SVG 入门

什么是SVG

SVG 是一种基于 XML 语法的图像格式，全称是可缩放矢量图（Scalable Vector Graphics）。其他图像格式都是基于像素处理的，SVG 则是属于对图像的形状描述，所以它本质上是文本文件，体积较小，且不管放大多少倍都不会失真。

SVG的优劣

优点

- SVG 可被非常多的工具读取和修改（比如记事本）
- SVG 与 JPEG 和 GIF 图像比起来，尺寸更小，且可压缩性更强。
- SVG 是可伸缩的
- SVG 图像可在任何的分辨率下被高质量地打印
- SVG 可在图像质量不下降的情况下被放大
- SVG 图像中的文本是可选的，同时也是可搜索的（很适合制作地图）
- SVG 可以与 Java 技术一起运行
- SVG 是开放的标准
- SVG 文件是纯粹的 XML

缺点

但是人无完人,也没有绝对的好标准.相对于png来讲,如果图片特别复杂,SVG所需要的大小是远远大于jpg等的.

为了证明网上的观点,我从维基百科上下载了一份莫奈的<印象 日出>,你可以点击这里,你会发现,作为jpg版本的图片大小只有1.8mb,而转换成的SVG图片,大小足足为4.9mb.这是由于是svg其本质是利用向量(Vector)来描述图片,对于简单的图片而言,其可以使用很少的向量来描述,所以大小优于使用像素来描述的其他格式.

补充: 在仔细看了各个jpg转SVG的网站,我发现实质上jpg与SVG的转换是指将图片转为base64,所以这种证明的办法似乎存在错误

过时的缺点:

有很多前辈学习SVG时,各大浏览器对其的支持度还不容乐观,但是在今天(2019年1.22),它的支持度已经达到95%以上了.你可以点击下面的链接查看支持度:

支持度

SVG使用方式

（1）使用 `<embed>` 标签 **(不推荐)**

优势：所有主要浏览器都支持，并允许使用脚本 缺点：不推荐在HTML4和XHTML中使用（但在HTML5允许）
示例：

```
<embed width="300px" height="300px" src="img/demo.svg" type="image/svg+xml" />
```

主要是为了把所有形式都写出来,但是需要注意的是,大多数现代浏览器已经弃用并取消了对浏览器插件的支持，所以如果您希望您的网站可以在普通用户的浏览器上运行，那么依靠 <embed> 通常是不明智的。

（2）使用 <object> 标签

HTML <object> 元素（或者称作 HTML 嵌入对象元素）表示引入一个外部资源

优势：所有主要浏览器都支持，并支持HTML4，XHTML和HTML5标准 缺点：不允许使用脚本。 示例：

```
<object width="300px" height="300px" data="img/demo.svg" type="image/svg+xml">
</object>
```

（3）使用 <iframe> 标签

优势：所有主要浏览器都支持，并允许使用脚本 缺点：不推荐在HTML4和XHTML中使用（但在HTML5允许） 示例：

```
<iframe width="300px" height="300px" src="img/demo.svg"></iframe>
```

（4）直接在HTML嵌入SVG代码

示例：

```
<svg width="500px" height="500px" style="margin:50px;" version="1.1"
xmlns="http://www.w3.org/2000/svg">
    <rect x="20" y="20" rx="10" ry="10" width="300" height="300"
style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0);fill-opacity:0.1;stroke-
opacity:0.9;opacity:0.9;"/>
</svg>
```

（5）使用<img>标签

示例：

```
<img src="img/demo.svg" width="300px" height="300px"/>
```

（6）链接到svg文件

示例：

```
<a href="img/demo.svg">查看svg</a>
```

（7）在css中使用

示例：

```
.container{
  background: white url(img/demo.svg) repeat;
}
```

各个方式SVG支持列表

SVG 格式	支持列表
inline SVG	支持资源外链 支持CSS 支持JS
img SVG	不支持资源外链 支持内部CSS 不支持JS
background-img SVG	不支持资源外链 支持内部CSS 不支持JS
background-img BASE64 SVG	不支持资源外链 支持内部CSS 不支持JS
object SVG	支持资源外链 支持内部CSS 支持内部JS
embed SVG	支持资源外链 支持内部CSS 支持内部JS
iframe SVG	支持资源外链 支持内部CSS 支持内部JS

有了这个表, 你会发现几乎在所有情况下. SVG都支持内部CSS. 即在SVG内部写 style 标签定义其自身的样式. (注意: inline SVG 的 style 标签会污染外部 HTML 的 style)

SVG文件探秘

这里参考自菜鸟教程,涉及到很多XML的内容

一个简单的SVG图形例子：

这里是SVG文件（SVG文件的保存与SVG扩展）：

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

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

SVG 代码解析：

第一行包含了 XML 声明。请注意 standalone 属性！该属性规定此 SVG 文件是否是"独立的"，或含有对外部文件的引用。

standalone="no" 意味着 SVG 文档会引用一个外部文件 - 在这里，是 DTD 文件。

第二和第三行引用了这个外部的 SVG DTD。该 DTD 位于 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"。该 DTD 位于 W3C，含有所有允许的 SVG 元素。

SVG 代码以`<svg>`元素开始，包括开启标签 `<svg>` 和关闭标签 `</svg>` 。这是根元素。width 和 height 属性可设置此 SVG 文档的宽度和高度。version 属性可定义所使用的 SVG 版本，xmlns 属性可定义 SVG 命名空间。

SVG 的 `<circle>` 用来创建一个圆。cx 和 cy 属性定义圆中心的 x 和 y 坐标。如果忽略这两个属性，那么圆点会被设置为 (0, 0)。r 属性定义圆的半径。

stroke 和 stroke-width 属性控制如何显示形状的轮廓。我们把圆的轮廓设置为 2px 宽，黑边框。

fill 属性设置形状内的颜色。我们把填充颜色设置为红色。

关闭标签的作用是关闭 SVG 元素和文档本身。

注释：所有的开启标签必须有关闭标签！

SVG标签

这些是常用的标签

- text: 创建一个 text 元素
- circle: 创建一个圆
- rect: 创建一个矩形
- line: 创建一条线
- path: 在两点之间创建一条路径
- textPath: 在两点之间创建一条路径，并创建一个链接文本元素
- polygon: 允许创建任意类型的多边形
- g: 单独的元素
    - <g> 元素g是用来组合对象的容器。添加到g元素上的变换会应用到其所有的子元素上。添加到g元素的属性会被其所有的子元素继承。有点像div的感觉

这么看起来似乎SVG很简单,但是随着慢慢学习,发现单是一个path就够我搞得了.

这里甚至还涉及到了很多数学知识.

所幸找到了一个台湾大佬写的博客SVG研究之路我在这个专栏获益匪浅.

path

这是暂时最难的一个部分,它的d属性有很多部分

指令	参数	指令说明
M	xy	起始点的x , y 座标(move to)
L	xy	从目前点的座标画直线到指定点的x , y 座标(line to)
H	x	从目前点的座标画水平直线到指定的x 轴座标(horizontal line to)
V	y	从目前点的座标画垂直线到指定的y 轴座标(vertical line to)
C	x1 y1 x2 y2 xy	从目前点的座标画条贝兹曲线到指定点的x, y 座标：其中x1, y1 及x2, y2 为控制点(curve)
S	x2 y2 xy	从目前点的座标画条反射的贝兹曲线到指定点的x, y 座标：其中x2, y2 为反射的控制点(smooth curve)
Q	x1 y1 xy	从目前点的座标画条二次贝兹曲线到指定点的x, y 座标：其中x1, y1 为控制点(quadratic Bézier curve)
T	xy	从目前点的座标画条反射二次贝兹曲线到指定点的x, y 座标：以前一个座标为反射控制点(smooth quadratic Bézier curve)
A	rx ry x-axis-rotation large-arc-flag sweep-flag xy	从目前点的座标画个椭圆形到指定点的x, y 座标：其中rx, ry 为椭圆形的x 轴及y 轴的半径，x-axis-rotation 是弧线与x 轴的旋转角度，large- arc-flag 则设定1 最大角度的弧线或是0 最小角度的弧线，sweep-flag 设定方向为1 顺时针方向或0 逆时针方向(Arc)
Z		关闭路径，将目前点的座标与第一个点的座标连接起来(closepath)

光看怎么能记得住,尝试写一个demo吧.

预览效果

源代码

贝塞尔曲线

直接看demo,这个实际上就是一个调整曲线的过程

demo

深入理解弧线

这里似乎是SVG中最难理解的部分,但是对于数学专业似乎很好理解~

首先我们先来看一个普通的A参数长什么样子

```
<path d="M50 50 A50 10,0 0 0 100 0" stroke="#f00" fill="none"/>
```

- rx ： 椭圆的x 轴半径(根据不同的终点换算成比例)

- ry： 椭圆的y 轴半径(根据不同的终点换算成比例)
- x-axis-rotation： 弧线与x 轴的夹角
- large-arc-flag： 1 为大角度弧线，0 为小角度弧线(必须有三个点)
- sweep-flag： 1 为顺时针方向，0 为逆时针方向
- x： 终点x 座标
- y： 终点y 座标

这里的 rx 和 ry很好理解,就是椭圆的两个轴长.

而x , y 不但是终点,也是椭圆上的一个点.

M是起点,这个也不需要多说.

所以抛开中间三个看不懂的参数,我们现在已经有了椭圆的长轴长度rx短轴长度ry,还有起点M和终点(x,y),

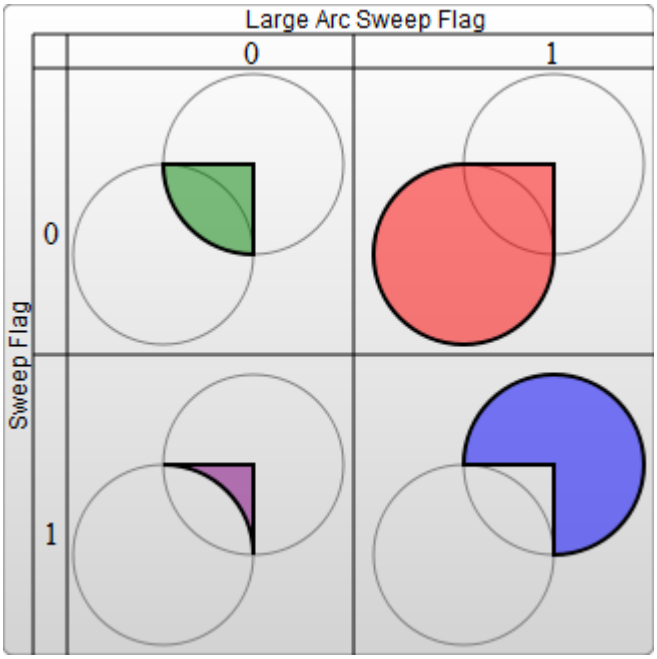
已知起点和终点都在椭圆上,我们就可以套用椭圆的一般方程确定位置.

我从维基百科上找到了椭圆的一般方程定义.

中心位于点(h,k)的主轴平行于 x 轴的椭圆由如下方程指定

$$x=h+a\cos t$$ $$y=k+a\sin t$$

这里需要特别注意的是 **平行于 x 轴** ,而我们根据前文的信息,很难确定椭圆的主轴是否平行于X轴,这就引出了我们的第三个参数,x-axis-rotation,它代表弧线于x轴的夹角.也就是主轴相对于x轴的偏移量. 后面的两个就更简单了,一张图片就能很好诠释



后记: 后来还找到了一个特别详细的文章,SVG之旅：路径

在这里遇到的问题: 这个H和V的单位是什么?

最开始在纠结H和V的单位是什么,后来释然了,因为H和V的单位其实就相当于L的横纵坐标,在W3C里也是这样写的.这也是SVG与其他格式的本质区别,SVG是一个标准的笛卡尔坐标系,在历来的数学学习中,从来没有一个坐标系有单位.只有现实应用的时候,会产生单位的对应.SVG相当于对图像的抽象.

附:W3C原文:

```
When a relative l command is used, the end point of the line is (cpx + x, cpy +
y).

When a relative h command is used, the end point of the line is (cpx + x, cpy).
This means that an h command with a positive x value draws a horizontal line in
the direction of the positive x-axis.

When a relative v command is used, the end point of the line is (cpx, cpy + y).
```

一些基础练习

text的一些属性练习:

预览

代码

fill的一些属性练习:

预览

代码

后续练习

练习1. 根据表格画出折线图

年龄	身高
14	150
15	156
16	161
17	168
18	170

在编写这个练习的时候,需要特别注意: 在使用js或jQuery操作SVG的元素时,不能直接使用createElement来构建元素.因为对于SVG而言,创建SVG元素需要指定命名空间，就像需要在svg标签上设定xmlns为`http://www.w3.org/2000/svg`。正确的构造方式是调用createElentNS()方法，并将”`http://www.w3.org/2000/svg`”作为第一参数传入。

W3C是这样写的

```
When SVG is parsed as a XML, for compliance with the Namespaces in XML
Recommendation [xml-names], an SVG namespace declaration must be provided so that
all SVG elements are identified as belonging to the SVG namespace.
```

关于更多的使用JS操作SVG的方法可以参考这里

JavaScript操作SVG的一些知识

最后实现的效果就是一个简单的折线图.

实际效果

源代码

练习2. 根据表格画出饼状图

我们都知道饼状图是为了突出数据之间的比例.

我们暂且捏造一份数据,用它来制作一份饼状图

佩奇午餐种类	占比
三明治	10%
汉堡包	20%
蛋炒饭	30%
番茄炒蛋	40%

这里需要处理的核心问题就是如何在圆上选点,这里只需要借助椭圆的一般表达式就好啦.

实际效果 源代码

练习3. 实现可以随意拖拽的贝塞尔曲线

因为贝塞尔曲线有些理解困难,我制作了一个可以随意拖拽的贝塞尔曲线,只要拖动点P0就可以方便的看到曲线的变化,这里只是用了一些简单的js方法.

实际效果 源代码

练习4. 练习并阅读一些动画代码

仓库地址

一些参考链接 :

jenkov

数据可视化：你想知道的经典图表全在这