

03 | 声明式图形系统：如何用SVG图形元素绘制可视化图表？

2020-06-26 月影

跟月影学可视化

[进入课程 >](#)



讲述：月影

时长 16:14 大小 14.88M



你好，我是月影。今天，我们来讲 SVG。

SVG 的全称是 Scalable Vector Graphics，可缩放矢量图，它是浏览器支持的一种基于 XML 语法的图像格式。

对于前端工程师来说，使用 SVG 的门槛很低。因为描述 SVG 的 XML 语言本身和 HTML 非常接近，都是由标签 + 属性构成的，而且浏览器的 CSS、JavaScript 都能够正常作用于 SVG 元素。这让我们在操作 SVG 时，没什么特别大的难度。甚至，我们可以认为，**SVG 就是 HTML 的增强版。**



对于可视化来说，SVG 是非常重要的图形系统。它既可以用 JavaScript 操作绘制各种几何图形，还可以作为浏览器支持的一种图片格式，来独立使用 img 标签加载或者通过


Canvas 绘制。即使我们选择使用 HTML 和 CSS、Canvas2D 或者 WebGL 的方式来实现可视化，但我们依然可以且很有可能会使用到 SVG 图像。所以，关于 SVG 我们得好好学。

那这一节课，我们就来聊聊 SVG 是怎么绘制可视化图表的，以及它的局限性是什么。希望通过今天的讲解，你能掌握 SVG 的基本用法和使用场景。

利用 SVG 绘制几何图形

在第 1 节课我们讲过，SVG 属于**声明式绘图系统**，它的绘制方式和 Canvas 不同，它不需要用 JavaScript 操作绘图指令，只需要和 HTML 一样，声明一些标签就可以实现绘图了。

那 SVG 究竟是如何绘图的呢？我们先来看一个 SVG 声明的例子。

 复制代码

```
1 <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
2   <circle cx="100" cy="50" r="40" stroke="black"
3     stroke-width="2" fill="orange" />
4 </svg>
```

在上面的代码中，svg 元素是 SVG 的根元素，属性 xmlns 是 xml 的名字空间。那第一行代码就表示，svg 元素的 xmlns 属性值是"<http://www.w3.org/2000/svg>"，浏览器根据这个属性值就能够识别出这是一段 SVG 的内容了。

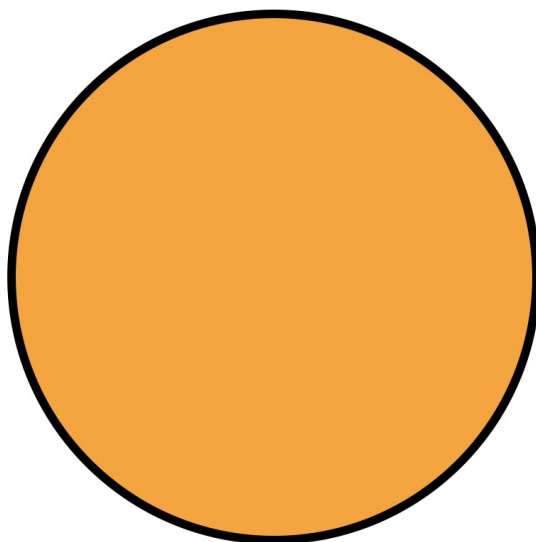
svg 元素下的 circle 元素表示这是一个绘制在 SVG 图像中的圆形，属性 cx 和 cy 是坐标，表示圆心的位置在图像的 x=100、y=50 处。属性 r 表示半径，r=40 表示圆的半径为 40。

以上，就是这段代码中的主要属性。如果仔细观察你会发现，我们并没有给 100、50、40 指定单位。这是为什么呢？

因为 SVG 坐标系和 Canvas 坐标系完全一样，都是以图像左上角为原点，x 轴向右，y 轴向下的左手坐标系。而且在默认情况下，SVG 坐标与浏览器像素对应，所以 100、50、40 的单位就是 px，也就是像素，不需要特别设置。

说到这，你还记得吗？在 Canvas 中，为了让绘制出来的图形适配不同的显示设备，我们要设置 Canvas 画布坐标。同理，我们也可以通过给 svg 元素设置 viewBox 属性，来改变 SVG 的坐标系。如果设置了 viewBox 属性，那 SVG 内部的绘制就都是相对于 SVG 坐标系的了。

好，现在我们已经知道上面这段代码的含义了。那接下来，我们把它写入 HTML 文档中，就可以在浏览器中绘制出一个带黑框的橙色圆形了。



黑色外框的圆形

现在，我们已经知道了 SVG 的基本用法了。总的来说，它和 HTML 的用法基本一样，你可以参考 HTML 的用法。那接下来，我还是以上一节课实现的层次关系图为例，来看看使用 SVG 该怎么实现。

利用 SVG 绘制层次关系图

我们先来回忆一下，上一节课我们要实现的层次关系图，是在一组给出的层次结构数据中，体现出同属于一个省的城市。数据源和前一节课相同，所以数据的获取部分并没有什么差别。这里我就不列出来了，我们直接来讲绘制的过程。

首先，我们要将获取 Canvas 对象改成获取 SVG 对象，方法是一样的，还是通过选择器来实现。

```
1 const svgroot = document.querySelector('svg');
```

然后，我们同样实现 draw 方法从 root 开始遍历数据对象。不过，在 draw 方法里，我们不是像上一讲那样，通过 Canvas 的 2D 上下文调用绘图指令来绘图，而是通过创建 SVG 元素，将元素添加到 DOM 文档里，让图形显示出来。具体代码如下：

```
1 function draw(parent, node, {fillStyle = 'rgba(0, 0, 0, 0.2)', textColor = 'wh'}  
2     const {x, y, r} = node;  
3     const circle = document.createElementNS('http://www.w3.org/2000/svg', 'circle');  
4     circle.setAttribute('cx', x);  
5     circle.setAttribute('cy', y);  
6     circle.setAttribute('r', r);  
7     circle.setAttribute('fill', fillStyle);  
8     parent.appendChild(circle);  
9     ...  
10 }  
11  
12 draw(svgroot, root);  
13  
14
```

从上面的代码中你可以看到，我们使用 **document.createElementNS** 方法来创建 SVG 元素的。这里你要注意，与使用 document.createElement 方法创建普通的 HTML 元素不同，SVG 元素要使用 document.createElementNS 方法来创建。

其中，第一个参数是名字空间，对应 SVG 名字空间 <http://www.w3.org/2000/svg>。第二个参数是要创建的元素标签名，因为要绘制圆型，所以我们还是创建 circle 元素。然后我们将 x、y、r 分别赋给 circle 元素的 cx、cy、r 属性，将 fillStyle 赋给 circle 元素的 fill 属性。最后，我们将 circle 元素添加到它的 parent 元素上去。


接着，我们遍历下一级数据。这次，我们创建一个 SVG 的 g 元素，递归地调用 draw 方法。具体代码如下：

```
1 if(children) {  
2     const group = document.createElementNS('http://www.w3.org/2000/svg', 'g');  
3     for(let i = 0; i < children.length; i++) {
```

```
4     draw(group, children[i], {fillStyle, textColor});
5   }
6   parent.appendChild(group);
7 }
```

SVG 的 `g` 元素表示一个分组，我们可以用它来对 SVG 元素建立起层级结构。而且，如果我们给 `g` 元素设置属性，那么它的子元素会继承这些属性。

最后，如果下一级没有数据了，那我们还是需要给它添加文字。在 SVG 中添加文字，只需要创建 `text` 元素，然后给这个元素设置属性就可以了。操作非常简单，你看我给出的代码就可以理解了。

 复制代码

```
1  else {
2    const text = document.createElementNS('http://www.w3.org/2000/svg', 'text');
3    text.setAttribute('fill', textColor);
4    text.setAttribute('font-family', 'Arial');
5    text.setAttribute('font-size', '1.5rem');
6    text.setAttribute('text-anchor', 'middle');
7    text.setAttribute('x', x);
8    text.setAttribute('y', y);
9    const name = node.data.name;
10   text.textContent = name;
11   parent.appendChild(text);
12 }
```

这样，我们就实现了 SVG 版的层次关系图。你看，它是不是看起来和前一节利用 Canvas 绘制的层次关系图没什么差别？纸上得来终觉浅，你可以自己动手实现一下，这样理解得更深刻。

文设置状态属性，`context.fillStyle` 设置填充颜色，`context.font` 设置元素的字体。我们设置的这些状态，在绘图指令执行时才会生效。

从写法上来看，因为 SVG 的声明式类似于 HTML 书写方式，本身对前端工程师会更加友好。但是，SVG 图形需要由浏览器负责渲染和管理，将元素节点维护在 DOM 树中。这样做的缺点是，在一些动态的场景中，也就是需要频繁地增加、删除图形元素的场景中，SVG 与一般的 HTML 元素一样会带来 DOM 操作的开销，所以 SVG 的渲染性能相对比较低。

那除了写法不同以外，SVG 和 Canvas 还有其他区别吗？当然是有的，不过我要先卖一个关子，我们讲完一个例子再来说。

2. 用户交互实现上的不同

我们尝试给这个 SVG 版本的层次关系图添加一个功能，也就是当鼠标移动到某个区域时，这个区域会高亮，并且显示出对应的省 - 市信息。

因为 SVG 的一个图形对应一个元素，所以我们可以像处理 DOM 元素一样，很容易地给 SVG 图形元素添加对应的鼠标事件。具体怎么做呢？我们一起来看看。

首先，我们要给 SVG 的根元素添加 `mousemove` 事件，添加代码的操作很简单，你可以直接看代码。

 复制代码

```
1  let activeTarget = null;
2  svgroot.addEventListener('mousemove', (evt) => {
3    let target = evt.target;
4    if(target.nodeName === 'text') target = target.parentNode;
5    if(activeTarget !== target) {
6      if(activeTarget) activeTarget.setAttribute('fill', 'rgba(0, 0, 0, 0.2)')
7    }
8    target.setAttribute('fill', 'rgba(0, 128, 0, 0.1)');
9    activeTarget = target;
10  });
11
```

就像是我们熟悉的 HTML 用法一样，我们通过事件冒泡可以处理每个圆上的鼠标事件。然后，我们把当前鼠标所在的圆的颜色填充成 `'rgba(0, 128, 0, 0.1)'`，这个颜色是带透明的

浅绿色。最终的效果就是当我们的鼠标移动到圆圈范围内的时候，当前鼠标所在的圆圈变为浅绿色。你也可以尝试设置其他的值，看看不同的实现效果。

接着，我们要实现显示对应的省 - 市信息。在这里，我们需要修改一下 draw 方法。具体的修改过程，可以分为两步。

第一步，是把省、市信息通过扩展属性 data-name 设置到 svg 的 circle 元素上，这样我们就可以在移动鼠标的时候，通过读取鼠标所在元素的属性，拿到我们想要展示的省、市信息了。具体代码如下：

 复制代码

```
1  function draw(parent, node, {fillStyle = 'rgba(0, 0, 0, 0.2)', textColor = '\n
2    ...
3    const circle = document.createElementNS('http://www.w3.org/2000/svg', 'circle')
4    ...
5    circle.setAttribute('data-name', node.data.name);
6    ...
7
8    if(children) {
9      const group = document.createElementNS('http://www.w3.org/2000/svg', 'g')
10     ...
11     group.setAttribute('data-name', node.data.name);
12     ...
13   } else {
14     ...
15   }
16 }
17
```

第二步，我们要实现一个 getTitle 方法，从当前鼠标事件的 target 往上找 parent 元素，拿到“省 - 市”信息，把它赋给 titleEl 元素。这个 titleEl 元素是我们添加到网页上的一个 h1 元素，用来显示省、市信息。

 复制代码

```
1  const titleEl = document.getElementById('title');
2
3  function getTitle(target) {
4    const name = target.getAttribute('data-name');
5    if(target.parentNode && target.parentNode.nodeName === 'g') {
6      const parentName = target.parentNode.getAttribute('data-name');
7      return `${parentName}-${name}`;
8    }
9  }
```



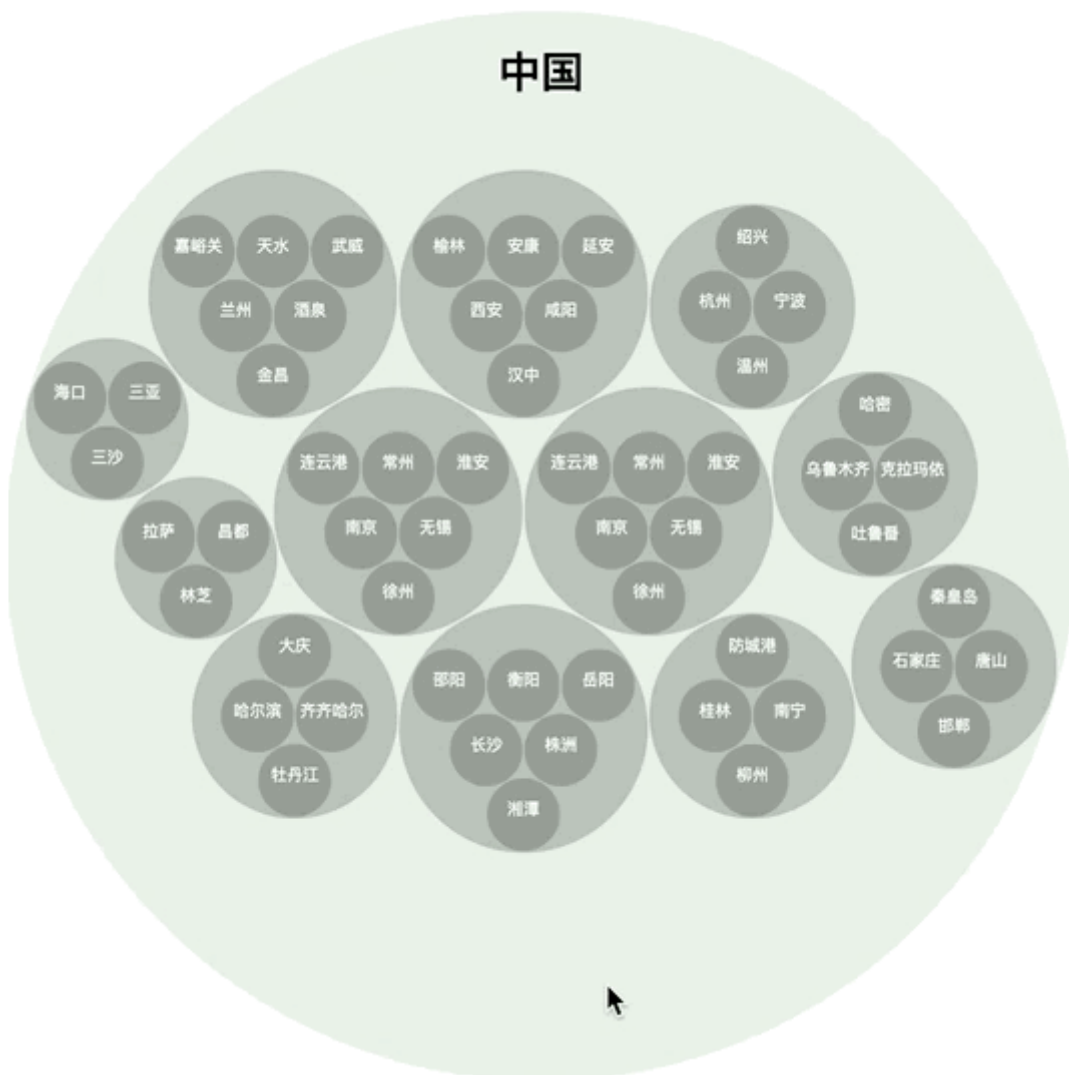
```
9     return name;
10 }
11
```

最后，我们就可以在 `mousemove` 事件中更新 `titleEl` 的文本内容了。

 复制代码

```
1
2  svgroot.addEventListener('mousemove', (evt) => {
3      ...
4      titleEl.textContent = getTitle(target);
5      ...
6  });
```

这样，我们就实现了给层次关系图增加鼠标控制的功能，最终的效果如下图所示，完整的代码我放在 [GitHub 仓库](#) 了，你可以自己去查看。



其实，我们上面讲的鼠标控制功能就是一个简单的用户交互功能。总结来说，利用 SVG 的一个图形对应一个 svg 元素的机制，我们就可以像操作普通的 HTML 元素那样，给 svg 元素添加事件实现用户交互了。所以，SVG 有一个非常大的优点，那就是**可以让图形的用户交互非常简单**。

和 SVG 相比，利用 Canvas 对图形元素进行用户交互就没有那么容易了。不过，对于圆形的层次关系图来说，在 Canvas 图形上定位鼠标处于哪个圆中并不难，我们只需要计算一下鼠标到每个圆的圆心距离，如果这个距离小于圆的半径，我们就可以确定鼠标在某个圆内部了。这实际上就是上一节课我们留下的思考题，相信现在你应该可以做出来了。

但是试想一下，如果我们要绘制的图形不是圆、矩形这样的规则图形，而是一个复杂得多的多边形，我们又该怎样确定鼠标在哪个图形元素的内部呢？这对于 Canvas 来说，就是一个比较复杂的问题了。不过这也不是不能解决的，在后续的课程中，我们就会讨论如何用数学计算的办法来解决这个问题。

绘制大量几何图形时 SVG 的性能问题

虽然使用 SVG 绘图能够很方便地实现用户交互，但是有得必有失，SVG 这个设计给用户交互带来便利性的同时，也带来了局限性。为什么这么说呢？因为它和 DOM 元素一样，以节点的形式呈现在 HTML 文本内容中，依靠浏览器的 DOM 树渲染。如果我们要绘制的图形非常复杂，这些元素节点的数量就会非常多。而节点数量多，就会大大增加 DOM 树渲染和重绘所需要的时间。

就比如说，在绘制如上的层次关系图时，我们只需要绘制数十个节点。但是如果是更复杂的应用，比如我们要绘制数百上千甚至上万个节点，这个时候，DOM 树渲染就会成为性能瓶颈。事实上，在一般情况下，当 SVG 节点超过一千个的时候，你就能很明显感觉到性能问题了。

幸运的是，对于 SVG 的性能问题，我们也是有解决方案的。比如说，我们可以使用虚拟 DOM 方案来尽可能地减少重绘，这样就可以优化 SVG 的渲染。但是这些方案只能解决一部分问题，当节点数太多时，这些方案也无能为力。这个时候，我们还是得依靠 Canvas 和 WebGL 来绘图，才能彻底解决问题。

那在上万个节点的可视化应用场景中，SVG 就真的一无是处了吗？当然不是。SVG 除了嵌入 HTML 文档的用法，还可以直接作为一种图像格式使用。所以，即使是在用 Canvas 和

WebGL 渲染的应用场景中，我们也依然可能会用到 SVG，将它作为一些局部的图形使用，这也会给我们的应用实现带来方便。在后续的课程中，我们会遇到这样的案例。

要点总结

这一节课我们学习了 SVG 的基本用法、优点和局限性。

我们知道，SVG 作为一种浏览器支持的图像格式，既可以作为 HTML 内嵌元素使用，也可以作为图像通过 img 元素加载，或者绘制到 Canvas 内。

而用 SVG 绘制可视化图形与用 Canvas 绘制有明显区别，SVG 通过创建标签来表示图形元素，然后将图形元素添加到 DOM 树中，交给 DOM 完成渲染。

使用 DOM 树渲染可以让图形元素的用户交互实现起来非常简单，因为我们可以直接对图形元素注册事件。但是这也带来问题，如果图形复杂，那么 SVG 的图形元素会非常多，这会导致 DOM 树渲染成为性能瓶颈。

小试牛刀

1. DOM 操作 SVG 元素和操作普通的 HTML 元素几乎没有差别，所以 CSS 也同样可以用于 SVG 元素。那你可以尝试使用 CSS，来设置这节课我们实现的层级关系图里，circle 的背景色和文字属性。接着你也可以进一步想一想，这样做有什么好处？
2. 因为 SVG 可以作为一种图像格式使用，所以我们可以将生成的 SVG 作为图像，然后绘制到 Canvas 上。那如果我们先用 SVG 生成层级关系图，再用 Canvas 来完成绘制的话，和我们单独使用它们来绘图有什么不同？为什么？

欢迎在留言区和我讨论，分享你的答案和思考，也欢迎你把这节课分享给你的朋友，我们下节课见！

源码

[!\[\]\(870f5d5e9c0d57485634be3ecf52f3ca_img.jpg\) 用 SVG 绘制层次关系图和给层次关系图增加鼠标控制的完整代码](#)

跟月影学可视化

系统掌握图形学与可视化核心原理

月影

奇虎 360 奇舞团团长

可视化 UI 框架 SpriteJS 核心开发者



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 02 | 指令式绘图系统：如何用Canvas绘制层次关系图？

下一篇 04 | GPU与渲染管线：如何用WebGL绘制最简单的几何图形？（上）

精选留言 (6)

 写留言



宇宙全栈

2020-06-27

1. 使用 CSS 设置样式的好处：可以将样式和节点解耦，有利于样式的模块化和复用，比如多种主题色，一键换色等。
2. 先用 SVG 生成层级关系图，再用 Canvas 来完成绘制，此时 SVG 将作为一张静态图片被绘制在 Canvas 中。和单独使用 Canvas 绘图相比，这种混合方式代码量更少，代码更加可读，易维护。

展开 ∨

作者回复：赞~ 说得挺好的



5



Geek_3469f6



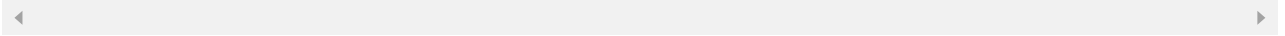
2020-06-26

```
if(target.nodeName === 'text') target = target.parentNode;
```

请问，这句代码是不是有问题？找到文本节点的父节点，是group节点。设置了fill，有什么用处？

展开 ▾

作者回复: 对，看得非常仔细，这里的确是有问题，应该要设置这个节点之前的那个circle节点而不是group节点，你可以试着把github示例代码里面这块修复一下，然后提交一个pr



💬 1

👍 2



Geek_07ad60

2020-06-29

还是不太理解声明式和指令式，请老师再详解一下，谢谢！

💬

👍



快乐小球球

2020-06-28

关于svg，我补充一点，如今微信公众号互动图文支持使用svg制作交互和动画，但不支持js语言，因此掌握原生svg语言非常重要，并不是任何场景都可以使用js来声明和调用该系统的。

作者回复: 很棒



💬 1

👍



特异型大光头

2020-06-26

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">...
```

展开 ▾

作者回复: document.createElementNS('http://http://www.w3.org/2000/svg', 'circle');

这里写错了，应该是http://www.w3.org/2000/svg，你多写了一个http:

1



姚凯伦

2020-06-26

最近也遇到过用svg绘制元素太多导致渲染卡顿问题，后来改用使用一条 path 绘制所有元素，但是 path 一长也同样存在渲染卡顿问题，不知道老师有没有遇到过类似的问题

作者回复: 是的，path长导致图形绘制得太复杂也会有问题，具体要看你的数据量，如果达到瓶颈，不太好彻底解决，只能换成canvas2d或者webgl。在后续课程性能篇中会有详细的介绍。

