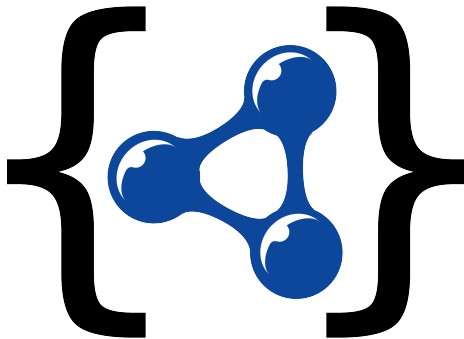


XML 原理与应用

夏天

中国人民大学

CH10 XML 的应用与挑战——SVG & JSON



本章学习目标

- 了解 SVG 的特点
- 能够基于 SVG 进行简单的图形表示
- 了解 d3.js 的功能特点
- 掌握 JSON 的数据结构和类型
- 能够通过 JavaScript 解析 JSON 字符串

目录

- 概述
- SVG
 - SVG 的基本形状
 - SVG 的样式设置
 - SVG 的层与重叠
 - SVG 的透明度
 - 基于 SVG 的 d3.js 图形绘制库
- JSON
 - JSON 的数据结构
 - JSON 的值类型
 - JSON 与 XML 的对比
 - 利用 JavaScript 解析 JSON

10.1 概述

- SVG
 - 可缩放矢量图形 (Scalable Vector Graphics)
 - 1999 年推出
- JSON
 - JavaScript Object Notation
 - 尤其适合作为数据传输格式
- SVG、JSON、XML 是 Web 技术体系的重要组成部分

PNG 与 SVG 文件缩放对比



PNG 文件



SVG 文件



放大后的 PNG 文件



放大后的 SVG 文件

10.2 SVG

- 通过 XML 定义线段、圆形、矩形等形状

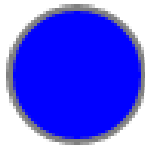
```
1 <svg xmlns="http://www.w3.org/2000/svg" width="50" height="50">
2   <circle cx="25" cy="25" r="22" fill="blue" stroke="gray"
3     stroke-width="2"/>
4 </svg>
```

SVG 示例 I

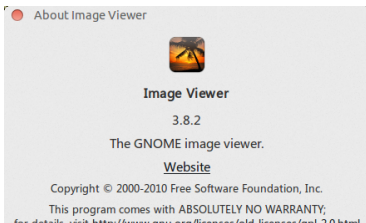
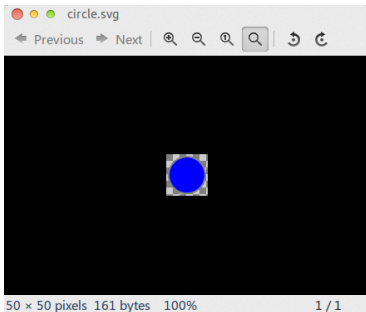
Code 1: "circle.html"

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>SVG Circle</title>
5   </head>
6   <body>
7     <svg width="50" height="50">
8       <circle cx="25" cy="25" r="22" fill="blue" stroke="gray"
9         stroke-width="2"/>
10    </svg>
11  </body>
12 </html>
```


SVG 示例 II



Linux Image Viewer



10.2.1 SVG 的基本形状 I

- 基本形状
 - 矩形：rect
 - 圆形：circle
 - 椭圆：Ellipse
 - 线条：line
 - 文本：Text
- 坐标系

10.2.1 SVG 的基本形状 II



绘制矩形 I

- 代码

```
<rect x="0" y="0" width="300" height="50"/>
```

- 效果



绘制矩形 II

- 代码

```
<rect x="0" y="0" width="300" height="50" rx="15" ry="15"/>
```

- 效果



绘制椭圆 I

- 代码

```
<ellipse cx="250" cy="25" rx="50" ry="25"/>
```

- 效果



绘制线条 I

- 代码

```
<line x1="0" y1="0" x2="350" y2="50" stroke="black"/>
```

- 效果



绘制文本 I

- 代码

```
<text x="250" y="25" font-family="华文隶书" font-size="30"  
fill="navy">可缩放矢量图形</text>
```

- 效果

可缩放矢量图形

10.2.2 SVG 的样式设置

- 默认样式为黑色填充、无 stroke
- fill
 - 填充色，取值为有效的 CSS 颜色值，如颜色名称 red、blue，或者 RGB、RGBA 值。
- stroke
 - 线条的颜色值
- stroke-width
 - 数值，通常采用像素单位
- opacity
 - 透明度，介于 0.0 到 1.0 之间的数值，0.0 表示完全透明，1.0 表示完全不透明
- font-family、font-size
 - 应用于文本

样式关联方式

- 内联方式

```
<circle cx="50" cy="50" r="30" fill="yellow" stroke="orange" stroke-width="10"/>
```

- CSS 样式属性关联方式

```
<circle cx="50" cy="50" r="30" class="pumpkin"/>
```

- 需要设置 pumpkin 样式

- 效果



示例网页 I

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>SVG Style</title>
5     <style>
6       .pumpkin {
7         fill: yellow;
8         stroke: orange;
9         stroke-width: 10;
10      }
11    </style>
12  </head>
13  <body>
14    <svg width="500px" height="200px">
15      <circle cx="50" cy="50" r="30" class="pumpkin"/>
16    </svg>
```

示例网页 II

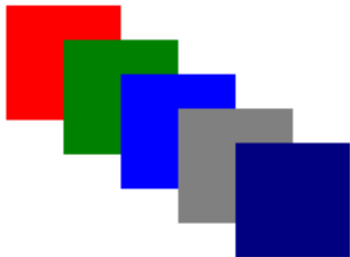
```
17     </body>
```

```
18 </html>
```

10.2.3 SVG 的层与重叠

- 根据绘制的先后顺序予以覆盖

```
1 <rect x="0" y="0" width="50" height="50" fill="red"/>  
2 <rect x="25" y="15" width="50" height="50" fill="green"/>  
3 <rect x="50" y="30" width="50" height="50" fill="blue"/>  
4 <rect x="75" y="45" width="50" height="50" fill="gray"/>  
5 <rect x="100" y="60" width="50" height="50" fill="navy"/>
```



10.2.4 SVG 的透明度

- 使用带 alpha 的 RGB 颜色函数 `rgba()`
- 设置 `opacity` 属性值。

- 代码：

```
1 <circle cx="50" cy="50" r="40" fill="rgba(0, 150, 150, 1.0)"/>
2 <circle cx="100" cy="50" r="40" fill="rgba(0, 0, 255, 0.75)"/>
3 <circle cx="150" cy="50" r="40" fill="rgba(0, 255, 0, 0.5)"/>
4 <circle cx="200" cy="50" r="40" fill="rgba(255, 255, 0, 0.6)"/>
5 <circle cx="250" cy="50" r="40" fill="rgba(255, 0, 0, 0.3)"/>
```

- 显示效果：



rgba() II

- 代码：

```
1 <circle cx="50" cy="50" r="40" fill="rgba(0, 150, 150, 1.0)"
2   stroke="rgba(0, 255, 0, 0.25)" stroke-width="10"/>
3 <circle cx="100" cy="50" r="40" fill="rgba(0, 0, 255, 0.75)"
4   stroke="rgba(0, 0, 255, 0.25)" stroke-width="10"/>
5 <circle cx="150" cy="50" r="40" fill="rgba(0, 255, 0, 0.5)"
6   stroke="rgba(0, 0, 255, 0.25)" stroke-width="10"/>
7 <circle cx="200" cy="50" r="40" fill="rgba(255, 255, 0, 0.6)"
8   stroke="rgba(255, 0, 0, 0.25)" stroke-width="10"/>
9 <circle cx="250" cy="50" r="40" fill="rgba(255, 0, 0, 0.3)"
10  stroke="rgba(55, 255, 0, 0.25)" stroke-width="10"/>
```

- 显示效果：

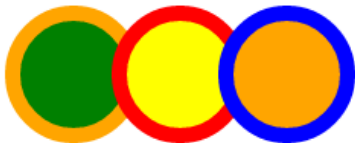


opacity I

- opacity 设置为 1.0 表示完全不透明，为默认值

```
1 <circle cx="50" cy="50" r="40" fill="green" stroke="orange"
  stroke-width="10" opacity="1.0"/>
2 <circle cx="120" cy="50" r="40" fill="yellow" stroke="red"
  stroke-width="10"/>
3 <circle cx="190" cy="50" r="40" fill="orange" stroke="blue"
  stroke-width="10"/>
```

- 显示效果：



opacity II

- 代码：

```
1 <circle cx="50" cy="50" r="40" fill="green" stroke="orange"  
  stroke-width="10" opacity="0.5"/>  
2 <circle cx="120" cy="50" r="40" fill="yellow" stroke="red"  
  stroke-width="10" opacity="0.3"/>  
3 <circle cx="190" cy="50" r="40" fill="orange" stroke="blue"  
  stroke-width="10" opacity="0.1"/>
```

- 显示效果：



同时设置 rgba 与 opacity

- 代码：

```
1 <circle cx="50" cy="50" r="40" fill="green" stroke="orange"
   stroke-width="10" opacity="0.5"/>
2 <circle cx="120" cy="50" r="40" fill="yellow"
   stroke="rgba(255, 0, 0, 0.5)" stroke-width="10" opacity="0.3"/>
3
4 <circle cx="190" cy="50" r="40" fill="orange" stroke="blue"
   stroke-width="10" opacity="0.1"/>
```

- 显示效果：



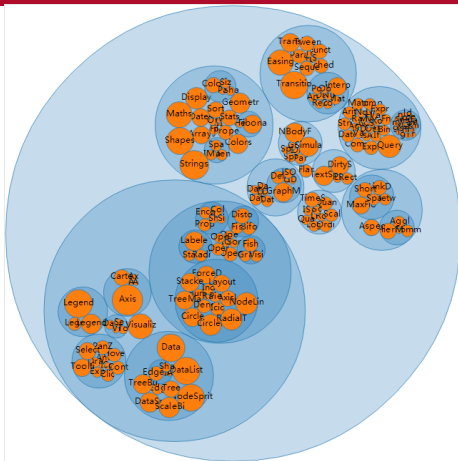
10.2.5 d3.js

- d3
 - Data-Driven Documents 的缩写
 - Data 表示用户提供的数据
 - Documents 代表可以被浏览器解析呈现的文档，
 - 支持 SVG

效果图1



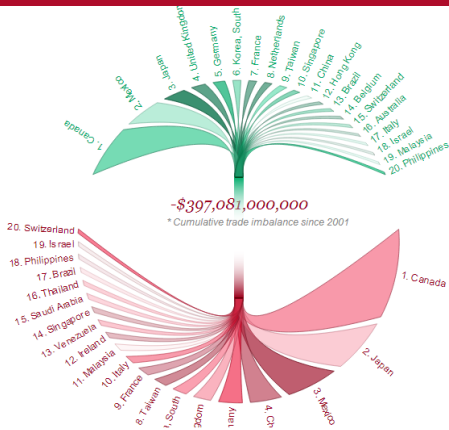
效果图 II



效果图 111



效果图 IV



如何引用 d3.js 进行测试

- 下载最新版本的 d3.js
- 在网页中引用 d3.js
- 在 Web Server 中测试 (可选)

基于 d3.js 绘制 SVG 图形 I

- 创建 svg 元素，设置属性

```
var svg = d3.select("body").append("svg");  
svg.attr("width", 500);  
svg.attr("height", 200);
```

- 采用链式语法改写

```
var svg = d3.select("body")  
  .append("svg")  
  .attr("width", 500)  
  .attr("height", 200);
```

- 继续增加若干圆形

基于 d3.js 绘制 SVG 图形 II

```
var dataset = [ 20, 15, 10, 5, 10, 15, 20 ];  
var circles = svg.selectAll("circle")  
    .data(dataset)  
    .enter()  
    .append("circle");
```

- 继续指定图形的位置和大小

```
circles.attr("cx", function(d, i) {  
    return (i * 50) + 25;  
})  
.attr("cy", 50)  
.attr("r", function(d) {  
    return d;  
});
```

基于 d3.js 绘制 SVG 图形 III

– 代码中 d 表示传入函数的绑定数据， i 代表图形顺序编号，起始编号为 0

- 显示效果



- 继续指定圆的样式

```
.attr("fill", "orange")  
.attr("stroke", "red")  
.attr("stroke-width", function(d, i) {  
    return d/3;  
});
```

基于 d3.js 绘制 SVG 图形 IV

- 显示效果



代码清单 I

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>D3 Circles</title>
6     <script type="text/javascript" src="d3/d3.v3.js"></script>
7   </head>
8   <body>
9     <script type="text/javascript">
10       var svg = d3.select("body")
11         .append("svg")
12         .attr("width", 500)
13         .attr("height", 200);
14       var dataset = [20, 15, 10, 5, 10, 15, 20];
15       var circles = svg.selectAll("circle")
16         .data(dataset)
```


代码清单 II

```
17         .enter()
18         .append("circle");
19
20     circles.attr("cx", function(d, i) {
21         return (i * 50) + 25;
22     })
23     .attr("cy", 50)
24     .attr("r", function(d) {
25         return d;
26     })
27     .attr("fill", "orange")
28     .attr("stroke", "red")
29     .attr("stroke-width", function(d, i) {
30         return d/3;
31     });
32 </script>
```

代码清单 III

```
33     </body>
```

```
34 </html>
```

10.3 数据传输的挑战者—JSON

- JSON

- JavaScript Object Notation
- 一种轻量级、基于文本、语言无关的数据交换格式
- JavaScript (Standard ECMA-262) 的一个子集
- 重要贡献者：Douglas Crockford
- 2006 年 7 月——RFC 4627

10.3.1 JSON 的数据结构

- 无序的键值对集合

```
{ "a":1,"b":2,"c":3 }
```

- 值的有序列表

```
[ 1, 2, 3, "hello world" ]
```

10.3.2 JSON 的值类型

- 字符串类型
 - JSON 字符串采用双引号包括所表示的字符内容，字符内容本身包含的双引号使用反斜线进行转义；
- 数值类型
- 对象类型
 - 键值对的无序集合；
- 数组类型
 - 值的有序列表；
- 布尔类型
 - true 表示真、false 表示假；
- 空类型 null

JSON 示例 I

Code 2: contact.json

```
1 {  
2   "name": "王某某",  
3   "age": 30,  
4   "spouse": {  
5     "name": "张某某",  
6     "age": 28  
7   },  
8   "addresses": [  
9     {  
10      "description": "工作住址",  
11      "street": "胜利大街9号",  
12      "city": "潍坊",  
13      "province": "山东"  
14    }  
15  ]  
16 }
```

JSON 示例 II

```
15 ],
16 "phoneNumbers": [
17     {
18         "description": "办公电话",
19         "number": "6666-7777"
20     },
21     {
22         "description": "手机",
23         "number": "1XX-1234-5678"
24     }
25 ]
26 }
```

10.3.3 JSON 与 XML 的对比

- XML 比 JSON 更易于人工阅读
- XML 是典型的标记语言，而 JSON 不是
- JSON 格式简洁，文件的体积较小，有利于数据传输。
- JSON 与 JavaScript 结合紧密，构造和解析都比较容易，而通过 JavaScript 解析 XML，则需要借助于额外的库

contact.json 对应的 XML I

Code 3: contact.xml

```
1 <contact>
2   <name>王某某</name>
3   <age>30</age>
4   <spouse>
5     <name>张某某</name>
6     <age>28</age>
7   </spouse>
8   <addresses>
9     <address>
10      <description>工作住址</description>
11      <street>胜利大街9号</street>
12      <city>潍坊</city>
13      <province>山东</province>
14    </address>
```

contact.json 对应的 XML II

```
15 </addresses>
16 <phoneNumbers>
17   <phoneNumber>
18     <description>办公电话</description>
19     <number>6666-7777</number>
20   </phoneNumber>
21   <phoneNumber>
22     <description>手机</description>
23     <number>1XX-1234-5678</number>
24   </phoneNumber>
25 </phoneNumbers>
26 </contact>
```

10.3.4 利用 JavaScript 解析 JSON

- 浏览器内置的 JSON.parse()

```
JSON.parse(text [, reviver])
```

示例 I

- 在浏览器控制台输入以下代码

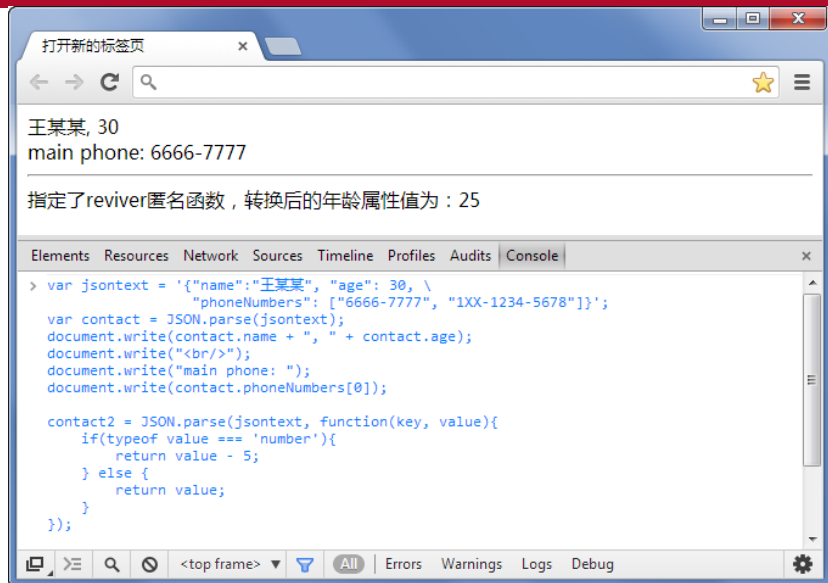
```
var jsontext = '{"name": "王某某", "age": 30, \\  
    "phoneNumbers": ["6666-7777", "1XX-1234-5678"]}';  
var contact = JSON.parse(jsontext);  
document.write(contact.name + ", " + contact.age);  
document.write("<br/>");  
document.write("main phone: ");  
document.write(contact.phoneNumbers[0]);  
  
contact2 = JSON.parse(jsontext, function(key, value){  
    if(typeof value === 'number'){  
        return value - 5;  
    } else {  
        return value;  
    }  
})
```

示例 II

```
});  
document.write("<br/><hr/>  
    指定了reviver匿名函数，转换后的年龄属性值为：");  
document.write(contact2.age);
```

- 浏览器解析结果

示例 III



END

