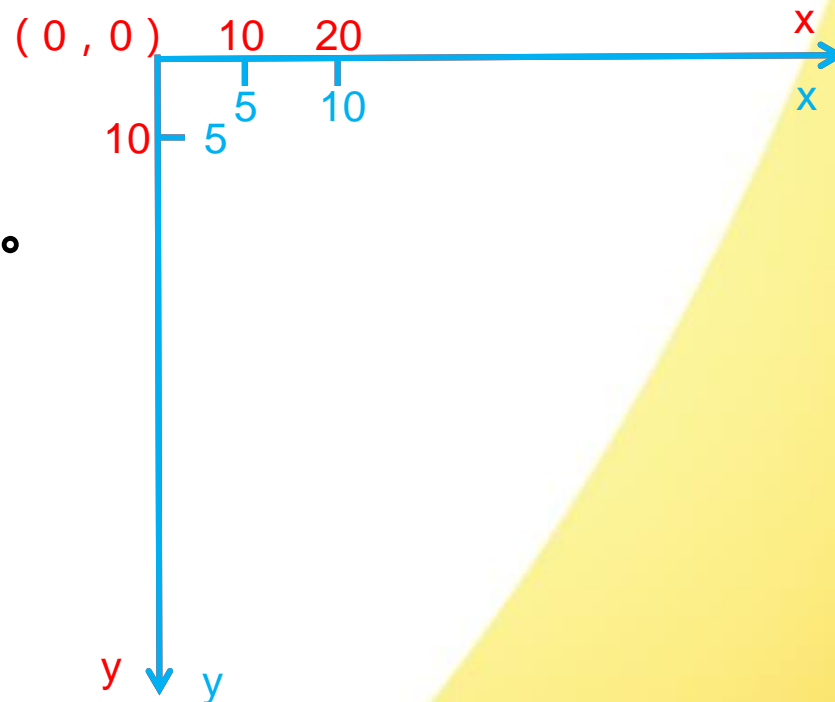
# 缩放

## 🎨 `scale(swidth, sheight)`

- 缩放画布的用户坐标系。
- 参数 `swidth` —— 坐标系 x 轴缩放倍数。
- 参数 `sheight` —— 坐标系 y 轴缩放倍数。

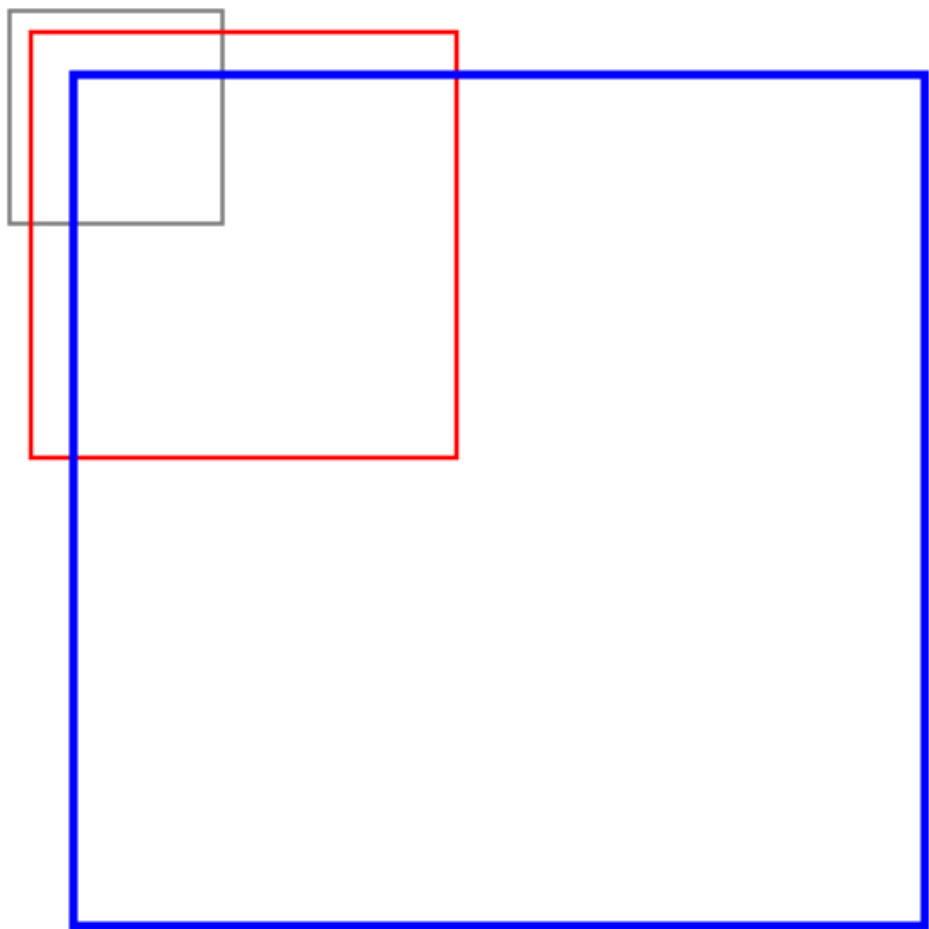
🎨 `scale()` 缩放，坐标系会累加缩放。

```
context.scale(2,2);
```



# 缩放

---



demo11-8.html

```
context.strokeRect(10, 10, 100, 100);
```

```
context.strokeStyle = 'red';
```

```
context.scale(2, 2);
```

```
context.strokeRect(10, 10, 100, 100);
```

```
// 定位也会缩放
```

```
context.strokeStyle = 'blue';
```

```
context.scale(2, 2);
```

```
context.strokeRect(10, 10, 100, 100);
```

# 画布坐标状态保存和恢复

---

## save()

- **保存**当前 canvas 绘图环境的所有属性、坐标变换信息等。

## restore()

- 将绘图环境状态**恢复**为保存值。

## 可以**嵌套式**的调用 save( )、restore( ) 方法。

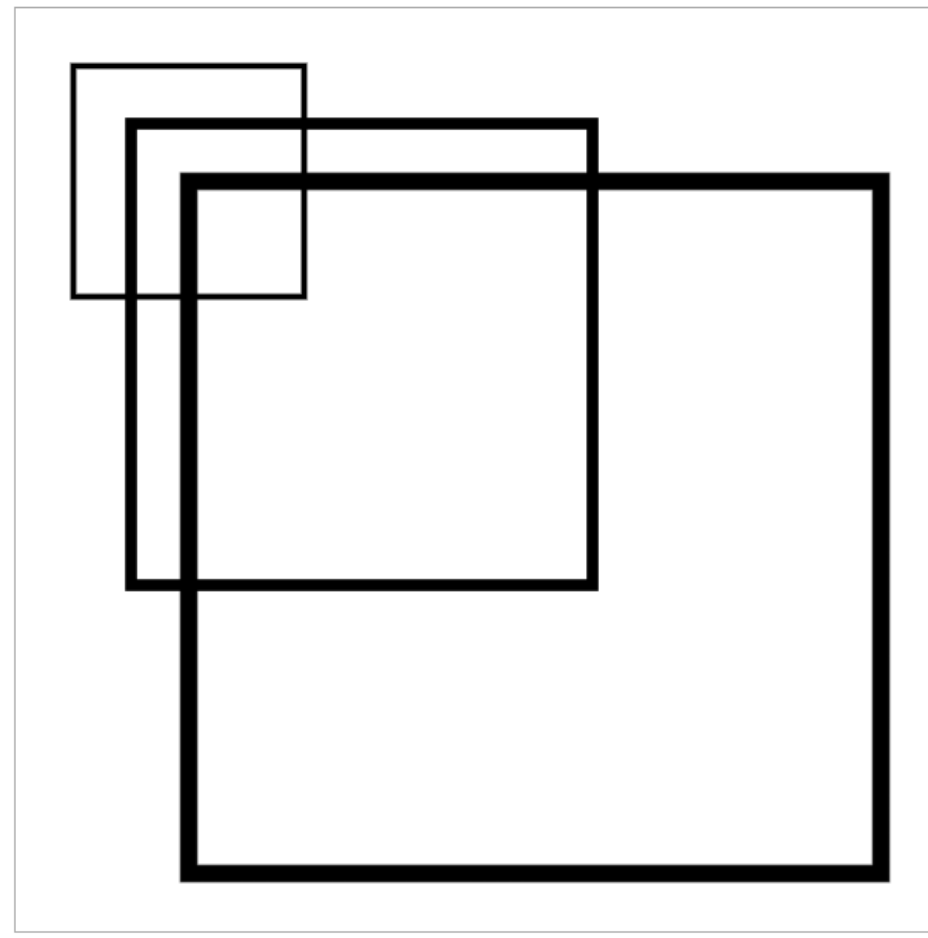
-  save() 把当前状态的一份拷贝压入到一个保存图像状态的**栈**中。  
restore() 是出栈。

# 坐标状态保存和恢复

```
context1.save();  
context1.scale(1,1);  
context1.strokeRect(50,50,200,200);  
context1.restore();
```

```
context1.save();  
context1.scale(2,2);  
context1.strokeRect(50,50,200,200);  
context1.restore();
```

```
context1.save();  
context1.scale(3,3);  
context1.strokeRect(50,50,200,200);  
context1.restore();
```

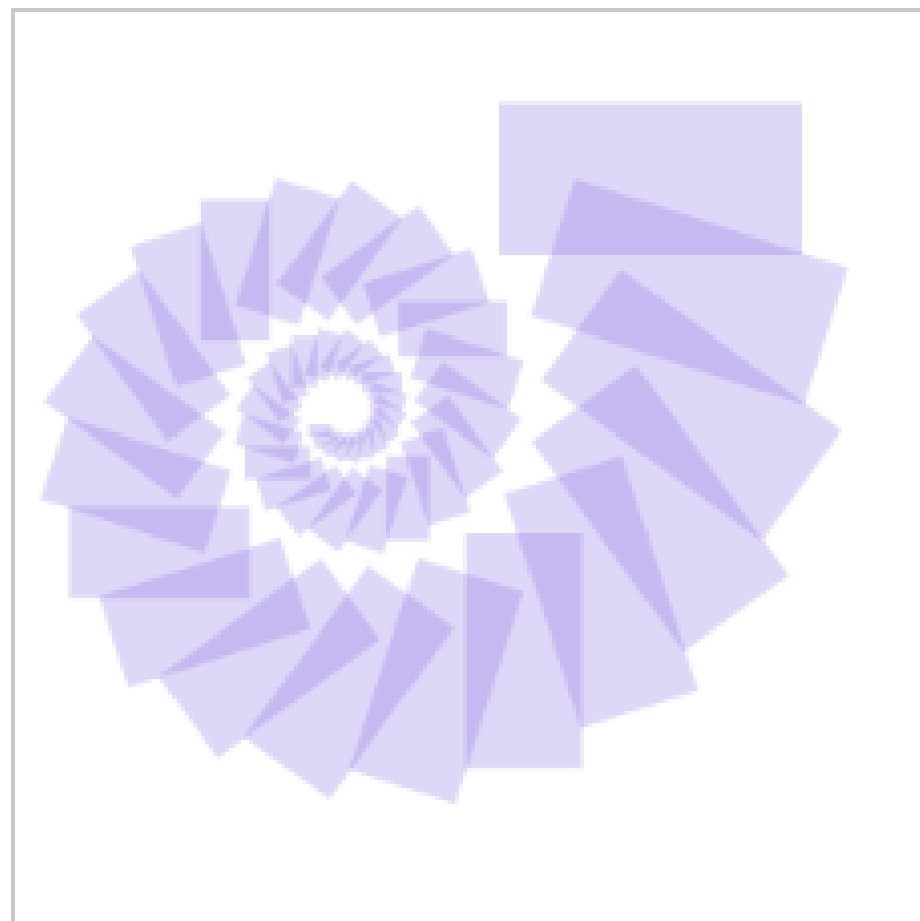


# 练习

---

```
context.translate(160, 30);
context.fillStyle = 'rgba(120,93,222,0.25)';
context.fillRect(0, 0, 100, 50);

for (var i = 0; i < 50; i++) {
    context.translate(25, 25);
    context.scale(0.95, 0.95);
    context.rotate(Math.PI / 10);
    context.fillRect(0, 0, 100, 50);
}
```



03

## 文字的渲染



# 绘制填充文字

---

## **fillText**( text, x, y, maxWidth ) 方法

- 在画布上绘制填色的文本。文本的默认颜色是黑色。


参数	描述
text	规定在画布上输出的文本。
x	开始绘制文本的 x 坐标位置（相对于画布）。
y	开始绘制文本的 y 坐标位置（相对于画布）。
maxWidth	可选。允许的最大文本宽度，以像素计。

## **font** 属性 — 设置画布上文本内容的当前字体属性



# 绘制描边文字

---

 **strokeText**( text, x, y, maxWidth ) 方法

- 在画布上绘制文本（没有填色）。文本的默认颜色是黑色。

参数	描述
text	规定在画布上输出的文本。
x	开始绘制文本的 x 坐标位置（相对于画布）。
y	开始绘制文本的 y 坐标位置（相对于画布）。
maxWidth	可选。允许的最大文本宽度，以像素计。

# 其他方法和属性

---

## `measureText(text)` 方法

- 返回一个对象，该对象包含以像素计的指定字体宽度。

## `textAlign` 属性 ( center | end | left | right )

- 根据锚点，设置文本内容的当前对齐方式。

## `textBaseline` 属性 ( top | middle | bottom )

- 根据锚点，设置在绘制文本时的当前文本基线。

```
var text = "河北师范大学";
```

```
var width = context.measureText(text).width;
```

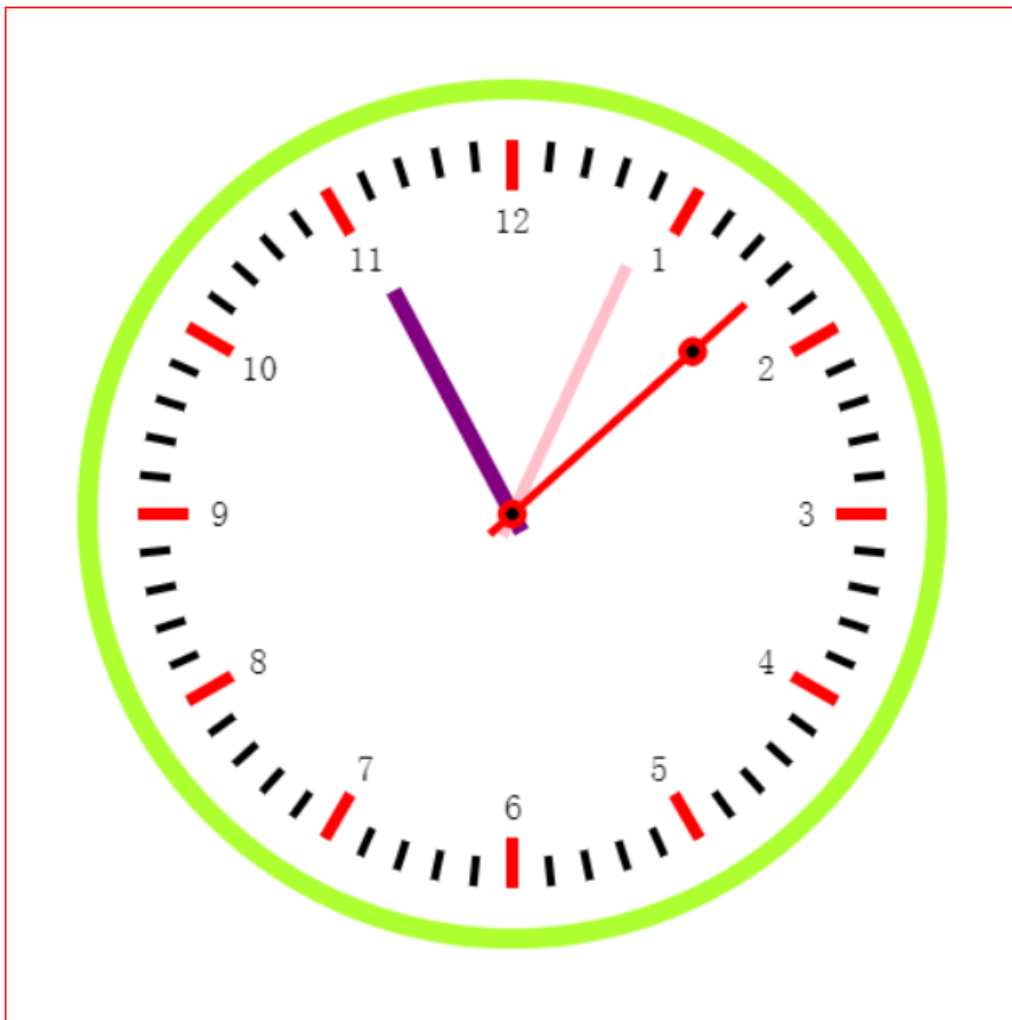
[demo11-13.html](#)

[demo11-14.html](#)

# 综合练习

---

## 钟表



04

阴影



# 阴影属性

---

- 🎨 shadowColor —— 设置用于阴影的**颜色**。
- 🎨 shadowBlur —— 设置用于阴影的**模糊**级别。
- 🎨 shadowOffsetX —— 设置形状与阴影的**水平距离**，默认值0。
- 🎨 shadowOffsetY —— 设置形状与阴影的**水平距离**，默认值0。
- 🎨 偏移量可正可负。

demo11-15.html

**THANKYOU**

