# canvas 绘图详解

HTML5 canvas

---

## 创建 canvas

```
<canvas id="canvas"></canvas>
<srcipt>
    var canvas = document.getElementById('canvas');

    canvas.width = 600;
    canvas.height = 600;

    var cxt = canvas.getContext('2d');
</script>
```

## 画一条直线

- canvas是基于状态绘图的，也就是说你要先有这个绘图的想法，想好绘制的路径，然后再去执行这个状态。

```
cxt.moveTo(50,50)  //相当于笔尖在50 50这个位置
cxt.lineTo(500,500) //画到了 500 500的位置
```

- 但是上面这两行代码都是绘制的状态，并没有真正的绘制，如果想画出来，单单去想怎么画是不行，一定要有实际的动作。体现在代码里就是：

```
cxt.stroke(); //这样就绘制了一条 从50 50 到 500 500 的直线
```

- 除了绘制路径状态，还可以指定绘制线条的宽度和颜色

```
cxt.lineWidth = 10;  //指定绘制线条的宽度
cxt.strokeStyle = '#f88';  //指定绘制线条的颜色
```

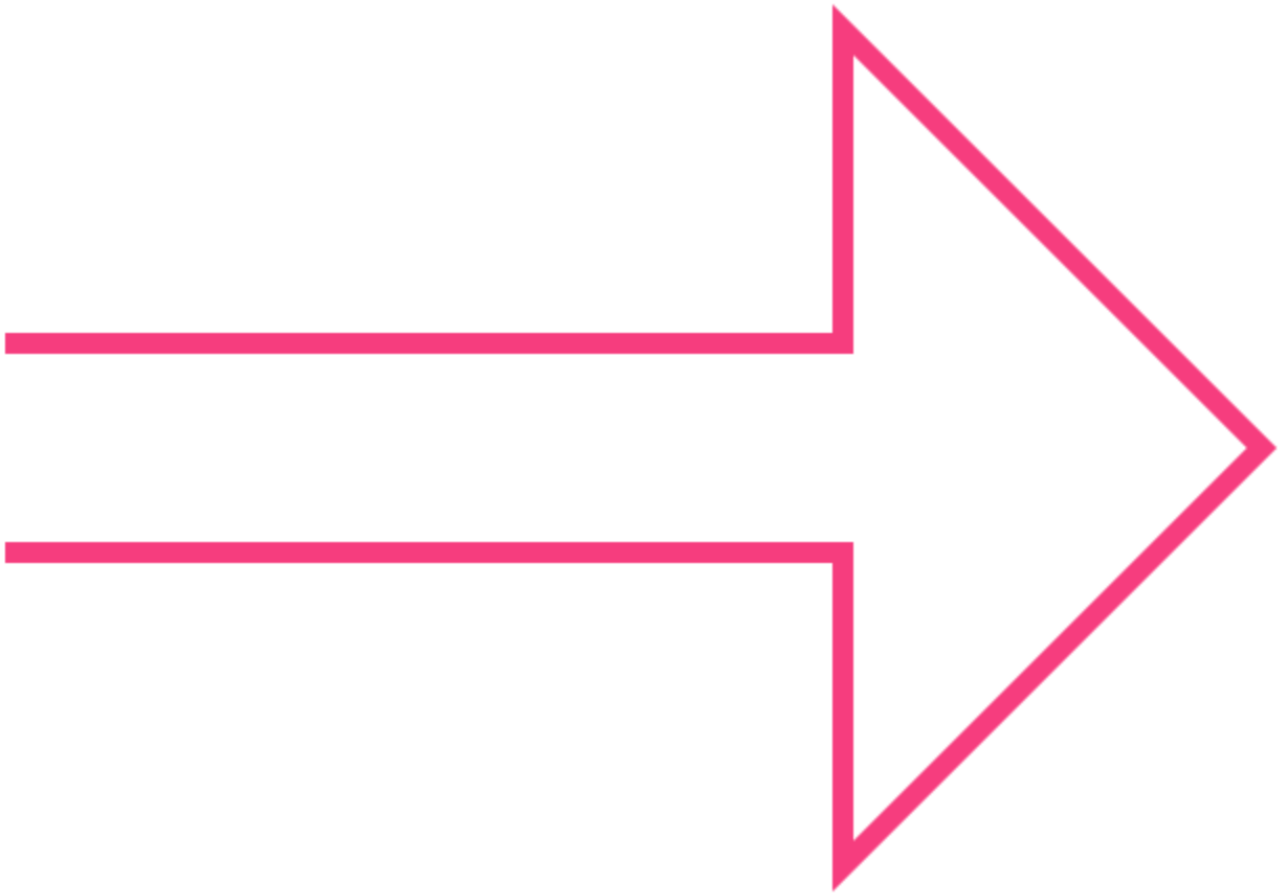
- 通过以上的代码可以看出，所有的状态都是基于cxt的，而不是对画的这个线条的。也就进

一步说明，canvas不是基于对象的。

## 画一条折线

```
<script>
    var cavnas = document.getElementById('canvas');
    canvas.width = canvas.height = 800;
    canvas.style.border = '1px solid #aaa';

    var cxt = canvas.getContext('2d');
```

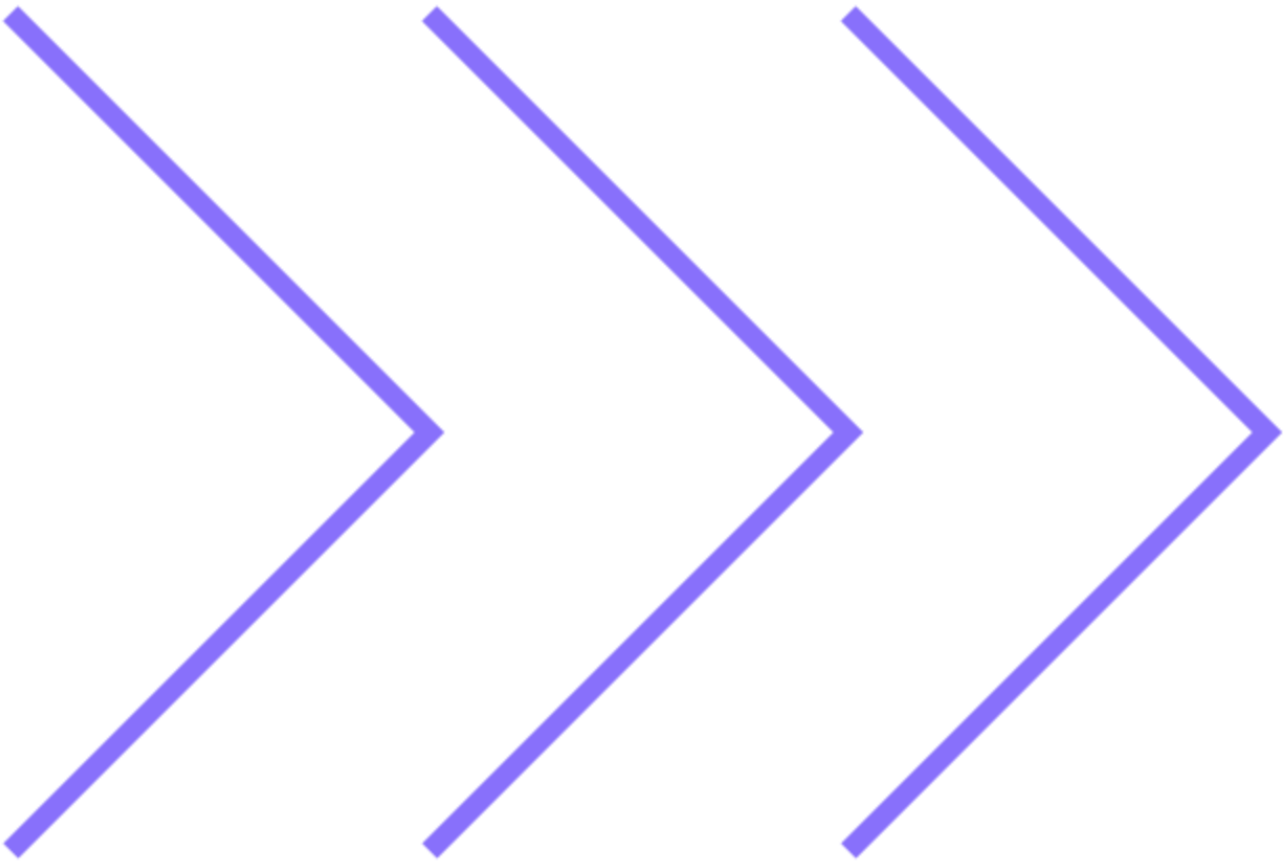
```
7.
8.      cxt.moveTo(100,350);
9.      cxt.lineTo(500,350);
10.     cxt.lineTo(500,200);
11.     cxt.lineTo(700,400);
12.     cxt.lineTo(500,600);
13.     cxt.lineTo(500,450);
14.     cxt.lineTo(100,450);
15.     cxt.lineWidth = 10;
16.     cxt.strokeStyle = "#F9387D";
17.     cxt.stroke();
18. </script>
```



## 画多条折线

```
1.  <canvas id="canvas"></canvas>
2.  <script>
3.      var cavnas = document.getElementById('canvas');
```

```
        canvas.width = canvas.height = 800;
        var cxt = canvas.getContext('2d');

        cxt.moveTo(100,200);
        cxt.lineTo(300,400);
        cxt.lineTo(100,600);

        cxt.moveTo(300,200);
        cxt.lineTo(500,400);
        cxt.lineTo(300,600);

        cxt.moveTo(500,200);
        cxt.lineTo(700,400);
        cxt.lineTo(500,600);

        cxt.lineWidth = 10;
        cxt.strokeStyle = "#886BFF";
        cxt.stroke();
    </script>
```



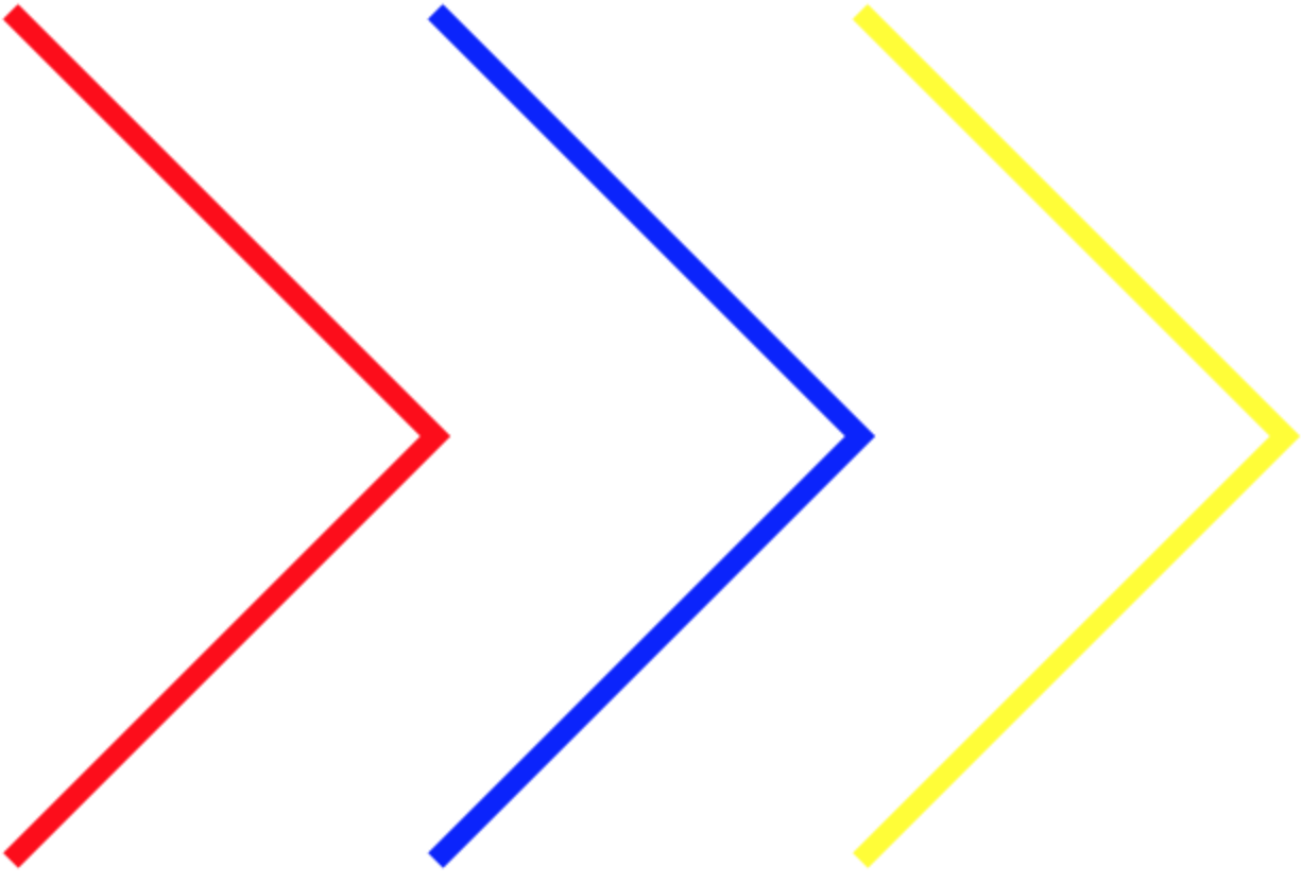
- 以上都是绘制一笔的线条，或者线条的宽度和颜色都是一样的。那么如果想绘制不同颜色和样式的线条，那么该怎么做呢？
- 可能大家会这么想：

```
cxt.moveTo(100,200);
cxt.lineTo(300,400);
cxt.lineTo(100,600);
cxt.lineWidth = 10;
cxt.strokeStyle = "red";  //红色状态
cxt.stroke();  //绘制 为红色

cxt.moveTo(300,200);
cxt.lineTo(500,400);
cxt.lineTo(300,600);
cxt.lineWidth = 10;
cxt.strokeStyle = "blue"; //蓝色状态 会覆盖 红色状态
cxt.stroke();  //绘制 所有线条都是蓝色

cxt.moveTo(500,200);
cxt.lineTo(700,400);
cxt.lineTo(500,600);

cxt.lineWidth = 10;
cxt.strokeStyle = "yellow"; //黄色状态 会覆盖 蓝色状态
cxt.stroke();  //绘制 所有线条都是黄色
```

- 但是这样的效果是不对的，这就是为什么说canvas是基于状态绘制的。在绘制过程中，canvas不会单单把每一段代码进行绘制，它会检查整个代码的绘制状态，然后基于这些状态进行绘制。
- 那么想基于3种不同的状态进行绘制，该如何完成呢？
- canvas 提供了 beginPath() 接口

```
cxt.lineWidth = 10;

cxt.beginPath();  //可以省略，但是为了维持代码的一致性，建议写上。
cxt.moveTo(100,200); //可以换成 lineTo()
cxt.lineTo(300,400);
cxt.lineTo(100,600);
cxt.strokeStyle = "red";
cxt.stroke();

```

```
cxt.beginPath();
cxt.moveTo(300,200);
cxt.lineTo(500,400);
cxt.lineTo(300,600);
cxt.strokeStyle = "blue";
cxt.stroke();

cxt.beginPath();
cxt.moveTo(500,200);
cxt.lineTo(700,400);
cxt.lineTo(500,600);
cxt.strokeStyle = "yellow";
cxt.stroke();
```



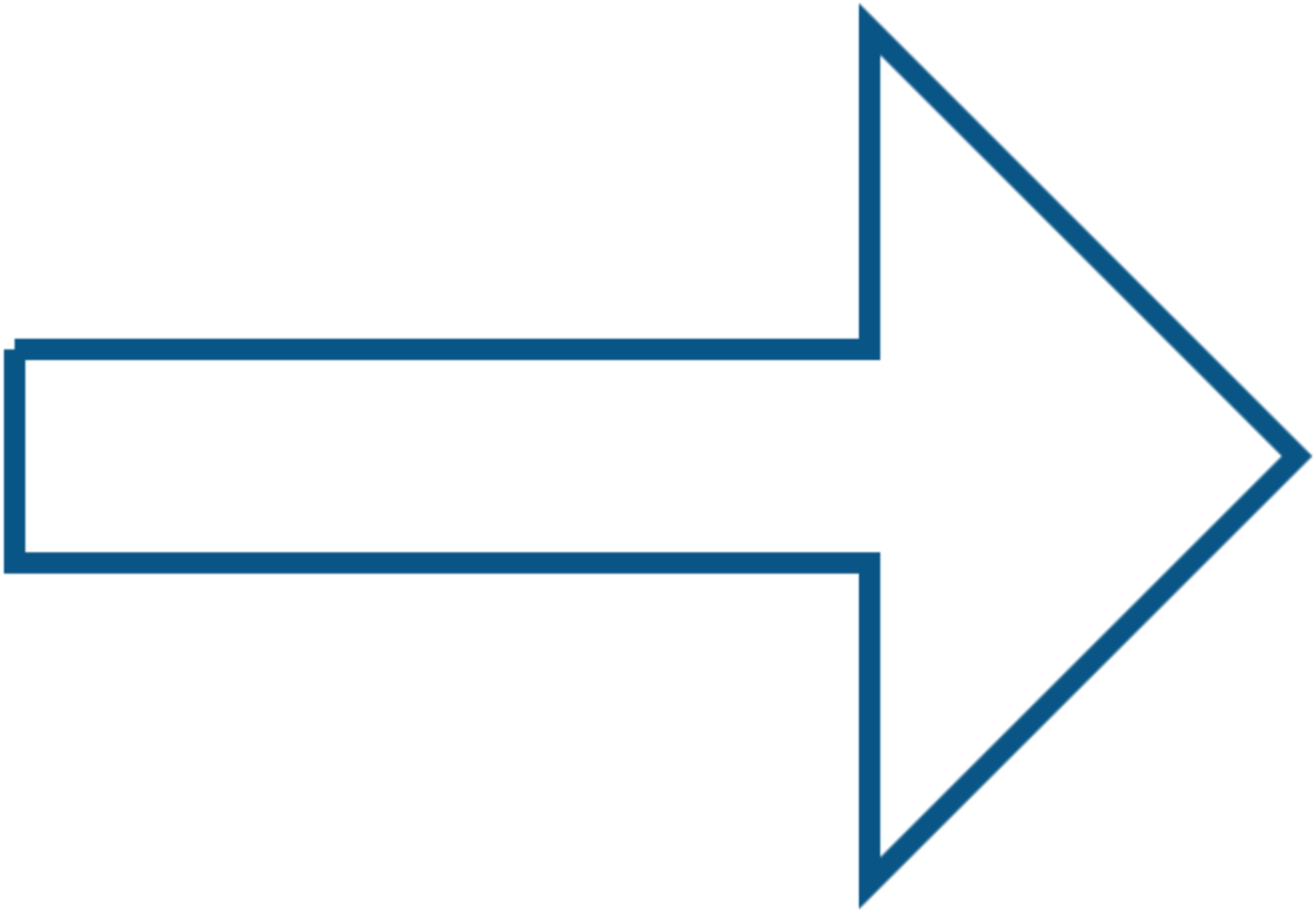
- beginPath 用于指定全新的路径，值得注意的是 cxt.lineWidth = 10; 对于没有改变的状态，beginPath 会一直使用这个状态。
- beginPath + lineTo 相当于一次新的 moveTo 所以moveTo可以省略。

所以对于什么状态一直保持着，什么状态改变了，在使用canvas绘图的时候，一定要搞

> 清楚。

## 绘制封闭图形

```
<script>
    var cavnas = document.getElementById('canvas');
    canvas.width = canvas.height = 800;
    canvas.style.border = '1px solid #aaa';

    var cxt = canvas.getContext('2d');

    cxt.moveTo(100,350);
    cxt.lineTo(500,350);
    cxt.lineTo(500,200);
    cxt.lineTo(700,400);
    cxt.lineTo(500,600);
    cxt.lineTo(500,450);
    cxt.lineTo(100,450);

    //cxt.lineTo(100,350);  //闭合后有缺口

    cxt.closePath(); //这样可以完美闭合线段

    cxt.lineWidth = 10;
    cxt.strokeStyle = "#058";
    cxt.stroke();
</script>
```



- 上面代码存在的问题，在闭合出有明显的一个缺口。

> 解决问题：使用closePath方法，closePath还有结束当前绘制状态的作用。

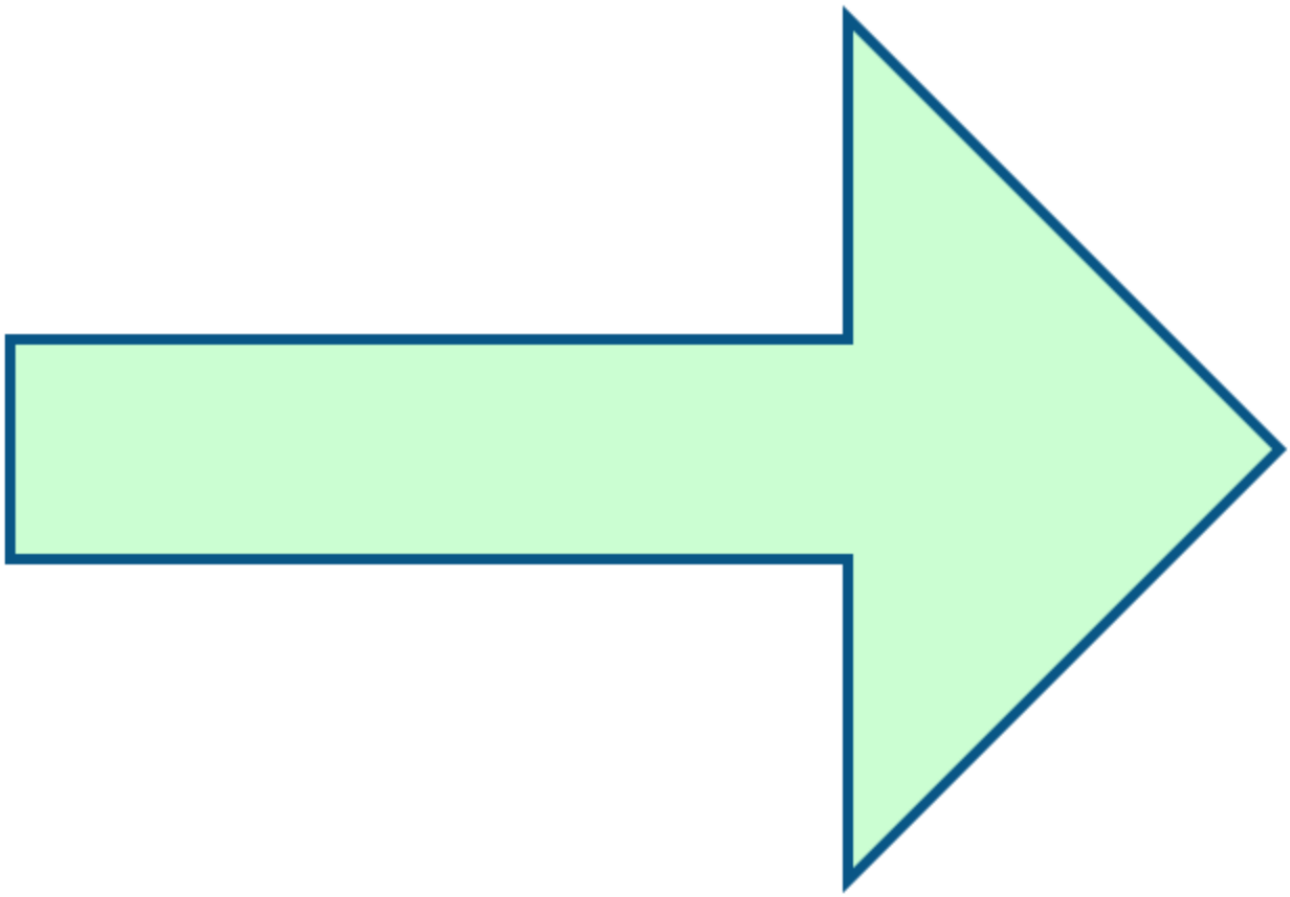
## 对多边形进行填充

> 使用：cxt.fillStyle = '颜色' 指定要填充的颜色值
> 可以为 单词、16进制、rgb、rgba、hsl、hsla
>
> 使用：cxt.fill(); 为指定绘制的状态区域进行填充。

下面来比较 下面两段代码位置不同所带来的不同的效果：

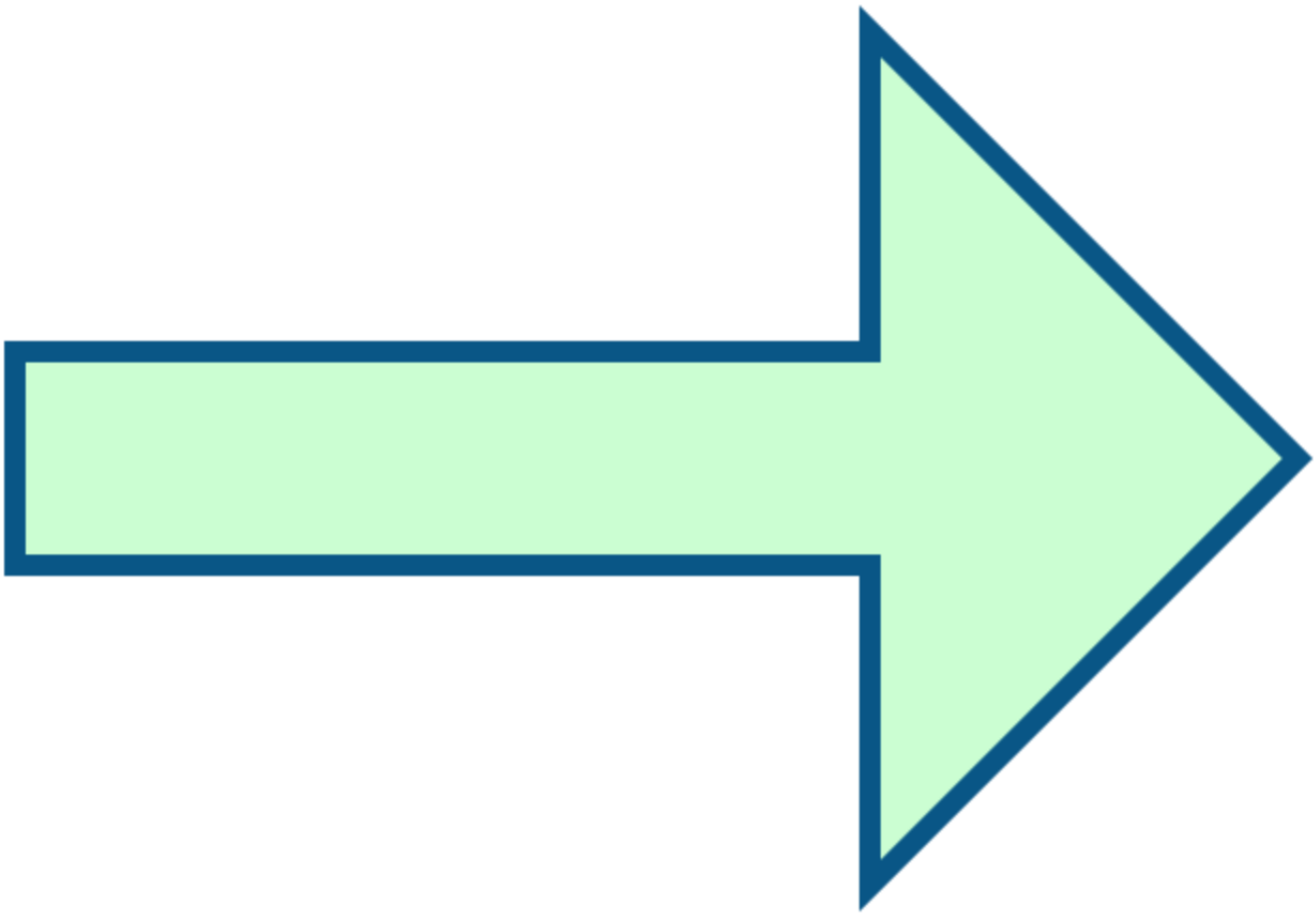
```
<script>
    var cavnas = document.getElementById('canvas');
```

```
        canvas.width = canvas.height = 800;
        var cxt = canvas.getContext('2d');

        cxt.moveTo(100,350);
        cxt.lineTo(500,350);
        cxt.lineTo(500,200);
        cxt.lineTo(700,400);
        cxt.lineTo(500,600);
        cxt.lineTo(500,450);
        cxt.lineTo(100,450);
        cxt.closePath();
        cxt.lineWidth = 10;

        cxt.strokeStyle = "#058";
        cxt.stroke();

        cxt.fillStyle = '#CAFFD1';
        cxt.fill();

    </script>
```



```
<script>
    var cavnas = document.getElementById('canvas');
    canvas.width = canvas.height = 800;
    var cxt = canvas.getContext('2d');

    cxt.moveTo(100,350);
    cxt.lineTo(500,350);
    cxt.lineTo(500,200);
    cxt.lineTo(700,400);
    cxt.lineTo(500,600);
    cxt.lineTo(500,450);
    cxt.lineTo(100,450);
    cxt.closePath();
    cxt.lineWidth = 10;

    cxt.fillStyle = '#CAFFD1';
    cxt.fill();

    cxt.strokeStyle = "#058";
    cxt.stroke();
```

```
</script>
```



> 先填充后描边（内侧一半的边会被覆盖），和先描边后填充的效果是不一样的，这一点要掌握。

## 封装绘制矩形函数

```
//绘制起点 宽 高 边框宽 边框颜色 填充颜色
function drawRect(x,y,w,h,bw,bc,fc){
    cxt.beginPath();
    cxt.moveTo(x, y);
    cxt.lineTo(x + w, y);
    cxt.lineTo(x + w, y + h );
    cxt.lineTo(x, y + h);
    cxt.closePath();

    cxt.strokeStyle = bc;
```

```
        cxt.lineWidth = bw;
        cxt.fillStyle = fc;

        cxt.fill();
        cxt.stroke();
    }
```

- canvas 绘制矩形的原生接口

```
cxt.rect(x,y,w,h)  // 绘制矩形状态
cxt.fillRect(x,y,w,h) //直接使用当前的fillStyle绘制一个填充的矩形
cxt.strokeRect(x,y,w,h) //直接使用当前的strokeStyle绘制一个边框矩形
```

- 请自行测试

```
var cavnas = document.getElementById('canvas');
    canvas.width = canvas.height = 800;
    var cxt = canvas.getContext('2d');

    cxt.fillStyle = '#C5FFDA';
    cxt.lineWidth = 10;
    cxt.fillRect(10,10,400,400);
    cxt.strokeRect(10,10,400,400);
```

## canvas 的遮挡关系

```
<canvas id="canvas"></canvas>
<script>
    var cavnas = document.getElementById('canvas');
    canvas.width = canvas.height = 800;
    var cxt = canvas.getContext('2d');

    cxt.beginPath();
    cxt.fillStyle = '#C5FFDA';
    cxt.fillRect(10,10,400,400);

    cxt.beginPath();
    cxt.fillStyle = 'rgba(222,111,234,.5)';
    cxt.fillRect(200,200,400,400);
</script>
```