

# 第三章 SVG

# 内容安排

- 3.1 **SVG**概述
- 3.2 使用**SVG**创建交互式应用
- 3.3 课后思考
- 3.4 小结

## 3.1 SVG概述

- 理解SVG
- 可缩放图形
- 使用SVG创建2D图形
- 在页面中添加SVG
- 简单的形状
- 变换SVG元素
- 复用内容
- 图案和渐变

## 3.1 SVG概述

- SVG路径
- 使用SVG文本
- 组合场景

# □常用的栅格文件格式

表：常用的栅格文件和矢量文件格式

栅格文件		矢量文件	
文件格式	文件命名规则	文件格式	文件命名规则
TIFF	.tif	AutoCAD DXF	.dxf / .dwg
JFIF	.jpg	MapInfo TAB	.tab
ERDAS Imagine	.img	MapInfo MIF/MID	.mif / .mid
Band Interleaved by Line	.bil	Shapefile	.shp / .shx / .dbf
Band Interleaved by Pixel	.bip	ESRI interchange file	.e00

# □常用的栅格文件格式

表：常用的栅格文件和矢量文件格式（续）

栅格文件		矢量文件	
文件格式	文件命名规则	文件格式	文件命名规则
Band Sequential	.bsq	Simple ASCII Format	
SGS—DEM	.dem	Point / Line / Polygon Coverage	〈目录结构〉
ArcInfo Binary Grid Format	〈目录结构〉	Geodatabase	各种数据库

## 常用的栅格文件格式

文件格式	典型数据	位	压缩情况	文件命名规则
TIFF	扫描、图形、单波段、多波段	1-8、16、24、32	压缩或不压缩	.tif
JFIF	单波段、多波段	24	压缩	.jpg
ERDAS Imagine	单波段和多波段	1、2、4、8、16、32	不压缩	.img
Band Interleaved by Line	分类和连续图、多波段	1、4、8、16、32	不压缩	.bil
Band Interleaved by Pixel	分类和连续图、多波段	1、4、8、16、32	不压缩	.bip
Band Sequential	分类和连续图、多波段	1、4、8、16、32	不压缩	.bsq
ArcInfo Binary	分类和连续图	32	压缩或不压缩	<目录结构>



## 常用的矢量文件格式

文件格式			文件命名规则
AutoCAD DXF			.dxf / .dwg
MapInfo MIF/MID			.tab
Mapinfo			.mif / .mid
Shapefile			.shp / .shx / .dbf
KML File			.kml
			.gpx
ESRI interchange file			.e00
Simple ASCLL Format			
Point / Line / Polygon Coverage			<目录结构>



## 3.1 SVG概述

### ➤ 栅格图形

图像由一组二维像素网格表示，每一个像素表示为一个栅格单元，放大到一定程度就失真，显示为像素单元格。PNG和JPEG是两种栅格图形的格式。

### ➤ 矢量图形

图像用数学描述的几何形状表示，不失真。SVG是一种矢量图形的格式。

# □ SVG历史

## ➤ SVG标准制定开发历史

- 2001年9月4日，发布SVG 1.0。
- 2003年1月4日，发布SVG 1.1。
- 2003年1月14日，推出SVG移动子版本：SVG Tiny 和SVG Basic。
- 2008年12月22日，发布SVG Tiny 1.2。
- 2011年8月16日，发布SVG 1.1（第2版），成为W3C当前推荐的标准。
- W3C当前仍正在研究制定SVG 2。

# □ 理解SVG

## ➤ 什么是SVG

- SVG(Scalable Vector Graphics)指伸缩矢量图形
- SVG用于定义网络中基于矢量的图形
- SVG使用 XML 格式定义图形
- SVG图像在放大或改变尺寸的情况下其图形质量不会有损失
- SVG是万维网联盟的标准

# □ 理解SVG

## ➤ SVG的优势

与其他图像格式相比（比如 **JPEG** 和 **GIF**），使用 **SVG** 的优势在于：

- **SVG** 图像可通过文本编辑器来创建和修改
- **SVG** 图像可被搜索、索引、脚本化或压缩
- **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



# □ 理解SVG

## ➤ SVG与Canvas的区别

### ● SVG

- SVG是一种使用XML描述2D图形的语言。
- SVG基于XML，可以为元素附加JavaScript 事件处理器。
- 在SVG中，每个被绘制的图形均被视为对象。如果SVG对象的属性发生变化，那么浏览器能够自动重现图形。

# □ 理解SVG

## ➤ SVG与Canvas的区别

### ● Canvas

- Canvas通过JavaScript来绘制2D图形。
- Canvas是逐像素进行渲染的。
- 在Canvas中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

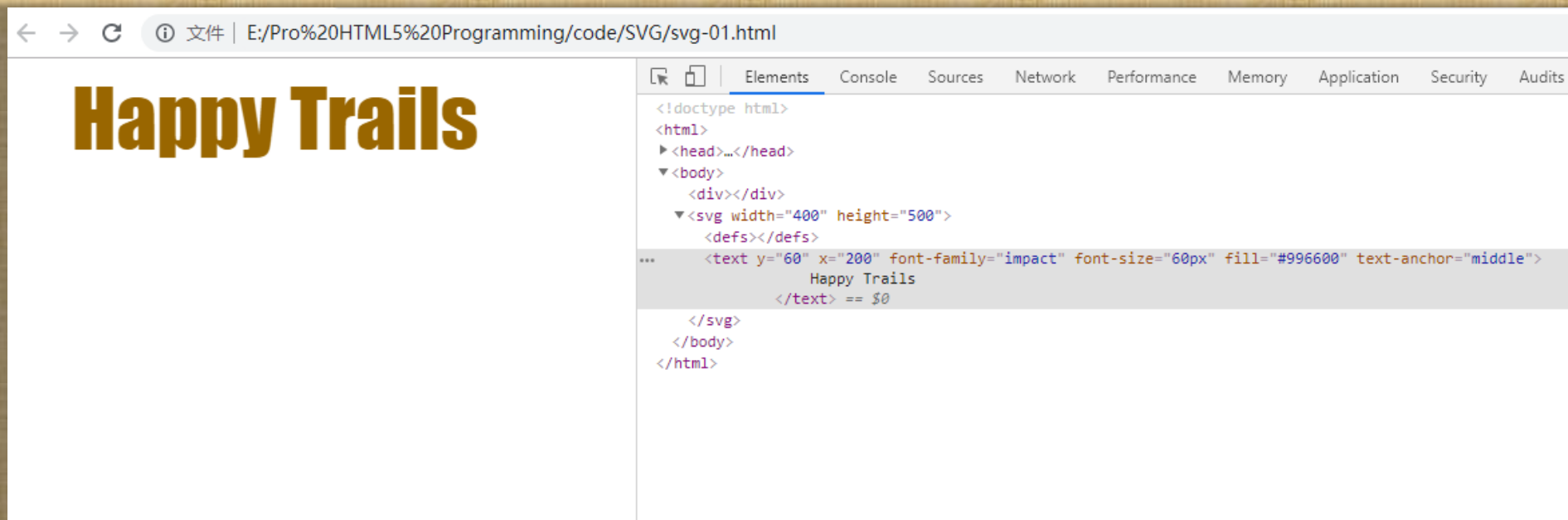
# □ 理解SVG

	SVG保留模式	Canvas即时模式
分辨率	不依赖分辨率	依赖分辨率
事件处理器	支持事件处理器	不支持事件处理器
渲染能力	最适合带有大型渲染区域的应用程序（比如谷歌地图），复杂高度会减慢渲染速度（任何过度使用 <b>DOM</b> 的应用都不快）	弱的文本渲染能力
保存	不能保存	能够以.png或.jpg格式保存结果图像
应用	不适合游戏应用	最适合图像密集型的游戏，其中的许多对象会被频繁重绘



# □ 理解SVG

在浏览器的开发工具中能够查看和编辑SVG结构。



**注释：**在开发环境中，可以添加、删除以及编辑SVG元素。修改的结果会立即显示在页面上。非常便于调测。

# □ 可缩放图形

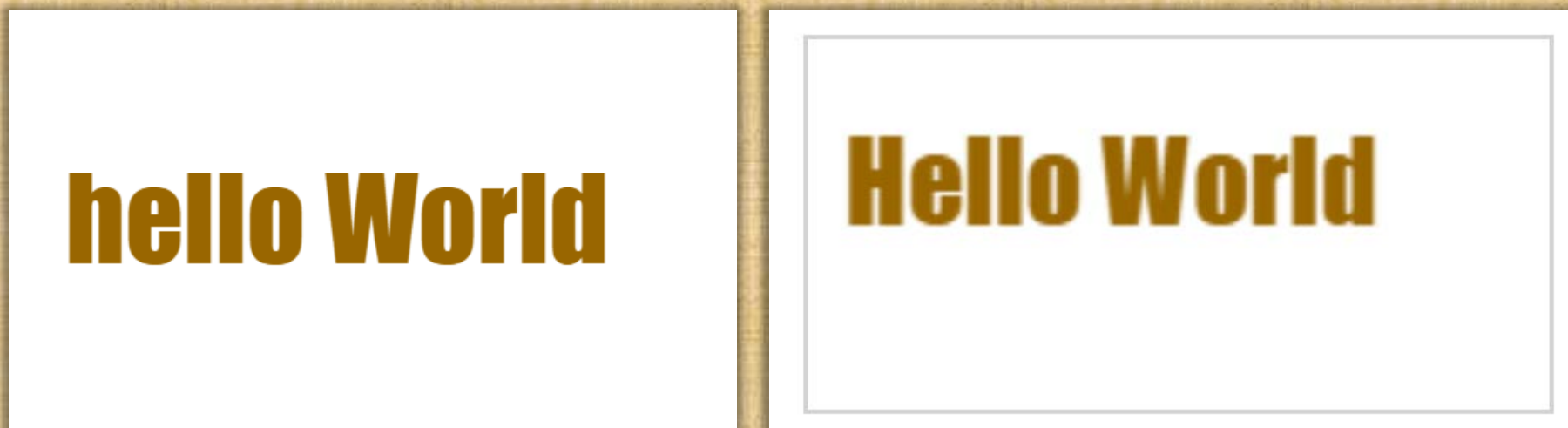
## ➤ SVG

放大、旋转或者用其他手段变换SVG内容的时候，渲染程序会立即重绘，使SVG不行质量不下降。

## ➤ Canvas

基于像素的，所以放大后图像会变得模糊，颗粒感强。

## □ 可缩放图形



图：SVG 图像和Canvas图像放大后的效果

# □ 使用SVG创建2D图形

- **SVG 形状**

矩形、圆形、椭圆、线条、多边形、折线、路径

- **SVG 滤镜**

可用来增加对SVG图形的特殊效果。

- **SVG 渐变**

线性渐变、放射渐变

# □ 在页面中添加SVG

- 使用内联SVG



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>使用内联SVG</title>
</head>
<body>
  <svg width="200" height="200" style="background-color: red;">
    <rect x="10" y="20" width="100" height="80" stroke="green" fill="white"></rect>
  </svg>
</body>
</html>
```



## □ 在页面中添加SVG

- 嵌入**SVG**文件到**HTML**文档（.svg）

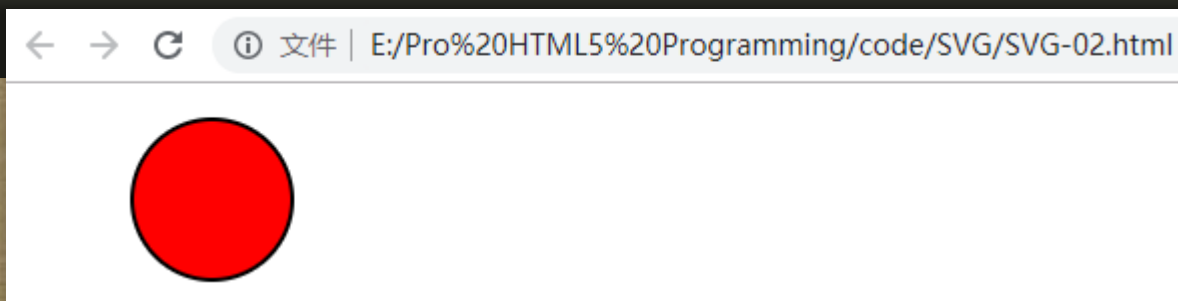
该属性规定此SVG文件是否是“独立的”，或含有对外部文件的应用。standalone="no"意味着SVG文档会引用一个外部文档。

```
<?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">
```

该DTD位于W3C，含有所有允许的SVG元。

```
<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>
```

xmlns属性可以定义SVG命名空间。



## □ 在页面中添加SVG

- **SVG**文件可通过以下标签嵌入**HTML**文档：

➤ `<embed>`、`<object>`或者`<iframe>`

(1) 使用`<embed>`标签

**优势：**所有主要浏览器都支持，并允许使用脚本。

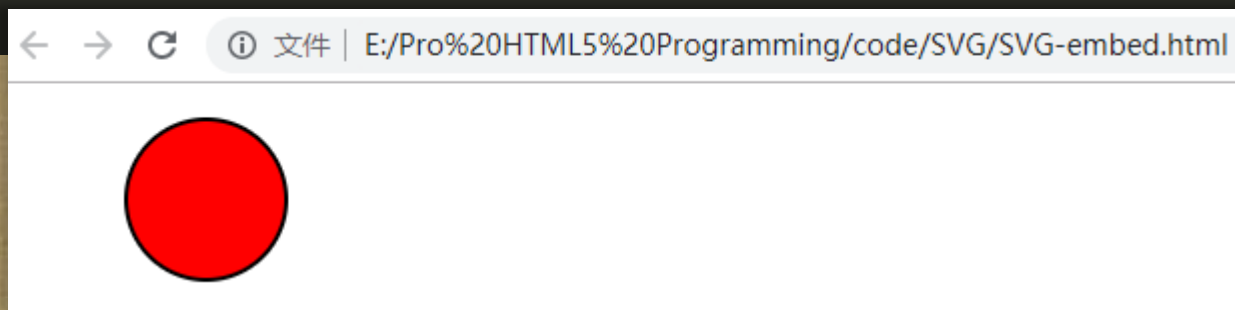
**缺点：**不推荐在HTML4和XHTML中使用（但在HTML5允许）。



## □ 在页面中添加SVG

- 使用<embed> 标签

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>使用embed标签</title>
</head>
<body>
  <embed src="circle.svg" width="300" height="100" type="image/svg+xml" ></embed>
</body>
</html>
```



## □ 在页面中添加SVG

### (2) 使用<object>标签

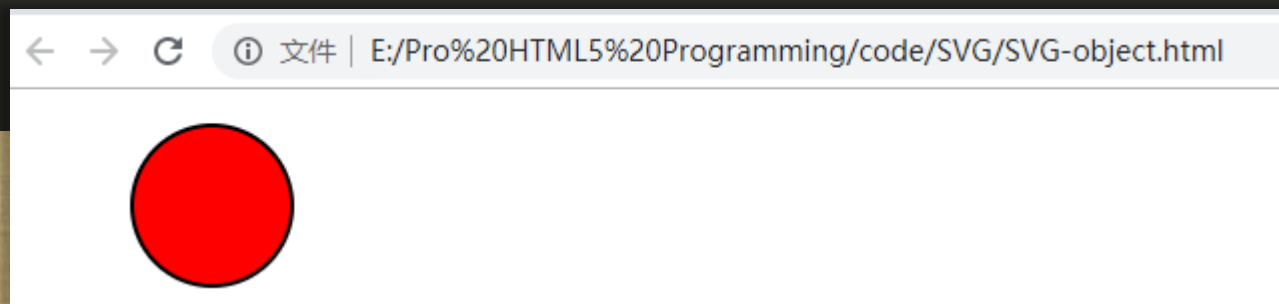
**优势：**所有主要浏览器都支持，并支持HTML4，XHTML和HTML5标准

**缺点：**不允许使用脚本。

## □ 在页面中添加SVG

- 使用<object>标签

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>使用object标签</title>
</head>
<body>
  <object data="circle.svg" type="image/svg+xml"></object>
</body>
</html>
```



## □ 在页面中添加SVG

### (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）。

## □ 在页面中添加SVG

- 使用<iframe>标签

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>使用object标签</title>
</head>
<body>
  <iframe src="circle.svg"></iframe>
</body>
</html>
```



## □ 简单的形状

- SVG语言包含了基本的形状元素，如矩形、圆形、椭圆、直线和多边形等。
- 形状元素的尺寸和位置被定义成了属性。
- SVG绘制形状对象时是按对象在文档中出现的顺序进行的。



## □ 简单的形状

表：创建简单形状的元素

元素	描述
<rect>	可用来创建矩形，以及矩形的变种
<circle>	可用来创建一个圆
<ellipse>	用来创建一个椭圆
<line>	用来创建一个直线
<polyline>	用于创建任何只有直线的形状
<polygon>	用来创建含有不少于三个边的图形

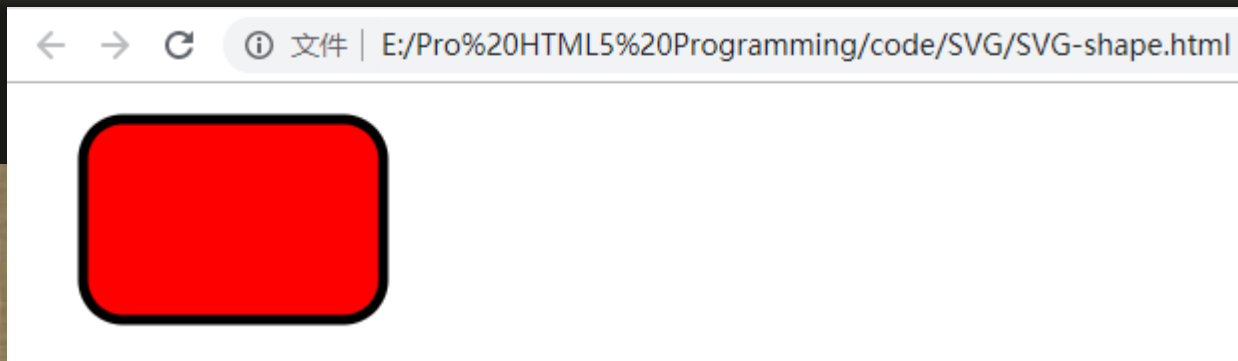


# □ 简单的形状

- 创建矩形--<rect>标签

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>矩形</title>
</head>
<body>
  <svg width="200" height="200" xmlns="http://www.w3.org/2000/svg" version="1.1">
    <rect x="30" y="10" rx="20" ry="20" width="150" height="100"
      style="fill:red;stroke:black;stroke-width:5;" />
  </svg>
</body>
</html>
```

- rect 元素的 width 和 height 属性可定义矩形的高度和宽度。
- style 属性用来定义 CSS 属性。
- x和y属性可定义矩形的左侧和顶端位置。
- rx 和 ry 属性可使矩形产生圆角。

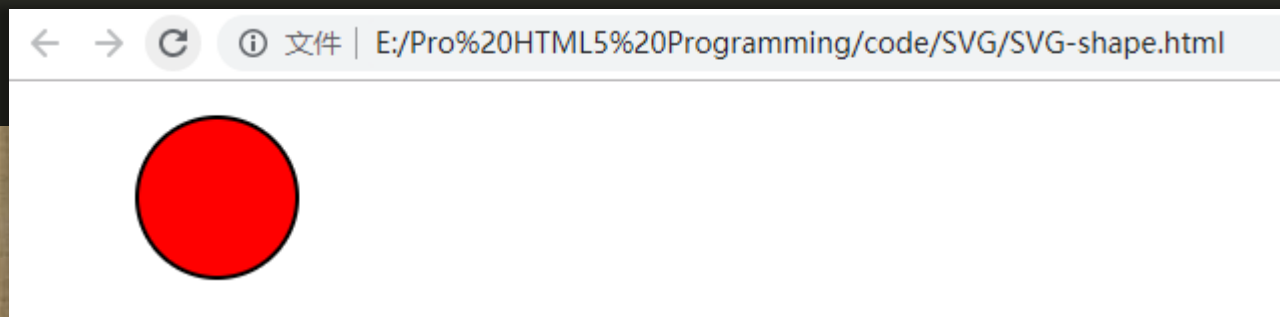


# □ 简单的形状

- 创建圆形--<circle> 标签

- cx和cy属性定义圆点的x和y坐标。如果省略cx和cy，圆的中心会被设置为(0, 0)
- r属性定义圆的半径

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>圆形</title>
</head>
<body>
  <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>
</body>
</html>
```

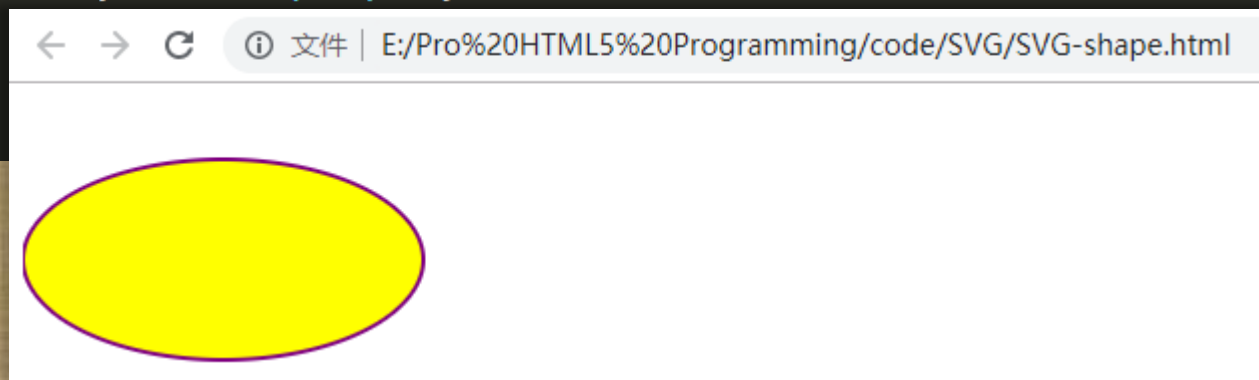


# □ 简单的形状

- 创建椭圆--<ellipse> 标签

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>椭圆</title>
</head>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <ellipse cx="100" cy="80" rx="100" ry="50"
      style="fill:yellow;stroke:purple;stroke-width:2" />
  </svg>
</body>
</html>
```

- cx属性定义的椭圆中心的x坐标。
- cy属性定义的椭圆中心的y坐标。
- rx属性定义的水平半径。
- ry属性定义的垂直半径。

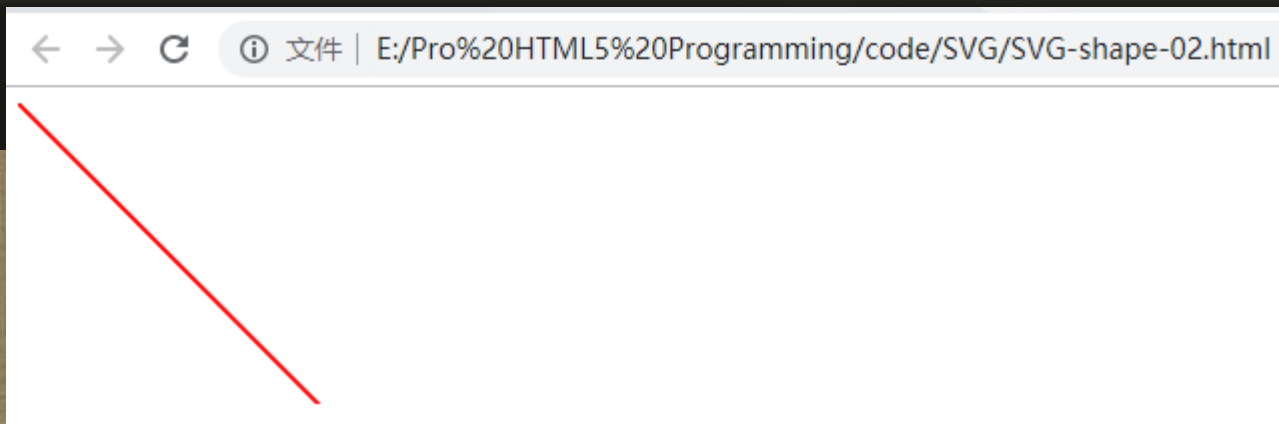


# □ 简单的形状

- 绘制直线--<line>标签

- x1 属性在 x 轴定义线条的开始。
- y1 属性在 y 轴定义线条的开始。
- x2 属性在 x 轴定义线条的结束。
- y2 属性在 y 轴定义线条的结束。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>直线</title>
</head>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <line x1="0" y1="0" x2="200" y2="200" style="stroke:#f00;stroke-width:2" />
  </svg>
</body>
</html>
```

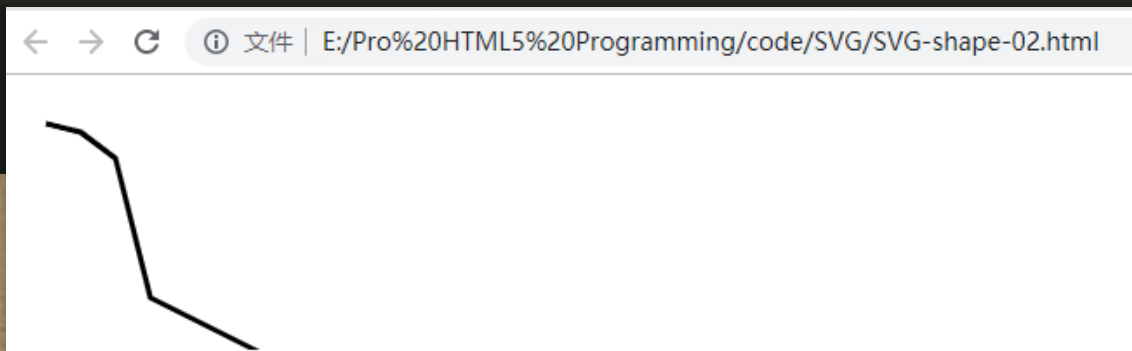


# □ 简单的形状

- 绘制曲线--<polyline> 标签

- points属性定义用来画一个<polyline> 元素的点的数列。
- 每个点用用户坐标系统中的一个X坐标和Y坐标定义。
- 用逗号分开每个点的X和Y坐标。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>曲线</title>
</head>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <polyline points="20,20 40,25 60,40 80,120 120,140 200,180"
      style="fill:none; stroke:black; stroke-width:3" />
  </svg>
</body>
</html>
```



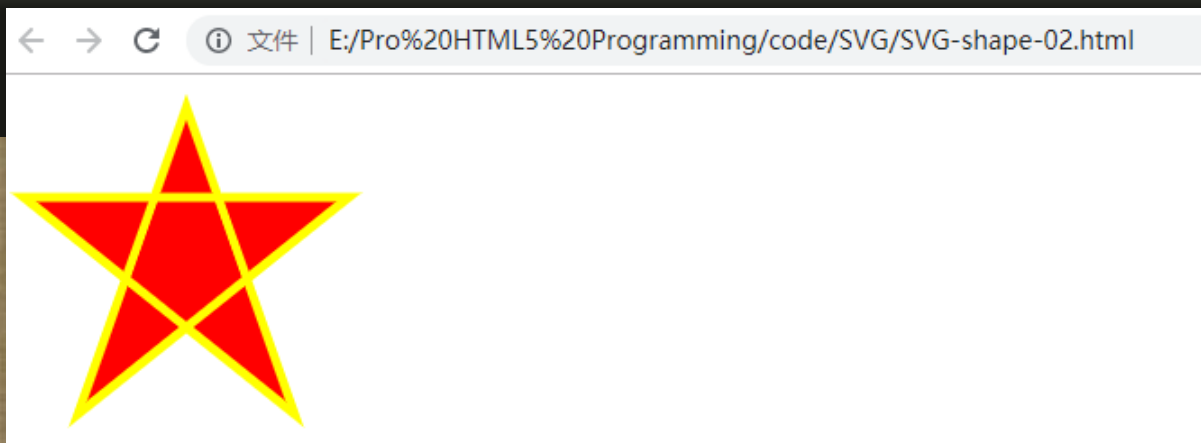


# □ 简单的形状

- 绘制多边形--<polygon>标签

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>多边形</title>
</head>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="200">
    <polygon points="100,10 40,180 190,60 10,60 160,180"
      style="fill:red; stroke:yellow; stroke-width:5; fill-rule:nonzero;"/>
  </svg>
</body>
</html>
```

- points 属性定义多边形每个角的 x 和 y 坐标。

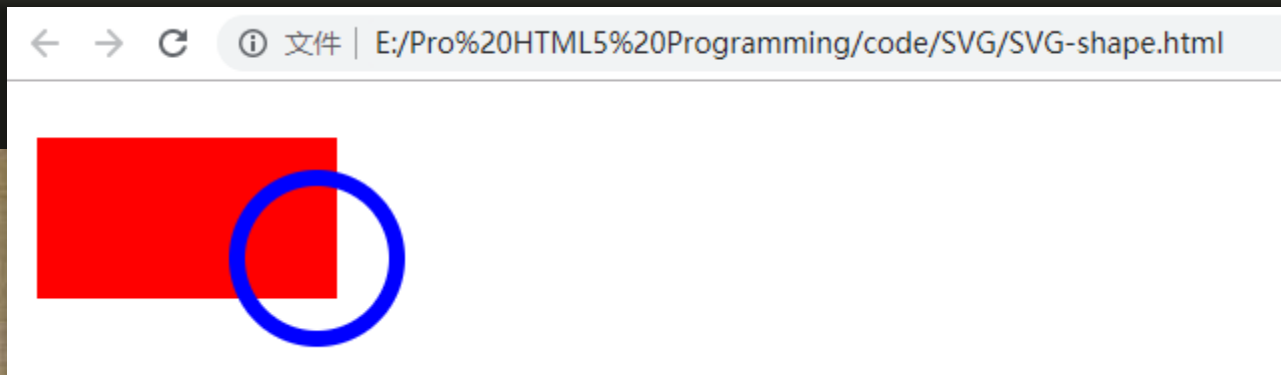


# □ 简单的形状

- **SVG**绘制多个形状

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>SVG绘制多个形状</title>
</head>
<body>
  <svg width="200" height="200">
    <rect x="10" y="20" width="150" height="80" stroke="red" fill="#f00"/>
    <circle cx="150" cy="80" r="40" stroke="#00f" stroke-width="8" fill="none" />
  </svg>
</body>
</html>
```

- SVG绘制形状对象时是按对象在文档中出现的顺序进行的。
- 在这里，圆形显示在矩形上。





## □ 变换SVG元素

- 有效的**SVG变换**有：旋转、缩放、移动和倾斜。
- transform属性中使用的变换函数类似于CSS的transform属性使用的CSS变换函数，除了参数不同。

表： transform属性的变换函数

变换函数	描述
translate(tx, ty)	声明水平和垂直移动值
scale(sx, sy)	声明水平和垂直缩放值
rotate(<rotate-angle>, [<cx, cy>])	对于给定的点和旋转角度执行旋转
matrix(a, b, c, d, e, f)	通过一个有6个值的变换矩阵声明一个变换

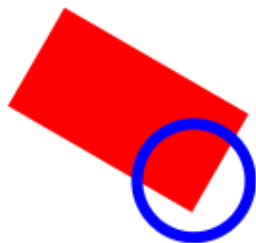
# □ 变换SVG元素

- 变换SVG元素

- transform属性用来对一个元素声明一个或多个变换。
- 它输入一个带有顺序的变换定义列表的<transform-list>值。
- 每个变换定义由空格或逗号隔开。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>变换SVG元素</title>
</head>
<body>
  <svg width="200" height="200">
    <g transform = "translate(60,0) rotate(30) scale(0.75)" id="ShapeGroup">
      <rect x="10" y="20" width="150" height="80" stroke="red" fill="#f00"/>
      <circle cx="150" cy="80" r="40" stroke="#00f" stroke-width="8" fill="none" />
    </g>
  </svg>
</body>
</html>
```

← → ↻ ⓘ 文件 | E:/Pro%20HTML5%20Programming/code/SVG/SVG-transform.html



## □ 复用内容

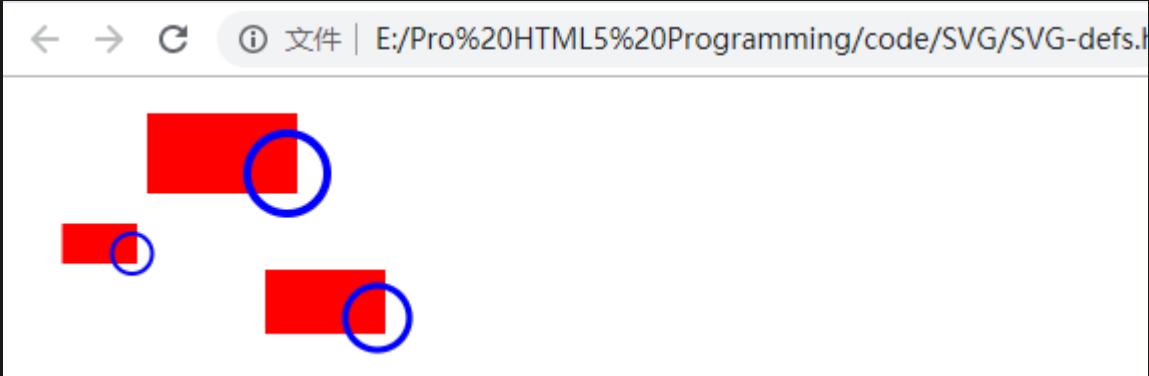
- `<defs>` 元素用来定义留待将来使用的内容。
- `<use>` 元素用于链接到 `<defs>` 定义的内容。
- 借助这两个元素，可以多次复用同一个内容，消除冗余。

# □ 复用内容

- 复用内容

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>SVG复用内容</title>
</head>
<body>
  <svg width="200" height="200">
    <defs>
      <g id="ShapeGroup">
        <rect x="10" y="20" width="150" height="80" stroke="red" fill="#f00"/>
        <circle cx="150" cy="80" r="40" stroke="#00f" stroke-width="8" fill="none" />
      </g>
    </defs>

    <use xlink:href="#ShapeGroup" transform="translate(60,0) scale(0.5)"/>
    <use xlink:href="#ShapeGroup" transform="translate(120,80) scale(0.4)"/>
    <use xlink:href="#ShapeGroup" transform="translate(20,60) scale(0.25)"/>
  </svg>
</body>
</html>
```



## □ 图案和渐变

- 图案<**pattern**>: 定义坐标、视图显示内容和视图大小。然后添加到模式形状中。

表: **pattern**元素的属性

属性	描述
id	用于引用这个模式的唯一ID。必需的
patternUnits	值为“useSpaceOnUse”或“objectBoundingBox”
patternContentUnits	值为“useSpaceOnUse”或“objectBoundingBox”
patternTransform	允许整个表达式进行转换
viewBox	各点“看到”这个SVG绘图区域



## □ 图案和渐变

- **渐变**：渐变是一种从一种颜色到另一种颜色的平滑过渡。另外，可以把多个颜色的过渡应用到同一个元素上。
- **SVG渐变**主要有两种类型：
  - 1) Linear
  - 2) Radial

## □ 图案和渐变

- **SVG线性渐变**：用<linearGradient>元素定义。
- <linearGradient>标签必须嵌套在<defs>的内部。
- 线性渐变可以定义为水平，垂直或角渐变：
  - 当y1和y2相等，而x1和x2不同时，可创建水平渐变；
  - 当x1和x2相等，而y1和y2不同时，可创建垂直渐变；
  - 当x1和x2不同，且y1和y2不同时，可创建角形渐变。

## □ 图案和渐变

- **<linearGradient>**标签

- id属性可为渐变定义一个唯一的名称。
- x1, x2, y1, y2属性定义渐变开始和结束位置。
- 渐变的颜色范围可由两种或多种颜色组成。每种颜色通过一个**<stop>**标签来规定。**offset**属性用来定义渐变的开始和结束位置。

## □ 图案和渐变

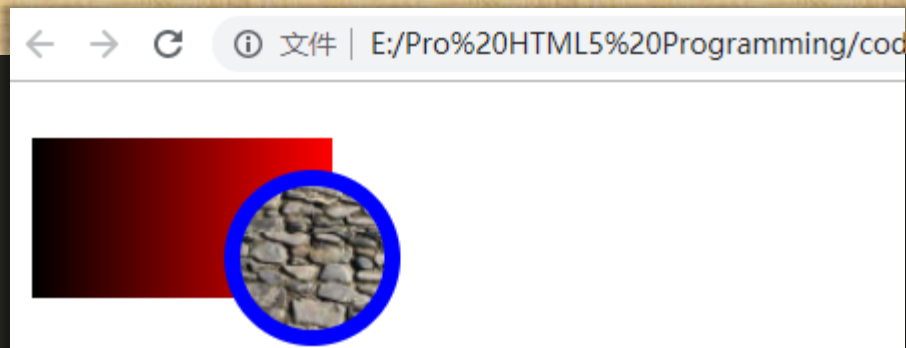
- **SVG 放射性渐变**：用<radialGradient>元素定义。
- <radialGradient>标签的使用方式与<linearGradient>标签类似，只有渐变位置不同：
  - **cx**，**cy**和 **r** 属性定义的最外层圆和 **fx** 和 **fy** 定义的最内层圆。

# □ 图案和渐变

- 为矩形和圆形添加纹理

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>图案和渐变</title>
</head>
<body>
  <svg width="200" height="200">
    <defs>
      <pattern id="GravelPattern" patternUnits="userSpaceOnUse" x="0" y="0" width="100"
        height="67" viewBox="0 0 100 67">
        <image x="0" y="0" width="100" height="67" xlink:href="gravel-1.jpg"></image>
      </pattern>
      <linearGradient id="redBlackGradient">
        <stop offset="0%" stop-color="#000"></stop>
        <stop offset="100%" stop-color="#f00"></stop>
      </linearGradient>
    </defs>

    <rect x="10" y="20" width="150" height="80" stroke="red"
      fill="url(#redBlackGradient)"/>
    <circle cx="150" cy="80" r="40" stroke="#00f" stroke-width="8"
      fill="url(#GravelPattern)" />
  </svg>
</body>
</html>
```





# □ SVG路径

- **<path>**元素用于定义一个路径。

d属性代表数据。

d属性的值是一系列绘制路径的命令；每条命令都可能带有坐标参数。

```
<path d="M150 0 L75 200 L225 200 Z" />
```



# □ SVG路径

➤ 下面的命令可用于路径数据:

- M = moveto
- L = lineto
- H = horizontal lineto
- V = vertical lineto
- C = curveto
- S = smooth curveto
- Q = quadratic Bézier curve
- T = smooth quadratic Bézier curveto
- A = elliptical Arc
- Z = closepath

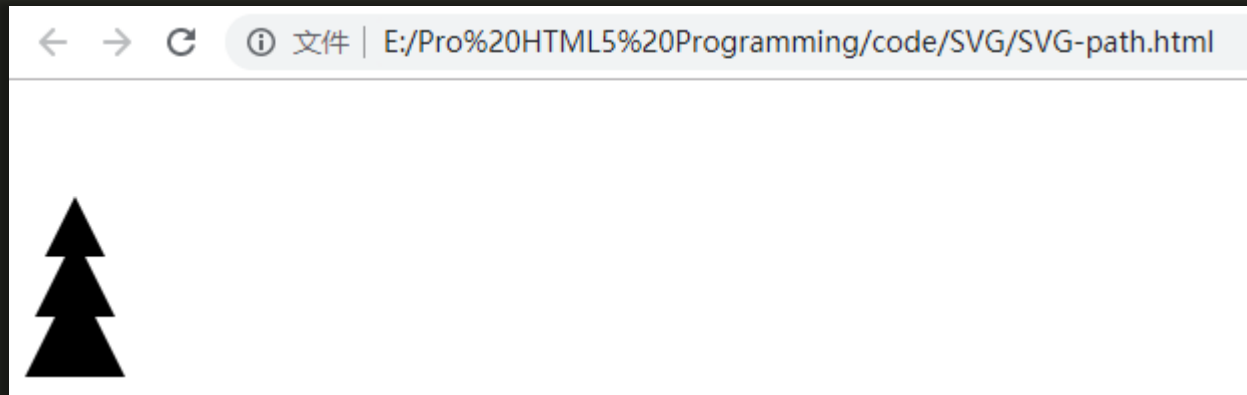
**注意：** 左边所有命令均允许小写字母。大写表示绝对定位，小写表示相对定位。

# □ SVG路径

- SVG路径定义的树冠

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>SVG路径</title>
</head>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <path d="M0 140
    L15 110
    L5 110
    L20 80
    L10 80
    L25 50
    L40 80
    L30 80
    L45 110
    L35 110
    L50 140
    Z" id="Canopy">
  </path>
</svg>

</body>
</html>
```



## □ 使用SVG文本

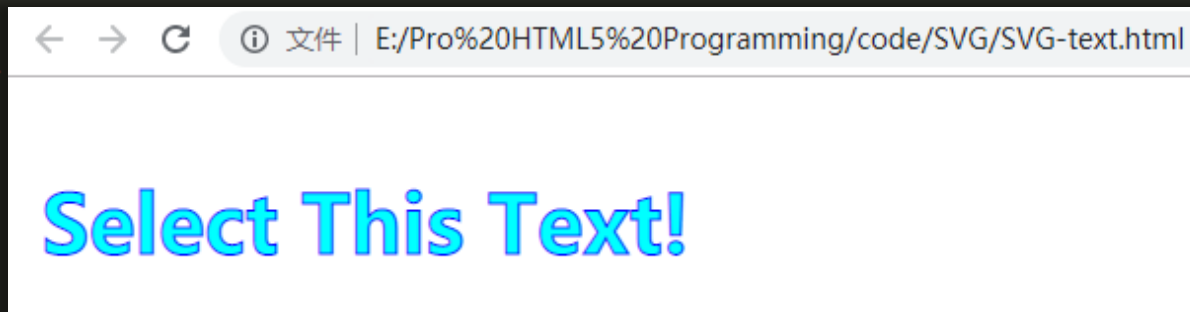
- **<text>**元素用于定义文本。
- SVG文本的属性类似于HTML里的CSS样式规则。

浏览器里，SVG中的文本是可选的。如果选用SVG文本，浏览器和搜索引擎能够支持用户搜索SVG text元素中的文本。这对可用性和可访问性有很大的益处。

# □ 使用SVG文本

- SVG文本

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>SVG文本</title>
</head>
<body>
  <svg width="600" height="200" xmlns="http://www.w3.org/2000/svg" version="1.1">
    <text
      x="10" y="80"
      font-family="Droid Sans"
      stroke="#00f"
      fill="#0ff"
      font-size="40px"
      font-weight="bold">
      Select This Text!
    </text>
  </svg>
</body>
</html>
```





# □ 组合场景

- SVG-HappyTrails.html完整代码

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Happy Trails in SVG</title>
  <style type="text/css">
    svg{
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <svg width="400" height="600">
    <defs>
      <!-- 小路 添加图案 -->
      <pattern id="GravelPattern" patternUnits="userSpaceOnUse" x="0" y="0" width="100"
height="67" viewBox="0 0 100 67">
        <image x=0 y=0 width=100 height=67 xlink:href="gravel.jpg"/>
      </pattern>
      <!-- 树干 渐变 -->
      <linearGradient id="TrunkGradient">
        <stop offset="0%" stop-color="#663300"></stop>
        <stop offset="40%" stop-color="#996600"></stop>
        <stop offset="100%" stop-color="#552200"></stop>
      </linearGradient>
```

## □ 组合场景

- SVG-HappyTrails.html完整代码（续）

```
<!-- 绘制树干 -->
<rect x="-5" y="-50" width="10" height="50" id="Trunk"/>
<!-- 绘制树冠 -->
<path d="M-25, -50
  L-10, -80
  L-20, -80
  L-5, -110
  L-15, -110
  L0, -140
  L15, -110
  L5, -110
  L20, -80
  L10, -80
  L25, -50
  Z" id="Canopy">
</path>
<!-- 树的阴影 -->
<linearGradient id="CanopyShadow" x=0 y=0 x2=0 y2=100%>
  <stop offset="0%" stop-color="#000" stop-opacity=".5"></stop>
  <stop offset="20%" stop-color="000" stop-opacity="0"></stop>
</linearGradient>
<!-- 复用内容 给每棵树都添加同样的样式-->
<g id="Tree">
  <use xlink:href="#Trunk" fill="url(#TrunkGradient)"/>
  <use xlink:href="#Trunk" fill="url(#CanopyShadow)"/>
  <use xlink:href="#Canopy" fill="none" stroke="#663300" stroke-linejoin="round"
    stroke-width="4px" />
```

## □ 组合场景

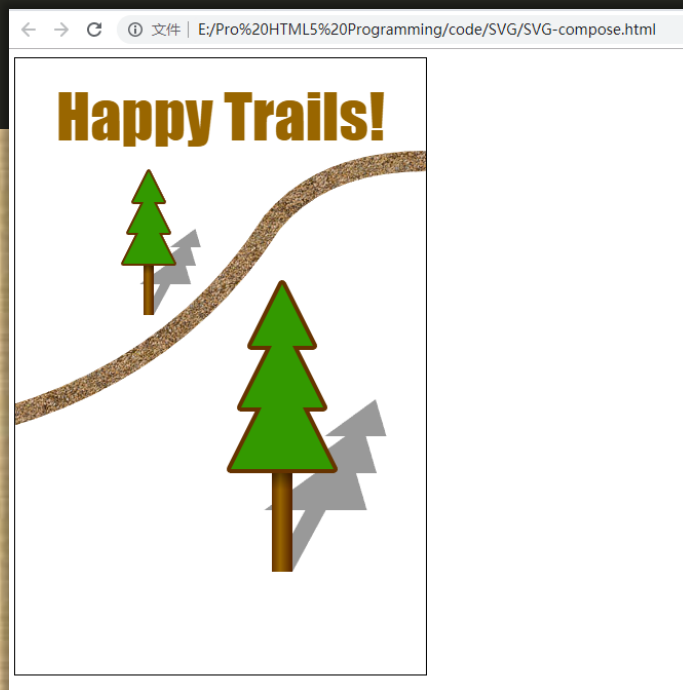
- SVG-HappyTrails.html完整代码（续）

```
<use xlink:href="#Canopy" fill="#339900" stroke="none"/>
</g>
<!-- 复用内容 给每棵树的树影添加同样的样式 -->
<g id="TreeShadow">
  <use xlink:href="#Trunk" fill="#000"/>
  <use xlink:href="#Canopy" fill="#000" stroke="none"/>
</g>
</defs>
<!-- 绘制小路 -->
<g transform="translate(-10, 350)"
  stroke-width="20"
  stroke = "url(#GravelPattern)" stroke-linejoin="round">
  <path d="M0,0 Q170,-50 260,-190 Q310,-250 410,-250" fill="none"></path>
</g>
<!-- 绘制SVG文字 -->
<text
  x="200" y="80"
  font-family="impact"
  fill="#996600"
  font-size="60px"
  text-anchor="middle">
  Happy Trails!
</text>
<!-- 绘制树和树影 -->
<use xlink:href="#TreeShadow" transform="translate(130, 250) scale(1,.6) skewX(-18)"
opacity="0.4"/>
<use xlink:href="#Tree" transform="translate(130, 250)"/>
```

## □ 组合场景

- SVG-HappyTrails.html完整代码（完）

```
<!-- 绘制树和树影 放大两倍-->  
<use xlink:href="#TreeShadow" transform="translate(260, 500) scale(2,1.2) skewX(-18)"  
opacity="0.4"/>  
<use xlink:href="#Tree" transform="translate(260, 500) scale(2)"/>  
</svg>  
</body>  
</html>
```



## 3.2 使用SVG创建交互式应用

- 添加树
- 添加updateTrees函数
- 添加removeTree函数
- 添加CSS样式



# □ 添加树

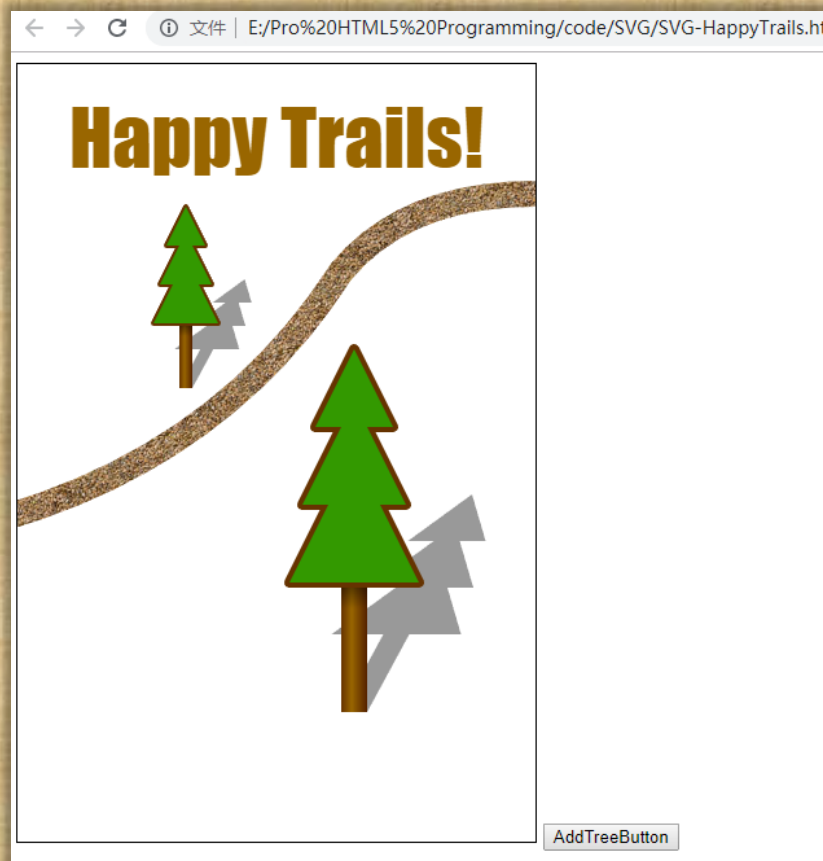
- 添加树的函数

```
<script type="text/javascript">
  document.getElementById("AddTreeButton").onclick = function(){
    var x = Math.floor(Math.random()*400);
    var y = Math.floor(Math.random()*600);
    var scale = Math.random() + 0.5;
    var translate = "translate(" + x + "," + y + ")";

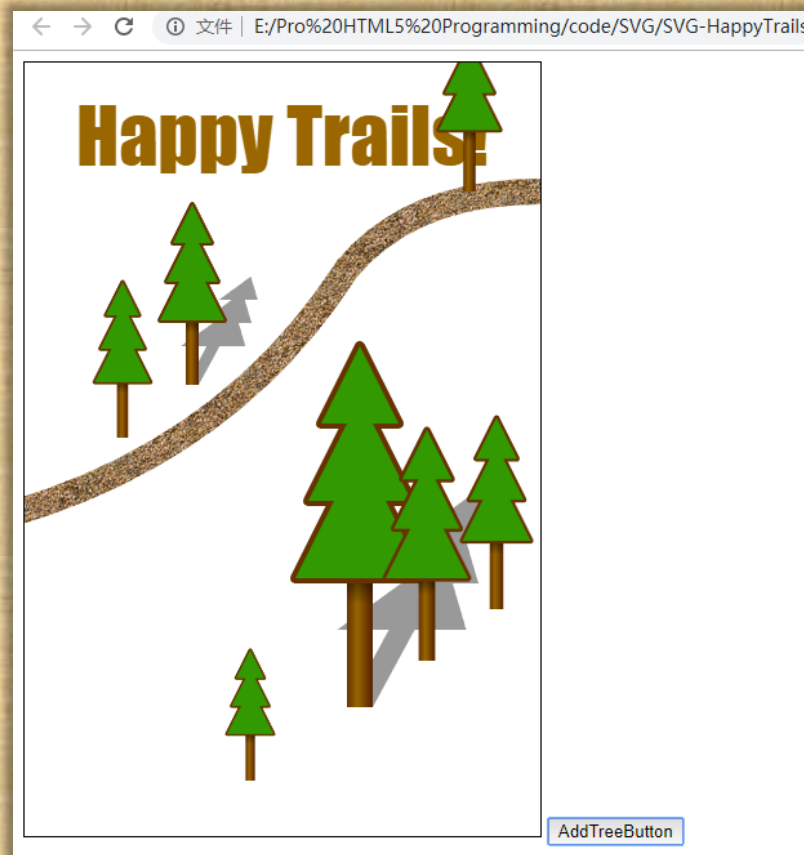
    var tree = document.createElementNS("http://www.w3.org/2000/svg", "use");
    tree.setAttributeNS("http://www.w3.org/1999/xlink", "xlink:href", "#Tree");
    tree.setAttribute("transform", translate + "scale(" + scale + ")");
    document.querySelector("svg").appendChild(tree);
    updateTrees();
  }
</script>
```

## □ 添加树

- 添加树效果



图：点击添加树按钮前效果



图：点击添加树按钮后效果

## □ 添加updateTrees函数

- 添加updateTrees函数

```
///更新树的数量
function updateTrees(){
    var list = document.querySelectorAll("use");
    var treeCount = 0;
    for(var i = 0; i < list.length; i++){
        if(list[i].getAttribute("xlink:href") == "#Tree"){
            treeCount++;
            list[i].onclick = removeTree;
        }
    }
    var counter = document.getElementById("TreeCounter");
    counter.textContent = treeCount + "trees in the forest";
}
```

## □ 添加updateTrees函数

- 添加removeTree函数

```
// 移除树
function removeTree(e){
    var elt = e.target;
    if(elt.correspondingUseElement){
        elt = elt.correspondingUseElement
    }
    elt.parentNode.removeChild(elt);
    updateTrees();
}
```

- 添加CSS样式

```
g[id=Tree]{
    opacity: 0.9;
    cursor: crosshair;
}
```

- 使用SVG创建交互式应用最终效果图





## 3.3 课后思考

- 讨论Canvas与SVG的区别。
- SVG path元素的哪个命令用于创建一条线？哪个命令用于关闭SVG中的路径？
- 什么是SVG模式？
- 请列出一些支持SVG的互联网浏览器。
- 请列举一些常用的SVG工具。

## 3.4 小结

- **HTML5 SVG**提供了强大功能，利用它可以创建二维图形交互应用。
- 介绍了**HTML5 SVG**和**Canvas**的区别，介绍了用于绘图的元素和属性，还学习了如何定义和复用内容、组合和变换元素以及如何绘制图形、路径和文本。
- 学习了在**SVG**文档中添加了**JavaScript**代码以实现交互式应用。