# HTML5 第四天笔记

## 基础内容

### 1. 什么是 SVG

SVG 全称为 Scalable Vector Graphics，译为可缩放矢量图形,简称矢量图。是一种用来描述二维矢量图形的 XML 标记语言。

### 2. SVG与Flash的区别

|  | **SVG** | **Flash** |
| --- | --- | --- |
| 相同点 | 用于二维矢量图形 | 用于二维矢量图形 |
| 不同点 | 是一个开放的W3C标准，基于XML | 封闭的基于二进制格式的 |

### 3. SVG与Canvas的区别

| **SVG** | **Canvas** |
| --- | --- |
| 不依赖于分辨率 | 依赖于分辨率 |
| 使用DOM及事件处理器（DOM专门为SVG开放接口） | 不能使用DOM及事件处理器 |
| 不能实现游戏开发 | 可以实现游戏开发 |
| 实现大型渲染区域的应用(例如百度地图) | 是以图片(png |

### 4. SVG的优势

* 不需要专门的编辑器,文本编辑器都可以
* 可被搜索、索引、脚本化及压缩
* 图像不失真(和分辨率无关)

**值得注意的是：**

* SVG技术并不是专属于HTML5的
* SVG技术本身是一套独立的用于描述二维图形
* HTML5版本之前,以图片形式进行引入
* HTML5版本之后,允许在HTML页面直接使用SVG技术

**可参考的资源：**

* SVG 标准：<http://www.w3.org/Graphics/SVG/>
* SVG 教程：<https://developer.mozilla.org/zh-CN/docs/Web/SVG/Tutorial>

### 5. 如何使用 SVG

#### 1）SVG 文件（了解）

SVG 文件的扩展名为 “.svg”，使用的是 XML 技术的语法内容，并且可以在浏览器中直接运行。

<?xml version="1.0" encoding="utf-8" ?>

<!-- SVG的语法标准(必要的) -->

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<!-- xmlns="http://www.w3.org/2000/svg" SVG的命名空间 -->

<svg version="1.1" xmlns="http://www.w3.org/2000/svg">

<rect x="100" y="100" width="300" height="100" fill="blue" stroke="black" stroke-width="4"/>

</svg>

也可以在 HTML 页面引入 SVG 文件。

<img src="01\_SvgFile.svg" width="800" height="500">

#### 2）HTML 直接定义 SVG

HTML5 的新特性允许在 HTML 页面中直接定义 SVG 元素（需要学习有关SVG的一些HTML新元素）。

**<svg>元素：**

|  | **<svg>元素** |
| --- | --- |
| 作用 | 类似于canvas元素 |
| 特点 | 默认的宽度和高度分别为300px和150px，不具有任何的样式(默认情况下不显示) |
| 目的 | 表示当前使用SVG语法内容 |

**<svg>元素的属性：**

| **属性名称** | **描述** |
| --- | --- |
| width | 设置 <svg> 元素的宽度 |
| height | 设置 <svg> 元素的高度 |
| style | 设置 <svg> 元素的样式 |

<svg width="500" height="300" style="background:pink;">

<rect x="100" y="100" width="300" height="100" fill="blue" stroke="black" stroke-width="4"/>

</svg>

## 图像元素

### 1. 样式

| **属性名称** | **描述** |
| --- | --- |
| fill | 设置填充样式 |
| stroke | 设置描边样式 |
| stroke-width | 设置描边宽度 |

### 2. 矩形

<rect x="" y="" rx="" ry="" width="" height="" />

| **属性名称** | **描述** |
| --- | --- |
| x和y | 表示绘制矩形的左上角坐标值 |
| width | 表示绘制矩形的宽度 |
| height | 表示绘制矩形的高度 |
| rx和ry | 表示绘制矩形的四个角的水平圆角和垂直圆角（如果rx和ry的值分别为width/2和height/2,绘制圆形） |

**值得注意的是：**

* 默认绘制出来的效果：实心矩形效果。
* 可以实现"空心"的效果：不能实现空心效果。

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制矩形 -->

<rect x="10" y="10" width="100" height="100" />

<rect x="10" y="120" width="100" height="100" fill="white" stroke="black" />

<rect x="10" y="230" width="100" height="100" rx="10" ry="10" />

<rect x="10" y="340" width="100" height="100" rx="50" ry="50" style="fill:white;stroke:black;" />

</svg>

### 3. 圆形

<cirlcle cx="" cy="" r="" />

| **属性名称** | **描述** |
| --- | --- |
| cx和cy | 表示绘制圆形的圆心坐标值 |
| r | 表示绘制圆形的半径 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制圆形 -->

<circle cx="170" cy="60" r="50" />

<circle cx="170" cy="170" r="50" fill="white" stroke="black" />

</svg>

### 4. 椭圆

<ellipse cx="" cy="" rx="" ry="" />

| **属性名称** | **描述** |
| --- | --- |
| cx和cy | 表示绘制椭圆的圆心坐标值 |
| rx | 表示绘制椭圆的水平方向半径 |
| ry | 表示绘制椭圆的垂直方向半径 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制椭圆 -->

<ellipse cx="280" cy="70" rx="50" ry="60" />

<ellipse cx="280" cy="190" rx="50" ry="50" />

<ellipse cx="280" cy="300" rx="50" ry="30" />

</svg>

### 5. 线条

<line x1="" y1="" x2="" y2="" stroke="" />

| **属性名称** | **描述** |
| --- | --- |
| x1和y1 | 表示绘制直线的起点坐标值 |
| x2和y2 | 表示绘制直线的终点坐标值 |
| stroke | 设置绘制直线的样式(颜色) |
| stroke-width | 设置绘制直线的宽度 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制直线 -->

<line x1="350" y1="10" x2="500" y2="200" stroke="black" stroke-width="10" />

</svg>

### 6. 折线

<polyline points="" />

| **属性名称** | **描述** |
| --- | --- |
| points | 表示绘制折线的起点、折点及终点坐标值（格式：x1,y1 x2,y2 x3,y3 xn,yn） |
| stroke | 设置折线的颜色 |
| stroke-width | 设置折线的线宽 |
| fill | 设置与<svg>元素的背景色相同 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制折线 -->

<polyline points="520,20 520,200" stroke="black" stroke-width="10" />

<polyline points="550,20 550,200 700,200 700,20 540,20" stroke="black" stroke-width="20" fill="pink" />

</svg>

### 7. 多边形

<polygon points="" />

| **属性名称** | **描述** |
| --- | --- |
| points | 表示绘制多边形的所有点坐标值 |

<svg width="1000" height="600" style="background:pink;">

<!-- 绘制多边形 -->

<polygon points="550,300 550,500 800,500" stroke="black" fill="pink" stroke-width="10" />

</svg>

**值得注意的是：**

* 一个HTML页面允许包含多个<svg>元素
* 一个<svg>元素允许包含多个图形元素
* SVG的图形元素基本都是起始元素
* 定义图形元素时,只定义开始元素,没有结束元素
  + 浏览器在运行页面时,并不报错
  + 浏览器解析这段元素代码时,自动补全结束元素
  + 自动补全的结束元素是不正确的

### 8. 渐变

#### 1）线性渐变

<linearGradient id="" x1="" y1="" x2="" y2=""></linearGradient>

| **属性名称** | **描述** |
| --- | --- |
| id | 标识，便于其他元素进行引用 |
| x1和y1 | 表示基准线的起点坐标值（值范围为 0-1, 是百分值 0%-100%） |
| x2和y2 | 表示基准线的终点坐标值（值范围为 0-1, 是百分值 0%-100%） |

<stop offset="" stop-color=""/>

| **属性名称** | **描述** |
| --- | --- |
| offset | 表示设置渐变颜色的位置 |
| stop-color | 表示设置的渐变颜色 |
| stop-opacity | 表示设置渐变颜色的透明度 |

<svg width="800" height="500">

<defs>

<linearGradient id="grd" x1="0" y1="0" x2="1" y2="1">

<stop offset="0" stop-color="red" stop-opacity="0.5" />

<stop offset="0.5" stop-color="green" stop-opacity="0.5" />

<stop offset="1" stop-color="blue" stop-opacity="0.5" />

</linearGradient>

</defs>

<rect x="0" y="0" width="200" height="200" fill="url(#grd)" />

<circle cx="400" cy="400" r="100" fill="url(#grd)" />

</svg>

#### 2）射线渐变

<radialGradient id="" cx="" cy="" r="" fx="" fy=""></radialGradient>

| **属性名称** | **描述** |
| --- | --- |
| id | 标识，便于其他元素进行引用 |
| cx和cy | 表示基准线中的中心点坐标值（值范围为 0-1, 是百分值 0%-100%） |
| fx和fy | 表示基准线中的焦点坐标值（值范围为 0-1, 是百分值 0%-100%） |
| r | 设置其边缘位置（值范围为 0-1, 是百分值 0%-100%） |

<stop offset="" stop-color=""/>

| **属性名称** | **描述** |
| --- | --- |
| offset | 表示设置渐变颜色的位置 |
| stop-color | 表示设置的渐变颜色 |
| stop-opacity | 表示设置渐变颜色的透明度 |

<svg width="800" height="500">

<defs>

<radialGradient id="grd1">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd2" cx="0" cy="0">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd3" r="1">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd4" fx="1" fy="1">

<stop offset="0" stop-color="red" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd5">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd6" cx="0" cy="0">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd7" r="1">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

<radialGradient id="grd8" fx="1" fy="1">

<stop offset="0" stop-color="red" />

<stop offset="0.5" stop-color="yellow" />

<stop offset="1" stop-color="blue" />

</radialGradient>

</defs>

<rect x="0" y="0" width="200" height="200" fill="url(#grd1)" />

<rect x="210" y="0" width="200" height="200" fill="url(#grd2)" />

<rect x="420" y="0" width="200" height="200" fill="url(#grd3)" />

<rect x="630" y="0" width="200" height="200" fill="url(#grd4)" />

<rect x="0" y="210" width="200" height="200" fill="url(#grd5)" />

<rect x="210" y="210" width="200" height="200" fill="url(#grd6)" />

<rect x="420" y="210" width="200" height="200" fill="url(#grd7)" />

<rect x="630" y="210" width="200" height="200" fill="url(#grd8)" />

</svg>

**值得注意的是：**

* 渐变元素需要定义id属性,便于其他元素进行引用
* 渐变元素定义在元素内
  + 元素内定义：表示该元素允许重复使用

### 9. 滤镜

<filter id="myfilter"></filter>

**值得注意的是：**该元素只是滤镜的容器。

#### 高斯模糊

<feGaussianBlur in="" stdDeviation="" />

| **属性名称** | **描述** |
| --- | --- |
| in | 设置高斯模糊的样式 |
| stdDeviation | 设置模糊的程度 |

<svg width="800" height="500">

<defs>

<filter id="myfilter">

<feGaussianBlur in="SourceGraphic" stdDeviation="5" />

</filter>

</defs>

<rect x="100" y="100" width="100" height="100" filter="url(#myfilter)" />

</svg>

## Two.js库

### 1. 基础内容

Two.js支持不同的上下文环境：

* SVG（默认）
* Canvas
* WebGL

官网地址：<http://jonobr1.github.io/two.js/>

#### 如何使用 Two.js

##### 1）HTML 页面

* 引入Two.js库文件
* 定义容器元素 <div></div>

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>如何使用Two.js库</title>

<!-- 1. 引入Two.js库文件 -->

<script src="two.js"></script>

</head>

<body>

<!--

2. 定义用于显示矢量图的容器元素

\* <div></div>元素 - 建议使用

-->

<div id="d1"></div>

</body>

</html>

##### 2）JavaScript逻辑

* 获取HTML页面的容器元素
* 通过 Two.js 库提供的 Two() 构造函数创建 Two 对象

var params = {// 创建svg时初始化的数据

width : 宽度,

height : 高度

}

var two = new Two(params);

* 将创建的 Two 对象添加到 HTML 页面容器元素内

two.appendTo(elem);

* 使用 Two.js 库提供的 API 方法绘制图形
* 调用 update() 方法进行绘制

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>如何使用Two.js库</title>

<!-- 1. 引入Two.js库文件 -->

<script src="two.js"></script>

</head>

<body>

<!--

2. 定义用于显示矢量图的容器元素

\* <div></div>元素 - 建议使用

-->

<div id="d1"></div>

<script>

// 3. 获取HTML页面的容器元素

var elem = document.getElementById("d1");

/\*

4. 通过Two(params)构造函数来创建Two对象

\* params - 创建Two对象的初始化数据

\* Object类型,{ key : value }格式

\*/

var params = {

type : Two.Types.canvas,

width : 285, // 默认为640px

height : 200 // 默认为480px

};

var two = new Two(params);

/\*

5. 将创建的Two对象添加到页面容器元素中

\* Two.js库提供了appendTo()方法向容器添加Two对象

\* 注意 - appendTo()方法并不是jQuery的

\*/

two.appendTo(elem);

// 6. 通过调用Two对象提供的API方法进行绘制

var circle = two.makeCircle(72, 100, 50);

var rect = two.makeRectangle(213, 100, 100, 100);

// 7. 调用update()方法进行绘制

two.update();

</script>

</body>

</html>

### 2. 绘制图像

#### 1）绘制图像的方法

| **图像形状** | **方法** |
| --- | --- |
| 圆形 | makeCircle(x, y, radius) |
| 直线 | makeLine(x1, y1, x2, y2) |
| 矩形 | makeRectangle(x, y, width, height) |
| 圆角矩形 | makeRoundedRectangle(x, y, width, height, radius) |
| 椭圆 | makeEllipse(x, y, width, height) |
| 多边形 | makePolygon(ox, oy, r, sides) |
| 路径 | makePath(x1, y1, x2, y2, xN, yN, open) |
| 星形 | makeStar(ox, oy, or, ir, sides) |

#### 2）绘制图像的属性

| **属性名称** | **描述** |
| --- | --- |
| fill | 设置填充样式 |
| stroke | 设置描边样式 |
| linewidth | 设置线条宽度 |
| opacity | 设置透明度 |
| cap | 设置线条端点形状，默认为round |
| join | 设置线条交点形状，默认为round |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>如何使用Two.js库</title>

<script src="two.js"></script>

</head>

<body>

<div id="d1"></div>

<script>

var elem = document.getElementById("d1");

var params = {

type : Two.Types.canvas,

width : 285, // 默认为640px

height : 200 // 默认为480px

};

var two = new Two(params);

two.appendTo(elem);

var circle = two.makeCircle(72, 100, 50);

var rect = two.makeRectangle(213, 100, 100, 100);

// 设置绘制的圆形和矩形的样式

circle.fill = '#FF8000';

circle.stroke = 'orangered';

circle.linewidth = 5;

rect.fill = 'rgb(0, 200, 255)';

rect.opacity = 0.75;

rect.noStroke();// 设置矩形没有描边

two.update();

</script>

</body>

</html>

### 3. 图像分组

调用 Two 对象的 makeGroup() 方法对图像进行分组操作。

* makeGroup() 方法任意图形对象作为参数
* makeGroup() 方法返回分组对象

**值得注意的是：**在统一对一组图像设置后,针对不同图像进行个性化设置。

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Two.js库进行图形分组</title>

<script src="two.js"></script>

</head>

<body>

<div id="d1"></div>

<script>

var elem = document.getElementById("d1");

var two = new Two({

width : 285,

height : 200

}).appendTo(elem);

var circle = two.makeCircle(72, 100, 50);

var rect = two.makeRectangle(213, 100, 100, 100);

var group = two.makeGroup(circle,rect);

group.fill = 'rgb(0, 200, 255)';

group.opacity = 0.75;

group.noStroke();// 设置矩形没有描边

// 针对不同的图形进行个性化样式设置

circle.stroke = 'orangered';

circle.linewidth = 5;

two.update();

</script>

</body>

</html>

### 4. 动画效果（了解）

| **方法名称** | **描述** |
| --- | --- |
| play() | 提供一组循环动画 |
| pause() | 提供中止动画效果 |
| update() | 提供更新当前绘制或设置 |

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Two.js库实现动画效果</title>

<script src="two.js"></script>

</head>

<body>

<div id="d1"></div>

<script>

var elem = document.getElementById("d1");

var two = new Two({

width : 285,

height : 200

}).appendTo(elem);

// 设置圆形和矩形

var circle = two.makeCircle(-70, 0, 50);

var rect = two.makeRectangle(70, 0, 100, 100);

// 设置圆形和矩形的样式

circle.fill = '#FF8000';

rect.fill = 'rgba(0, 200, 255, 0.75)';

// 将圆形和矩形分为一组

var group = two.makeGroup(circle, rect);

// 针对这一组进行样式设置

group.translation.set(two.width / 2, two.height / 2);// 将"画布"平移水平和垂直一半距离

group.scale = 0;// 将圆形和矩形缩放为 0(不显示)

group.noStroke();// 设置圆形和矩形没有边框

// Two对象绑定update事件(方法)

two.bind('update', function(frameCount) {

// 判断如果缩放为原大小,将缩放和旋转设置为 0

if (group.scale > 0.9999) {

group.scale = group.rotation = 0;

}

// 每次执行的增量

var t = (1 - group.scale) \* 0.125;

// 每次执行后缩放值进行累加

group.scale += t;

// 每次执行后旋转至进行累加

group.rotation += t \* 4 \* Math.PI;

}).play();// 开始无限循环动画

</script>

</body>

</html>

**原创声明：**如发布或转载请注明以下作者信息，谢谢！

本文作者 - 金云龙

联系方式 - QQ：2080954535；微信：imlonge；微信公众号：longestory

个人网站：<http://www.longestory.com>