

数据可视化 Canvas

刘军 liujun

目录

content



1 邂逅 Canvas

2 Canvas绘制图形

3 Canvas样式和颜色

4 Canvas状态和形变

5 Canvas动画和案例






■ 什么是Canvas

- Canvas 最初由Apple于2004 年引入，用于Mac OS X WebKit组件，同时给Safari浏览器等应用程序提供支持。后来，它被Gecko内核的浏览器（尤其是Mozilla Firefox），Opera和Chrome实现，并被W3C提议为下一代的**标准元素（HTML5新增元素）**。
- Canvas提供了非常多的JavaScript绘图API（比如：绘制路径、矩形、圆、文本和图像等方法），集合<canvas>元素可以绘制各种2D图形。
- Canvas API 主要用于绘制 2D 图形。当然也可以使用 Canvas 提供的 WebGL API 来绘制 3D 图形。

■ Canvas的应用场景

- 可以用于**动画、游戏画面、数据可视化、图片编辑以及实时视频处理**等方面。

■ Canvas 浏览器兼容性

Element					
<canvas>	4.0	9.0	2.0	3.1	9.0

■ Canvas 优点:

- Canvas提供的功能更加原始，适合像素级处理，动态渲染和量大数据的绘制，如：图片编辑、热力图、炫光尾迹特效等。
- Canvas非常适合图像密集型的游戏开发，适合频繁重绘大量的图形对象。
- Canvas能够以 .png 或 .jpg 格式保存图像，适合对图片进行像素级的处理。

■ Canvas 缺点:

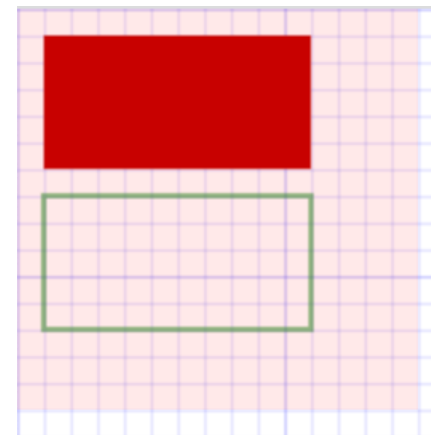
- 在移动端如果Canvas使用数量过多，会使内存占用超出了手机承受范围，可能会导致浏览器崩溃。
- Canvas 绘图只能通过JavaScript脚本操作。
- Canvas 是由一个个像素点构成的图形，放大会使图形变得颗粒状和像素化（导致图形模糊）。

■ 使用Canvas的注意事项:

- `<canvas>` 和 `` 元素很相像, 不同就是它没有 `src` 和 `alt` 属性。
- `<canvas>` 标签只有两个属性——`width`和`height`(单位默认为`px`)。没宽高时, `canvas` 会初始化宽为 `300px` 和高为 `150px`。
- `<canvas>` 元素**必须需要结束标签** (`</canvas>`)。如结束标签不存在, 则文档其余部分会被认为是**替代内容**, 将不显示出来。
- 可以通过判断 `canvas.getContext()` 方法是否存在来检查浏览器是否支持Canvas (现代浏览器基本都支持) 。

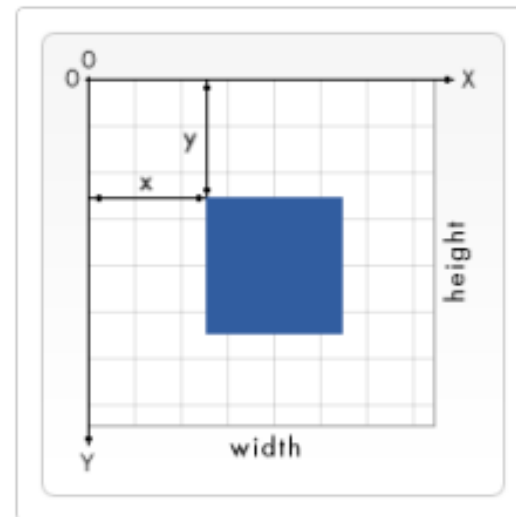
■ 初体验Canvas

- 1.Canvas通用模板
- 2.Canvas绘制正方形



Canvas Grid 和 坐标空间

- 在开始画图之前，我们需要了解一下Canvas 网格（Canvas Grid）和 坐标空间。
- Canvas Grid（或坐标空间）
 - 假如，HTML 模板中有个宽 150px, 高 150px 的 `<canvas>` 元素。<canvas>元素默认被网格所覆盖。
 - 通常来说网格中的一个单元相当于 canvas 元素中的一像素。
 - 网格的原点位于坐标 (0,0) 的左上角。所有图形都相对于该原点绘制。
 - 网格也可理解为是坐标空间（坐标系），坐标原点位于canvas元素左上角（被称为初始坐标系）
 - ✓ 如右图中蓝色正方形，左上角的坐标为 (x, y)
 - 网格或坐标空间是可以变换的，后面会讲如何将原点转换到不同的位置，旋转网格甚至缩放它。
 - 注意：移动、旋转、缩放坐标系后，默认所有后续变换都将基于新坐标系的变换。



绘制矩形(Rectangle)

■ Canvas支持两种方式来绘制矩形：矩形方法和 路径方法。

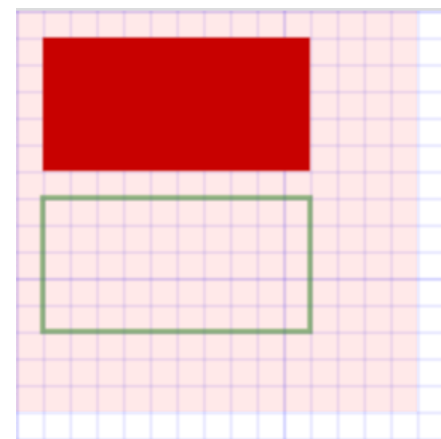
- 路径是点列表，由线段连接。这些线段可以具有不同的形状、弯曲或不弯曲的、连续或不连续的、不同的宽度和不同的颜色。
- 除了矩形，其他的图形都是通过一条或者多条路径组合而成的。
- 通常我们会通过众多的路径来绘制复杂的图形。

■ Canvas 绘图的矩形方法：

- `fillRect(x, y, width, height)`： 绘制一个填充的矩形
- `strokeRect(x, y, width, height)`： 绘制一个矩形的边框
- `clearRect(x, y, width, height)`： 清除指定矩形区域，让清除部分完全透明。

■ 方法参数：

- 上面的方法都包含了相同的参数。
- `x` 与 `y` 指定了在canvas画布上所绘制矩形的左上角（相对于原点）的坐标（不支持 `undefined`）。
- `width` 和 `height` 设置矩形的尺寸。



■ 什么是路径？

- 图形的基本元素是路径。路径是点列表，由线段连接。这些线段可以具有不同形状、弯曲或不弯曲的、连续或不连续的、不同的宽度和颜色。
- 路径是可由很多子路径构成，这些子路径都是在一个列表中，列表中所有子路径（线、弧形等）将构成图形。
- 绘制一个路径，甚至一个子路径，通常都是闭合的（会调用closePath来闭合）。

■ 使用路径绘制图形的步骤：

- 1.首先需要创建路径起始点 (beginPath) 。
- 2.然后使用绘图命令去画出路径(arc 、 lineTo)。
- 3.之后把路径闭合(closePath , 不是必须)。
- 4.一旦路径生成，就能通过描边(stroke)或填充路径区域(fill)来渲染图形。

■ 以下是绘制路径时，所要用到的函数

- beginPath(): 新建一条路径，生成之后，图形绘制命令被指向到新的路径上绘图，不会关联到旧的路径。
- closePath(): 闭合路径之后图形绘制命令又重新指向到 beginPath 之前的上下文中。
- stroke(): 通过线条来绘制图形描边（针对当前路径图形）。
- fill(): 通过填充路径的内容区域生成实心的图形（针对当前路径图形）。

```
ctx.beginPath();
ctx.arc(75, 75, 50, 0, Math.PI * 2, true); // 绘制
ctx.moveTo(110, 75);
ctx.arc(75, 75, 35, 0, Math.PI, false); // 口 (顺时针)
ctx.moveTo(65, 65);
ctx.arc(60, 65, 5, 0, Math.PI * 2, true); // 左眼
ctx.moveTo(95, 65);
ctx.arc(90, 65, 5, 0, Math.PI * 2, true); // 右眼
ctx.stroke();
```



路径-绘制直线

■ 移动画笔 moveTo(x, y) 方法

- moveTo 方法是不能画出任何东西，但是它也是**路径列表**的一部分
- moveTo 可以想象为在纸上绘画，如：一支钢笔的笔尖从一个点到另一个点的移动过程。可将笔移动到指定的坐标 x、y 上。
- 当 canvas 初始化或者beginPath()调用后，通常会使用moveTo(x, y)函数设置绘画的起点。
- 使用moveTo函数能够**绘制一些不连续的路径**。

■ 绘制直线 lineTo(x, y) 方法, 绘制一条从当前位置到指定 (x, y)位置的直线

- 有两个参数, (x, y) 代表坐标系中**直线结束的点**。
- **开始点和之前绘制的路径有关, 之前路径结束点就是接下来的开始点。**
- 当然开始点也可以通过moveTo(x, y)函数改变。

■ 绘制一条直线

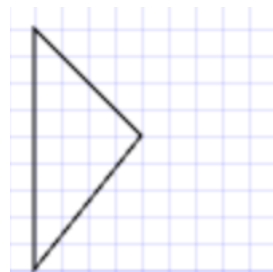
- 第一步：调用 beginPath() 来生成路径。本质上，路径是由很多子路径（线、弧形、等）构成。
- 第二步：调用moveTo、lineTo函数来绘制路径（路径可以是连续也可以不连续）。
- 第三步：闭合路径 closePath(), 虽然不是必需的，但是通常都是要闭合路径。
- 第四步：调用stroke()函数来给直线描边。



路径-绘制三角形(Triangle)

■ 绘制一个三角形步骤

- 第一步：调用 `beginPath()` 来生成路径。
- 第二步：调用 `moveTo()`、`lineTo()` 函数来绘制路径。
- 第三步：闭合路径 `closePath()`，**不是必需的**。
 - ✓ `closePath()` 方法会通过绘制一条**从当前点到开始点的直线来闭合图形**。
 - ✓ 如果图形是已经闭合了的，即当前点为开始点，该函数什么也不做。
- 第四步：调用 `stroke()` 函数来给线描边，或者调用 `fill()` 函数来填充（**使用填充 `fill` 时，路径会自动闭合，而 `stroke` 不会**）。

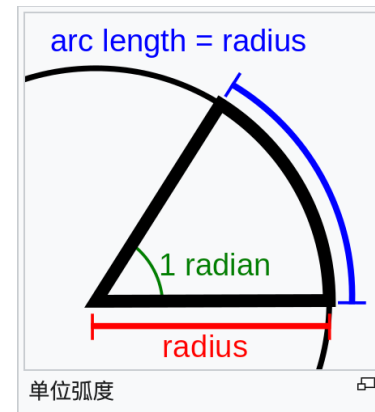


路径-绘制圆弧 (Arc)、圆 (Circle)

■ 绘制圆弧或者圆，使用arc()方法。

□ arc(x, y, radius, startAngle, endAngle, anticlockwise), 该方法有六个参数：

- ✓ x、y: 圆心坐标。
- ✓ radius: 圆弧半径。
- ✓ startAngle、endAngle: 指定开始和结束的弧度。以x轴为基准（注意：单位是弧度，不是角度）。
- ✓ anticlockwise: 为一个布尔值。为 true，是逆时针方向，为false，是顺时针方向，默认为false。



■ 认识弧度

□ 弧度 (英语: radian), 是平面角的单位。1单位弧度为: 圆弧长度等于半径时的圆心角, 而一个完整的圆的弧度是 $\text{Math.PI} * 2$, 半圆弧度是 Math.PI

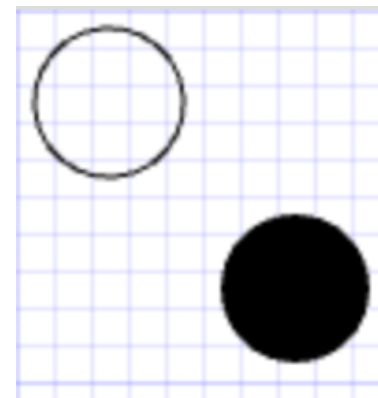
■ 角度和弧度关系

□ 角度与弧度的 JS 表达式: 弧度 = $(\text{Math.PI} / 180) * \text{角度}$, 即 1角度 = $\text{Math.PI} / 180$ 个弧度

- ✓ 比如: 旋转90°: $\text{Math.PI} / 2$; 旋转180°: Math.PI ; 旋转360°: $\text{Math.PI} * 2$; 旋转-90°: $-\text{Math.PI} / 2$;

■ 绘制一个圆弧的步骤

- 第一步: 调用 beginPath() 来生成路径。
- 第二步: 调用arc()函数来绘制圆弧。
- 第三步: 闭合路径 closePath(), 不是必需的。
- 第四步: 调用stroke()函数来描边, 或者调用fill()函数来填充 (使用填充 fill 时, 路径会自动闭合)。



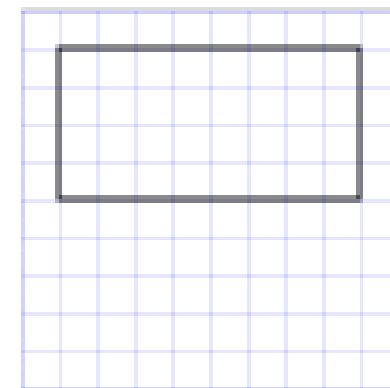
路径-矩形 (Rectangle)

■ 绘制矩形的另一个方法:

- 用 `rect()` 函数绘制, 即将一个矩形路径增加到当前路径上
- `rect(x, y, width, height)`
 - ✓ 绘制一个左上角坐标为 (x,y) , 宽高为 `width`、`height` 的矩形。

■ 注意:

- 当该方法执行的时候, `moveTo(x, y)` 方法自动设置坐标参数 $(0,0)$ 。也就是说, 当前笔触自动重置回默认坐标。



■ 前面已经学过了很多绘制图形的方法。如果我们**想要给图形上色**，有两个重要的属性可以做到：

□ `fillStyle = color`：设置图形的填充颜色，需在 `fill()` 函数前调用。

□ `strokeStyle = color`：设置图形轮廓的颜色，需在 `stroke()` 函数前调用。

■ **color颜色**

□ `color` 可以是表示 CSS 颜色值的字符串，支持：关键字、十六进制、`rgb`、`rgba`格式。

□ 默认情况下，线条和填充颜色都是黑色（CSS 颜色值 `#000000`）。

■ **注意**

□ 一旦设置了 `strokeStyle` 或者 `fillStyle` 的值，那么这个新值就会成为新绘制的图形的默认值。

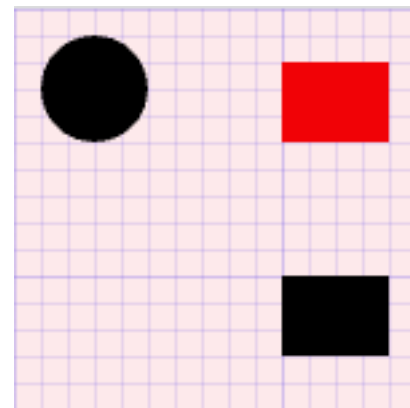
□ 如果要给图形上不同的颜色，你需要重新设置 `fillStyle` 或 `strokeStyle` 的值。

■ **额外补充**

□ `fill()` 函数是图形填充，`fillStyle`属性是设置填充色

□ `stroke()` 函数是图形描边，`strokeStyle`属性是设置描边色

```
// 这些 fillStyle 的值均为 '橙色'  
ctx.fillStyle = "orange";  
ctx.fillStyle = "#FFA500";  
ctx.fillStyle = "rgb(255,165,0)";  
ctx.fillStyle = "rgba(255,165,0,1)";
```



透明度 Transparent

- 除了可以绘制实色图形，我们还可以用 canvas 来绘制半透明的图形。
- 方式一：strokeStyle 和 fillStyle 属性结合 RGBA：
- 方式二：globalAlpha 属性
 - globalAlpha = 0 ~ 1
 - ✓ 这个属性影响到 canvas 里所有图形的透明度
 - ✓ 有效的值范围是 0.0（完全透明）到 1.0（完全不透明），默认是 1.0。

```
// 设置透明度值
ctx.globalAlpha = 0.2;

// 画半透明圆
for (var i=0;i<7;i++){
  ctx.beginPath();
  ctx.arc(75,75,10+10*i,0,Math.PI*2,true);
  ctx.fill();
}
```

```
// 指定透明颜色，用于描边和填充样式
ctx.strokeStyle = "rgba(255,0,0,0.5)";
ctx.fillStyle = "rgba(255,0,0,0.5)";
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/248004115114007002>