# 什么是SVG？

* SVG 指可伸缩矢量图形 (Scalable Vector Graphics)
* SVG 用于定义用于网络的基于矢量的图形
* 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

# SVG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

# 浏览器支持

* Internet Explorer 9+
* Firefox
* Opera
* Chrome
* Safari 支持内联SVG

# 文件SVG

|  |
| --- |
| <?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！

## 链接到SVG文件

可以用<a>标签，<img>标签，链接到一个SVG文件：链接到SVG文件

|  |
| --- |
| <a href="circle1.svg">View SVG file</a>  <img src="circle1.svg" alt="circle1"> |

# 内联SVG

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">  <polygon points="100,10 40,180 190,60 10,60 160,180"  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;"/>  </svg>    </body>  </html> |

# SVG 与 Canvas两者间的区别

* SVG 是一种使用 XML 描述 2D 图形的语言。
* Canvas 通过 JavaScript 来绘制 2D 图形。
* SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 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。
* 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。

# Canvas 与 SVG 的比较

下表列出了 canvas 与 SVG 之间的一些不同之处。

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

# SVG预定义的形状元素

## 矩形 <rect>

### 实例1

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0)"/>  </svg> |

**代码解析:**

* 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
* CSS 的 fill 属性定义矩形的填充颜色（rgb 值、颜色名或者十六进制值）
* CSS 的 stroke-width 属性定义矩形边框的宽度
* CSS 的 stroke 属性定义矩形边框的颜色

### 实例2

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <rect x="50" y="20" width="150" height="150" style="fill:blue;stroke:pink;stroke-width:5;fill-opacity:0.1;stroke-opacity:0.9"/>  </svg> |

**代码解析：**

* x 属性定义矩形的左侧位置（例如，x="0" 定义矩形到浏览器窗口左侧的距离是 0px）
* y 属性定义矩形的顶端位置（例如，y="0" 定义矩形到浏览器窗口顶端的距离是 0px）
* CSS 的 fill-opacity 属性定义填充颜色透明度（合法的范围是：0 - 1）
* CSS 的 stroke-opacity 属性定义笔触颜色的透明度（合法的范围是：0 - 1）

### 实例 3

定义整个元素的不透明度：

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <rect x="50" y="20" width="150" height="150" style="fill:blue;stroke:pink;stroke-width:5;opacity:0.5"/>  </svg> |

**代码解析：**

* CSS opacity 属性用于定义了元素的透明值 (范围: 0 到 1)。

### 实例 4

创建一个圆角矩形：

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <rect x="50" y="20" rx="20" ry="20" width="150" height="150" style="fill:red;stroke:black;stroke-width:5;opacity:0.5"/>  </svg> |

* rx 和 ry 属性可使矩形产生圆角。

## 圆形 <circle>

<circle> 标签可用来创建一个圆：

|  |
| --- |
| <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> |

**代码解析：**

* 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

## 椭圆 <ellipse>

### 实例 1

<ellipse> 元素是用来创建一个椭圆：

椭圆与圆很相似。不同之处在于椭圆有不同的x和y半径，而圆的x和y半径是相同的：

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <ellipse cx="300" cy="80" rx="100" ry="50" style="fill:yellow;stroke:purple;stroke-width:2"/>  </svg> |

**代码解析：**

* 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

### 实例 2

下面的例子创建了三个累叠而上的椭圆：

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <ellipse cx="240" cy="100" rx="220" ry="30" style="fill:purple"/>  <ellipse cx="220" cy="70" rx="190" ry="20" style="fill:lime"/>  <ellipse cx="210" cy="45" rx="170" ry="15" style="fill:yellow"/>  </svg> |

### 实例 3

下面的例子组合了两个椭圆（一个黄的和一个白的）：

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <ellipse cx="240" cy="50" rx="220" ry="30" style="fill:yellow"/>  <ellipse cx="220" cy="50" rx="190" ry="20" style="fill:white"/>  </svg> |

## 线 <line>

<line> 元素是用来创建一个直线：

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2"/>  </svg> |

* 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

## 多边形 <polygon>

### 实例 1

<polygon> 标签用来创建含有不少于三个边的图形。

多边形是由直线组成，其形状是"封闭"的（所有的线条 连接起来）。

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <polygon points="200,10 250,190 160,210" style="fill:lime;stroke:purple;stroke-width:1"/>  </svg> |

**代码解析：**

* points 属性定义多边形每个角的 x 和 y 坐标

### 实例 2

下面的示例创建一个四边的多边形：

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <polygon points="220,10 300,210 170,250 123,234" style="fill:lime;stroke:purple;stroke-width:1"/>  </svg> |

### 实例 3

使用 <polygon> 元素创建一个星型:

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <polygon points="100,10 40,180 190,60 10,60 160,180" style="fill:lime;stroke:purple;stroke-width:5;fill-rule:nonzero;" />  </svg> |

### 实例 4

改变 fill-rule 属性为 "evenodd":

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <polygon points="100,10 40,180 190,60 10,60 160,180" style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />  </svg> |

## 折线 <polyline>

### 实例 1

<polyline> 元素是用于创建任何只有直线的形状：

|  |
| --- |
| <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> |

### 实例 2

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160" style="fill:white;stroke:red; stroke-width:4" />  </svg> |