SVG 图像

目录
概述
语法
<svg>标签
<circle>标签
<line>标签
<polyline>标签
<rect>标签
<ellipse>标签
<polygon>标签
<path>标签
<text>标签
<use>标签
<g>标签
<defs>标签
<pattern>标签
<image>标签
<animate>标签
<animateTransform>标签
JavaScript 操作
DOM 操作
获取 SVG DOM
读取 SVG 源码
SVG 图像转为 Canvas 图像
实例：折线图
参考链接

概述

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 文件可以直接插入网页，成为 DOM 的一部分，然后用 JavaScript 和 CSS 进行操作。

```
<!DOCTYPE html>
<html>
<head></head>
<body>
<svg
  id="mysvg"
  xmlns="http://www.w3.org/2000/svg"
  viewBox="0 0 800 600"
  preserveAspectRatio="xMidYMid meet"
>
  <circle id="mycircle" cx="400" cy="300" r="50" />
</svg>
</body>
</html>
```

上面是 SVG 代码直接插入网页的例子。

SVG 代码也可以写在一个独立文件中，然后用 `<img>`、`<object>`、`<embed>`、`<iframe>` 等标签插入网页。

```
<img src="circle.svg">
<object id="object" data="circle.svg" type="image/svg+xml"></object>
<embed id="embed" src="icon.svg" type="image/svg+xml">
<iframe id="iframe" src="icon.svg"></iframe>
```

CSS 也可以使用 SVG 文件。

```
.logo {
  background: url(icon.svg);
}
```

SVG 文件还可以转为 BASE64 编码，然后作为 Data URI 写入网页。

```
<img src="[data]">
```

语法

`<svg>`标签

SVG 代码都放在顶层标签 `<svg>` 之中。下面是一个例子。

```
<svg width="100%" height="100%">
  <circle id="mycircle" cx="50" cy="50" r="50" />
</svg>
```

`<svg>` 的 `width` 属性和 `height` 属性，指定了 SVG 图像在 HTML 元素中所占据的宽度和高度。除了相对单位，也可以采用绝对单位（单位：像素）。如果不指定这两个属性，SVG 图像的大小默认为 300像素（宽） x 150像素（高）。

如果只想展示 SVG 图像的一部分，就要指定 `viewBox` 属性。

```
<svg width="100" height="100" viewBox="50 50 50 50">
  <circle id="mycircle" cx="50" cy="50" r="50" />
</svg>
```

`<viewBox>` 属性的值有四个数字，分别是左上角的横坐标和纵坐标、视口的宽度和高度。上面代码中，SVG 图像是 100像素宽 x 100像素高，`viewBox` 属性指定视口从 `(50, 50)` 这个点开始。所以，实际看到的是右下角的四分之一圆。

注意，视口必须适配所在的空间。上面代码中，视口的大小是 50 x 50，由于 SVG 图像的大小是 100 x 100，所以视口会放大去适配 SVG 图像的大小，即放大了四倍。

如果不指定 `width` 属性和 `height` 属性，只指定 `viewBox` 属性，则相当于只给定 SVG 图像的长宽比。这时，SVG 图像的大小默认是所在的 HTML 元素的大小。

`<circle>`标签

`<circle>` 标签代表圆形。

```
<svg width="300" height="180">
  <circle cx="30"  cy="50" r="25" />
  <circle cx="90"  cy="50" r="25" class="red" />
```

```
  <circle cx="150" cy="50" r="25" class="fancy" />
</svg>
```

上面的代码定义了三个圆。`<circle>`标签的`cx`、`cy`、`r`属性分别为横坐标、纵坐标和半径，单位为像素。坐标都是相对于`<svg>`画布的左上角原点。

`class`属性用来指定对应的 CSS 类。

```
.red {
  fill: red;
}

.fancy {
  fill: none;
  stroke: black;
  stroke-width: 3pt;
}
```

SVG 的 CSS 属性与网页元素有所不同。

> - fill: 填充色
> - stroke: 描边色
> - stroke-width: 边框宽度

`<line>`标签

`<line>`标签用来绘制直线。

```
<svg width="300" height="180">
  <line x1="0" y1="0" x2="200" y2="0" style="stroke:rgb(0,0,0);stroke-width:5" />
</svg>
```

上面代码中，`<line>`标签的`x1`属性和`y1`属性，表示线段起点的横坐标和纵坐标；`x2`属性和`y2`属性，表示线段终点的横坐标和纵坐标；`style`属性表示线段的样式。

`<polyline>`标签

`<polyline>`标签用于绘制一根折线。

```
<svg width="300" height="180">
  <polyline points="3,3 30,28 3,53" fill="none" stroke="black" />
</svg>
```

<polyline> points

<rect>

<rect>

```
<svg width="300" height="180">
  <rect x="0" y="0" height="100" width="200" style="stroke: #70d5dd; fill: #dd524b" />
</svg>
```

<rect> x y width height

<ellipse>

<ellipse>

```
<svg width="300" height="180">
  <ellipse cx="60" cy="60" ry="40" rx="20" stroke="black" stroke-width="5" fill="silver"/>
</svg>
```

<ellipse> cx cy rx ry

<polygon>

<polygon>

```
<svg width="300" height="180">
  <polygon fill="green" stroke="orange" stroke-width="1" points="0,0 100,0 100,100 0,100
0,0"/>
</svg>
```

<polygon> points

`<path>` 标签

`<path>` 标签用于制路径。

```
<svg width="300" height="180">
<path d="
  M 18,3
  L 46,3
  L 46,40
  L 61,40
  L 32,68
  L 3,40
  L 18,40
  Z
"></path>
</svg>
```

`<path>` 的 `d` 属性表示绘制顺序，它的值是一个长字符串，每个字母表示一个绘制动作，后面跟着坐标。

> - M: 移动到（moveto）
> - L: 画直线到（lineto）
> - Z: 闭合路径

`<text>` 标签

`<text>` 标签用于绘制文本。

```
<svg width="300" height="180">
  <text x="50" y="25">Hello World</text>
</svg>
```

`<text>` 的 `x` 属性和 `y` 属性，表示文本区块基线（baseline）起点的横坐标和纵坐标。文字的样式可以用 `class` 或 `style` 属性指定。

`<use>` 标签

`<use>` 标签用于复制一个形状。

```
<svg viewBox="0 0 30 10" xmlns="http://www.w3.org/2000/svg">
  <circle id="myCircle" cx="5" cy="5" r="4"/>

  <use href="#myCircle" x="10" y="0" fill="blue" />
  <use href="#myCircle" x="20" y="0" fill="white" stroke="blue" />
</svg>
```

`<use>`的 `href`属性指定所要复制的节点，`x`属性和`y`属性是`<use>`左上角的坐标。另外，还可以指定`width`和`height`坐标。

`<g>`标签

`<g>`标签用于将多个形状组成一个组（group），方便复用。

```
<svg width="300" height="100">
  <g id="myCircle">
    <text x="25" y="20">圆形</text>
    <circle cx="50" cy="50" r="20"/>
  </g>

  <use href="#myCircle" x="100" y="0" fill="blue" />
  <use href="#myCircle" x="200" y="0" fill="white" stroke="blue" />
</svg>
```

`<defs>`标签

`<defs>`标签用于自定义形状，它内部的代码不会显示，仅供引用。

```
<svg width="300" height="100">
  <defs>
    <g id="myCircle">
      <text x="25" y="20">圆形</text>
      <circle cx="50" cy="50" r="20"/>
    </g>
  </defs>

  <use href="#myCircle" x="0" y="0" />
  <use href="#myCircle" x="100" y="0" fill="blue" />
```

```
  <use href="#myCircle" x="200" y="0" fill="white" stroke="blue" />
</svg>
```

`<pattern>`标签

`<pattern>`标签用于自定义一个形状，该形状可以被引用来平铺一个区域。

```
<svg width="500" height="500">
  <defs>
    <pattern id="dots" x="0" y="0" width="100" height="100" patternUnits="userSpaceOnUse">
      <circle fill="#bee9e8" cx="50" cy="50" r="35" />
    </pattern>
  </defs>
  <rect x="0" y="0" width="100%" height="100%" fill="url(#dots)" />
</svg>
```

上面代码中，`<pattern>`标签将一个圆形定义为`dots`模式。`patternUnits="userSpaceOnUse"`表示`<pattern>`的宽度和长度是实际的像素值。然后，指定这个模式去填充下面的矩形。

`<image>`标签

`<image>`标签用于插入图片文件。

```
<svg viewBox="0 0 100 100" width="100" height="100">
  <image xlink:href="path/to/image.jpg"
    width="50%" height="50%"/>
</svg>
```

上面代码中，`<image>`的`xlink:href`属性表示图像的来源。

`<animate>`标签

`<animate>`标签用于产生动画效果。

```
<svg width="500px" height="500px">
  <rect x="0" y="0" width="100" height="100" fill="#feac5e">
    <animate attributeName="x" from="0" to="500" dur="2s" repeatCount="indefinite" />
  </rect>
</svg>
```

上面代码中，矩形会不断移动，产生动画效果。

`<animate>`的属性含义如下。

> - attributeName: 发生动画效果的属性名。
> - from: 单次动画的初始值。
> - to: 单次动画的结束值。
> - dur: 单次动画的持续时间。
> - repeatCount: 动画的循环模式。

可以在多个属性上面定义动画。

```
<animate attributeName="x" from="0" to="500" dur="2s" repeatCount="indefinite" />
<animate attributeName="width" to="500" dur="2s" repeatCount="indefinite" />
```

`<animateTransform>`标签

`<animate>`标签对 CSS 的 `transform`属性不起作用，如果需要变形，就要使用 `<animateTransform>`标签。

```
<svg width="500px" height="500px">
  <rect x="250" y="250" width="50" height="50" fill="#4bc0c8">
    <animateTransform attributeName="transform" type="rotate" begin="0s" dur="10s" from="0 200
200" to="360 400 400" repeatCount="indefinite" />
  </rect>
</svg>
```

上面代码中，`<animateTransform>`的效果为旋转（`rotate`），这时`from`和`to`属性值有三个数字，第一个数字是角度值，第二个值和第三个值是旋转中心的坐标。`from="0 200 200"`表示开始时，角度为 0，围绕`(200, 200)`开始旋转；`to="360 400 400"`表示结束时，角度为 360，围绕`(400, 400)`旋转。

JavaScript 操作

DOM 操作

如果 SVG 代码直接写在 HTML 网页之中，它就成为网页 DOM 的一部分，可以直接用 DOM 操作。

```
<svg
  id="mysvg"
  xmlns="http://www.w3.org/2000/svg"
  viewBox="0 0 800 600"
  preserveAspectRatio="xMidYMid meet"
>
  <circle id="mycircle" cx="400" cy="300" r="50" />
<svg>
```

□□□□□□□□□□□□□□□□ CSS □□□□□

```
circle {
  stroke-width: 5;
  stroke: #f00;
  fill: #ff0;
}

circle:hover {
  stroke: #090;
  fill: #fff;
}
```

□□□□□□ JavaScript □□□□ SVG□

```
var mycircle = document.getElementById('mycircle');

mycircle.addEventListener('click', function(e) {
  console.log('circle clicked - enlarging');
  mycircle.setAttribute('r', 60);
}, false);
```

□□□□□□□□□□□□□□□□□□ `circle` □□□ `r` □□□

□□ SVG DOM

□□ `<object>`□ `<iframe>`□ `<embed>`□□□□ SVG □□□□□□□ SVG DOM□

```
var svgObject = document.getElementById('object').contentDocument;
var svgIframe = document.getElementById('iframe').contentDocument;
var svgEmbed = document.getElementById('embed').getSVGDocument();
```

□□□□□□□ <img>□□□ SVG □□□□□□□ SVG DOM□

□□ SVG □□

□□ SVG □□□□□ XML □□□□□□□□□ XML □□□□□□□ SVG □□

```
<div id="svg-container">
  <svg
    xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xml:space="preserve" width="500" height="440"
  >
    <!-- svg code -->
  </svg>
</div>
```

□□ XMLSerializer□□ serializeToString()□□□ SVG □□□□□

```
var svgString = new XMLSerializer()
  .serializeToString(document.querySelector('svg'));
```

SVG □□□ Canvas □□

□□□□□□□ Image□□ SVG □□□□□ Image□□ src□□

```
var img = new Image();
var svg = new Blob([svgString], {type: "image/svg+xml;charset=utf-8"});

var DOMURL = self.URL || self.webkitURL || self;
var url = DOMURL.createObjectURL(svg);

img.src = url;
```

□□□□□□□□□□□□□□□□□ <canvas>□□

```
img.onload = function () {
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext('2d');
  ctx.drawImage(img, 0, 0);
```

```
};
```

实例：折线图

下面将一张数据表格画成折线图。

```
Date |Amount
-----|------
2014-01-01 | $10
2014-02-01 | $20
2014-03-01 | $40
2014-04-01 | $80
```

上面的图形，可以画成一个坐标系，`Date`作为横轴，`Amount`作为纵轴，四行数据画成一个数据点。

```
<svg width="350" height="160">
  <g class="layer" transform="translate(60,10)">
    <circle r="2" cx="0" cy="105" />
    <circle r="2" cx="90" cy="90" />
    <circle r="2" cx="180" cy="60" />
    <circle r="2" cx="270" cy="0" />

    <polyline points="0,105 90,90 180,60 270,0" fill="none" stroke="red" />

    <g class="y axis">
      <line x1="0" y1="0" x2="0" y2="120" style="stroke:black;stroke-width:1" />
      <text x="-40" y="115" dy="5">$10</text>
      <text x="-40" y="5" dy="5">$80</text>
    </g>
    <g class="x axis" transform="translate(0, 120)">
      <line x1="0" y1="0" x2="270" y2="0" style="stroke:black;stroke-width:1" />
      <text x="-10" y="20">Jan.</text>
      <text x="255" y="20">Apr.</text>
    </g>
  </g>
</svg>
```

□□□

- Jon McPartland, An introduction to SVG animation
- Alexander Goedde, SVG - Super Vector Graphics
- Joseph Wegner, Learning SVG
- biovisualize, Direct svg to canvas to png conversion
- Tyler Sticka, Cropping Image Thumbnails with SVG
- Adi Purdila, How to Create a Loader Icon With SVG Animations

---

□□ #1
□ □□ □□ 2018-08-05 10:37:51 UTC
□ □□ □□ 2023-10-25 23:41:19 UTC