



D3.js入门教程

极客学院出版

前言

近年来，可视化越来越流行，许多报刊杂志、门户网站、新闻媒体都大量使用可视化技术，使得复杂的数据和文字变得十分容易理解，有一句谚语“一张图片价值于一千个字”，正是如此。D3 正是数据可视化工具中的佼佼者，基于 JavaScript 开发，项目托管于 GitHub。从 D3 诞生以来，不断受到好评，在 GitHub 上的项目仓库排行榜也不断上升。本教程的目的在于提供一个简单易懂的入门教程，希望读者能够喜欢。

适用人群

本教材适合需要在网页前端做数据可视化图表的开发，以及需要了解并学习 D3.js 的读者。

学习前提

本教程适合有一定编程基础的人阅读。对于有 JavaScript 基础的程序员来说，阅读本教程会比较轻松。没有 JavaScript 基础也没关系，可以先在 W3Cschool 上了解一些语法，此后遇到问题再查询亦可。

目录

前言	1
第 1 章 作者简介	5
版权许可	7
第 2 章 第一章 简介和安装	8
D3 是什么	10
为什么要数据可视化	11
D3 有多受欢迎	12
如何学习和使用 D3	13
学习 D3 需要什么预备知识	14
需要什么工具	15
第 3 章 第二章 第一个程序 HelloWorld	16
HTML 是怎么输出 HelloWorld 的	18
用 JavaScript 来更改 HelloWorld	19
用 D3 来更改 HelloWorld	20
第 4 章 第三章 选择元素和绑定数据	22
如何选择元素	24
如何绑定数据	25
第 5 章 第四章 选择、插入、删除元素	27
如何选择元素	24
插入元素	32
删除元素	33
第 6 章 第四章 选择、插入、删除元素	27
如何选择元素	24

	插入元素	32
	删除元素	33
第 7 章	第六章 比例尺的使用	41
	为什么需要比例尺	43
	有哪些比例尺	44
	给柱形图添加比例尺	46
	源代码	47
第 8 章	第七章 坐标轴	48
	坐标轴由什么构成	50
	定义坐标轴	52
	在 SVG 中添加坐标轴	53
	设定坐标轴的样式和位置	54
	源代码	47
第 9 章	第八章 完整的柱形图	56
	添加 SVG 画布	58
	定义坐标轴	52
	添加矩形和文字元素	60
	添加坐标轴的元素	61
	源代码	47
第 10 章	第九章 让图表动起来	63
	什么是动态效果	65
	实现动态的方法	66
	实现简单的动态效果	68
	给柱形图加上动态效果	69
	源代码	47
第 11 章	第十章 理解 Update、Enter、Exit	71
	什么是 Update、Enter、Exit	73

Update 和 Enter 的使用.....	75
Update 和 Exit 的使用	76
源代码.....	47



作者简介



馒头华华，喜读儒家经典、三国演义等中国古籍，喜看古装剧，热爱中华优秀传统文化。闲暇时练习书法，养气修身。工作之余喜欢旅游、健身、游泳。不喜与人相争，近来潜心研读儒佛道三家学说。

2014 年与好友创办 [OUR D3.JS 数据可视化专题站](#)，以 D3.js 为题发表一系列教学文章，获得读者好评。本教程即以专题站的文章为基础，整理简化而成。

版权许可

本教程首发于 [OUR D3.JS](#)，经作者同意，授权给极客学院转载。

本教程依据知识共享协议 3.0，在保有

署名（BY）-非商业性（NC）-禁止演绎（ND）

权利的情况下发布。具体解释如下：

署名 - 您（使用者）可以复制、发行、展览、表演、放映、广播或通过信息网络传播本作品；您必须按照作者或者许可人指定的方式对作品进行署名。

非商业性 - 您可以自由复制、散布、展示及演出本作品；您不得为商业目的而使用本作品。

禁止演绎 - 您可以自由复制、散布、展示及演出本作品；您不得改变、转变或更改本作品。



T



2

第一章 简介和安装



近年来，可视化越来越流行，许多报刊杂志、门户网站、新闻、媒体都大量使用可视化技术，使得复杂的数据和文字变得十分容易理解，有一句谚语“一张图片价值于一千个字”，的确是名副其实。各种数据可视化工具也如井喷式地发展，D3 正是其中的佼佼者。

D3 是什么

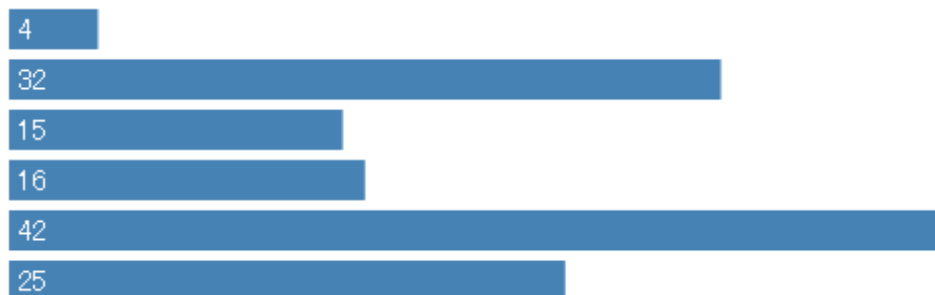
D3 的全称是 (Data-Driven Documents)，顾名思义可以知道是一个被数据驱动文档。听名字有点抽象，说简单一点，其实就是一个 JavaScript 的函数库，使用它主要是用来做数据可视化的。如果你不知道什么是 JavaScript，请先学习一点 JavaScript 的基础知识。

[W3School 的 JavaScript 教程](#)

JavaScript 文件的后缀名通常为 .js，故 D3 也常使用 D3.js 称呼。D3 提供了各种简单易用的函数，大大简化了 JavaScript 操作数据的难度。由于它本质上是 JavaScript，所以用 JavaScript 也是可以实现所有功能的，但它能大大减小你的工作量，尤其是在数据可视化方面，D3 已经将生成可视化的复杂步骤精简到了几个简单的函数，你只需要输入几个简单的数据，就能够转换为各种绚丽的图形。有过 JavaScript 基础的朋友一定很容易理解它。

为什么要数据可视化

现在有一组数据，【 4 ， 32 ， 15 ， 16 ， 42 ， 25 】，你能一眼看出它们的大小关系吗？当然这里的数据不算多，有那眼疾手快的家伙站出来说我能一眼看出来！但更直观的是用图形显示，如下图：










图片 2.1 柱形图

通过图形的显示，能很清楚地知道他们的大小关系。当然，D3 能力远不止如此，这只是一个很小的应用。把枯燥乏味复杂的数据，用简单明了的图形表示出来，这就是**数据可视化**。

D3 有多受欢迎

D3 是一个开源项目，作者是纽约时报的工程师。D3 项目的代码托管于 GitHub（一个开发管理平台，目前已经是全世界最流行的代码托管平台，云集了来自世界各地的优秀工程师）。

Repositories Ranking

	1. twbs/bootstrap	★ 77733
	2. vhf/free-programming-books	★ 35984
	3. angular/angular.js	★ 35311
	4. joyent/node	★ 34715
	5. mbostock/d3	★ 34492
	6. jquery/jquery	★ 33394
	7. FortAwesome/Font-Awesome	★ 30484

图片 2.2 Github仓库排名

在 GitHub 上最受关注的项目有哪些呢？

JQuery 的名声够大了，但排名第 6，D3 却排名第 5。

如何学习和使用 D3

以下是几个学习 D3 的站点：

- 官方网站

<http://d3js.org/>

包含有很多示例和 API，要想得心应手的使用 D3，熟悉 API 是避不开的。

- Mike Bostock 的博客和作品展示板

<http://bost.ocks.org/mike/>

- OUR D3.JS 数据可视化专题站

<http://www.ourd3js.com/>

笔者开设的站点，包含有 D3 的一系列教程。

D3 是一个 JavaScript 函数库，并不需要通常所说的“安装”。它只有一个文件，在 HTML 中引用即可。有两种方法：

（1）下载 D3.js 的文件

- [d3.zip](#)

解压后，在 HTML 文件中包含相关的 js 文件即可。

（2）直接包含网络的链接

```
<script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>
```

这种方法比较简单，但要保持网络连接有效。

学习 D3 需要什么预备知识

想要通过 D3 来开启数据可视化之旅的朋友，需要什么预备知识呢？

- HTML：超文本标记语言，用于设定网页的内容
- CSS：层叠样式表，用于设定网页的样式
- JavaScript：一种直译式脚本语言，用于设定网页的行为
- DOM：文档对象模型，用于修改文档的内容和结构
- SVG：可缩放矢量图形，用于绘制可视化的图形

路人甲：额，我需要学那么多才能开始学 D3 吗？心理压力有点点...大

馒头华华：不必，完全可以直接学 D3，遇到不明白的地方，再看相关内容即可

路人乙：HTML、CSS 啥的，我从来都没用过，也没有关系吗？

馒头华华：只要在 W3School，分别看看这几个词是什么意思，是用来干什么的，再看几个简单例子即可，没有必要全掌握了再学习 D3。

需要什么工具

制作网页常用的工具即可。

记事本软件：Notepad++、Editplus、Sublime Text 等，选择自己喜欢的即可。

浏览器：IE9 以上、Firefox、Chrome 等，推荐用 Chrome

服务器软件：Apache、Tomcat 等

其中，服务器软件可能不是必须的，不过 D3 中有些函数需要将 HTML 文件放置于服务器目录下，才能正常使用，关于这点以后会再做说明。

好了，可以开始你的 D3 之旅了。祝你好运。



3



第二章 第一个程序 HelloWorld



先尝试用 D3 写第一个 HelloWorld 程序。学编程入门的第一个程序都是在屏幕上输出 HelloWorld，本课稍微有些不同，不是单纯的输出。

HTML 是怎么输出 HelloWorld 的

都知道 HTML 吧，如果不知道请下百度一下吧。在 HTML 中输出 HelloWorld 是怎样的呢，先看下面的代码。

```
<html>
<head>
  <meta charset="utf-8">
  <title>HelloWorld</title>
</head>
<body>
  <p>Hello World 1</p>
  <p>Hello World 2</p>
</body>
</html>
```

如果你学习过 HTML，应该知道会在浏览器中输出两行文字，如下图：

Hello World 1

Hello World 2

图片 3.1 html输出两行段落

用 JavaScript 来更改 HelloWorld

对于上面输出的内容，如果想用 JavaScript 来更改这两行文字，怎么办呢？我们会添加代码变为：

```
<html>
<head>
  <meta charset="utf-8">
  <title>HelloWorld</title>
</head>
<body>
  <p>Hello World 1</p>
  <p>Hello World 2</p>
  <script>
    var paragraphs = document.getElementsByTagName("p");
    for (var i = 0; i < paragraphs.length; i++) {
      var paragraph = paragraphs.item(i);
      paragraph.innerHTML = "I like dog.";
    }
  </script>
</body>
</html>
```

结果变为：

I like dog

I like dog

图片 3.2 用 JavaScript 更改段落元素

可以看到，使用 JavaScript，我们添加了4行代码。

用 D3 来更改 HelloWorld

如果使用 D3.js 来修改这两行呢？只需添加一行代码即可。注意不要忘了引用 D3.js 源文件。

```
<html>
<head>
  <meta charset="utf-8">
  <title>HelloWorld</title>
</head>
<body>
  <p>Hello World 1</p>
  <p>Hello World 2</p>
  <script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>
  <script>
    d3.select("body").selectAll("p").text("www.ourd3js.com");
  </script>
</body>
</html>
```

结果会变为：

www.ourd3js.com

www.ourd3js.com

图片 3.3 用 D3.js 更改段落元素

也实现同样的功能，但是却显得十分简洁。不错，其实 D3.js 中的所有功能在 JavaScript 中都能实现，它仅仅是一个函数库而已。D3 所做的事就是减轻你的工作量，以及使你的代码十分简单易懂。

接下来改变字体的颜色和大小，稍微修改上述代码：

```
```html //选择中所有的
```

```
<
```

```
p>, 其文本内容为 www.ourd3js.com, 选择集保存在变量 p 中 var p = d3.select("body").selectAll("p").text("www.ourd3js.com");
```

```
//修改段落的颜色和字体大小 p.style("color","red").style("font-size","72px"); ```
```

上面的代码是先将选中的元素赋值给变量 p，然后通过变量 p 来改变样式，这样可以使代码更整洁。

这里涉及一个概念：**选择集**

使用 d3.select() 或 d3.selectAll() 选择元素后返回的对象，就是**选择集**。

另外，有人会发现，D3 能够连续不断地调用函数，形如：

```
d3.select().selectAll().text()
```

这称为链式语法，和 JQuery 的语法很像，常用 JQuery 的朋友一定会感到很亲切。



4

## 第三章 选择元素和绑定数据



选择元素和绑定数据是 D3 最基础的内容，本文将对其进行一个简单的介绍。



图片 4.1 绑定数据



## 如何选择元素

---

在 D3 中，用于选择元素的函数有两个：

- `d3.select()`：是选择所有指定元素的第一个
- `d3.selectAll()`：是选择指定元素的全部

这两个函数返回的结果称为选择集。

例如，选择集的常见用法如下。

```
var body = d3.select("body"); //选择文档中的body元素
var p1 = body.select("p"); //选择body中的第一个p元素
var p = body.selectAll("p"); //选择body中的所有p元素
var svg = body.select("svg"); //选择body中的svg元素
var rects = svg.selectAll("rect"); //选择svg中所有的svg元素
```

选择集和绑定数据通常是一起使用的。

## 如何绑定数据

D3 有一个很独特的功能：能将数据绑定到 DOM 上，也就是绑定到文档上。这么说可能不好理解，例如网页中有段落元素 `p` 和一个整数 5，于是可以将整数 5 与 `p` 绑定到一起。绑定之后，当需要依靠这个数据才操作元素的时候，会很方便。

D3 中是通过以下两个函数来绑定数据的：

- `datum()`：绑定一个数据到选择集上
- `data()`：绑定一个数组到选择集上，数组的各项值分别与选择集的各元素绑定

相对而言，`data()` 比较常用。

假设现在有三个段落元素如下。

```
<p>Apple</p>
<p>Pear</p>
<p>Banana</p>
```

### `datum()`

假设有一字符串 `China`，要将此字符串分别与三个段落元素绑定，代码如下：

```
var str = "China";

var body = d3.select("body");
var p = body.selectAll("p");

p.datum(str);

p.text(function(d, i){
 return "第 " + i + " 个元素绑定的数据是 " + d;
});
```

绑定数据后，使用此数据来修改三个段落元素的内容，其结果如下：

```
第 0 个元素绑定的数据是 China
第 1 个元素绑定的数据是 China
第 2 个元素绑定的数据是 China
```

在上面的代码中，用到了一个无名函数 `function(d, i)`。当选择集需要使用被绑定的数据时，常需要这么使用。其包含两个参数，其中：

- `d` 代表数据，也就是与某元素绑定的数据。

- $i$  代表索引，代表数据的索引号，从 0 开始。

例如，上述例子中：第 0 个元素 apple 绑定的数据是 China。

## data()

有一个数组，接下来要分别将数组的各元素绑定到三个段落元素上。

```
var dataset = ["I like dog", "I like cat", "I like snake"];
```

绑定之后，其对应关系的要求为：

- Apple 与 I like dog 绑定
- Pear 与 I like cat 绑定
- Banana 与 I like snake 绑定

调用 data() 绑定数据，并替换三个段落元素的字符串为被绑定的字符串，代码如下：

```
var body = d3.select("body");
var p = body.selectAll("p");

p.data(dataset)
 .text(function(d, i){
 return d;
 });
```

这段代码也用到了一个匿名函数 function(d, i)，其对应的情况如下：

- 当  $i == 0$  时， $d$  为 I like dog。
- 当  $i == 1$  时， $d$  为 I like cat。
- 当  $i == 2$  时， $d$  为 I like snake。

此时，三个段落元素与数组 dataset 的三个字符串是一一对应的，因此，在函数 function(d, i) 直接 return d 即可。

结果自然是三个段的文字分别变成了数组的三个字符串。

```
I like dog
I like cat
I like snake
```

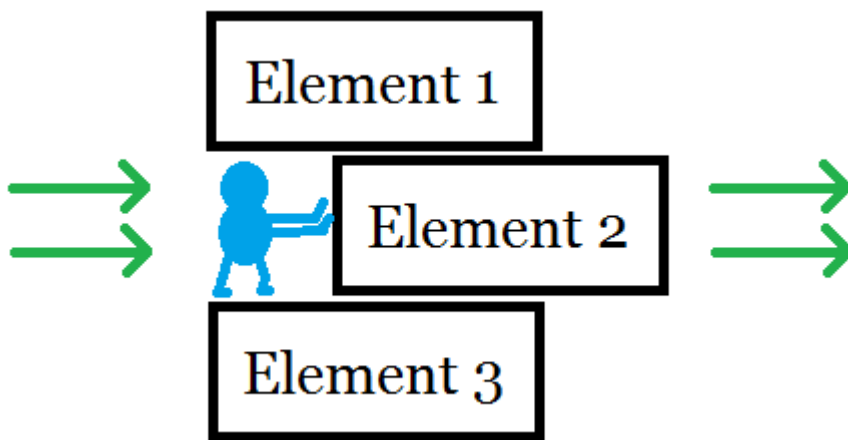


5

## 第四章 选择、插入、删除元素



本章介绍在 D3 中选择、插入、删除元素的方法。



图片 5.1 插入删除元素

## 如何选择元素

---

上一章已经讲解了 `select` 和 `selectAll`，以及选择集的概念。本节具体讲解这两个函数的用法。

假设在 `body` 中有三个段落元素：

```
<p>Apple</p>
<p>Pear</p>
<p>Banana</p>
```

现在，要分别完成以下四种选择元素的任务。

### 选择第一个 p 元素

使用 `select`，参数传入 `p` 即可，如此返回的是第一个 `p` 元素。

```
t("p");
p1.style("color","red");
```

结果如下图，被选择的元素标记为红色。

Apple

Pear

Banana

图片 5.2 选择第一个元素

### 选择三个 p 元素

使用 `selectAll` 选择 `body` 中所有的 `p` 元素。

```
var p = body.selectAll("p");
p.style("color","red");
```

结果如下：

Apple

Pear

Banana

图片 5.3 选择三个元素

## 选择第二个 p 元素

有不少方法，一种比较简单的是给第二个元素添加一个 id 号。

Pear

然后，使用 select 选择元素，注意参数中 id 名称前要加 # 号。

```
var p2 = body.select("#myid");
p2.style("color","red");
```

结果如下：

Apple

Pear

Banana

图片 5.4 选择第二个元素

## 选择后两个 p 元素

给后两个元素添加 class，

```
<p class="myclass">Pear</p>
<p class="myclass">Banana</p>
```

由于需要选择多个元素，要用 selectAll。注意参数，class 名称前要加一个点。

```
var p = body.selectAll(".myclass");
p.style("color","red");
```

结果如下：

Apple

Pear

Banana

图片 5.5 选择后两个元素

关于 `select` 和 `selectAll` 的参数，其实是符合 CSS 选择器的条件的，即用“井号（#）”表示 `id`，用“点（.）”表示 `class`。

此外，对于已经绑定了数据的选择集，还有一种选择元素的方法，那就是灵活运用 `function(d, i)`。我们已经知道参数 `i` 是代表索引号的，于是便可以用条件判定语句来指定执行的元素。



## 插入元素

---

插入元素涉及的函数有两个：

- `append()`：在选择集末尾插入元素
- `insert()`：在选择集前面插入元素

假设有三个段落元素，与上文相同。

### append()

```
body.append("p")
 .text("append p element");
```

在 `body` 的末尾添加一个 `p` 元素，结果为：

```
Apple
Pear
Banana
append p element
```

### insert()

在 `body` 中 `id` 为 `myid` 的元素前添加一个段落元素。

```
body.insert("p", "#myid")
 .text("insert p element");
```

已经指定了 `Pear` 段落的 `id` 为 `myid`，因此结果如下。

```
Apple
insert p element
Pear
Banana
```

## 删除元素

---

删除一个元素时，对于选择的元素，使用 remove 即可，例如：

```
var p = body.select("#myid");
p.remove();
```

如此即可删除指定 id 的段落元素。

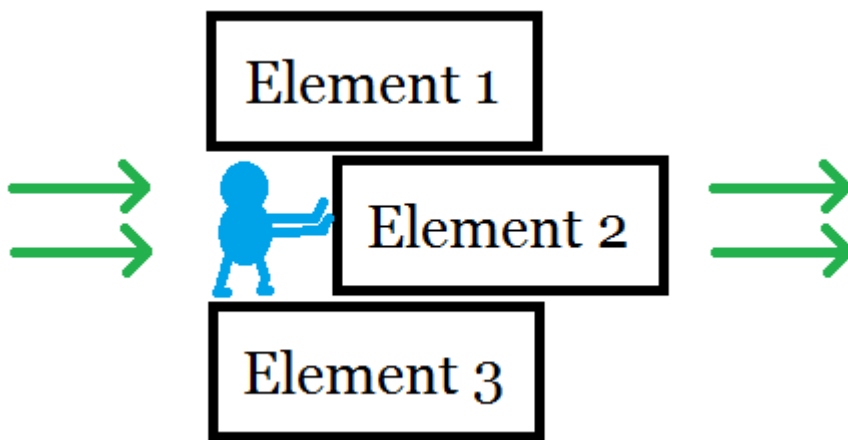


6

## 第四章 选择、插入、删除元素



本章介绍在 D3 中选择、插入、删除元素的方法。



图片 6.1 插入删除元素

## 如何选择元素

---

上一章已经讲解了 `select` 和 `selectAll`，以及选择集的概念。本节具体讲解这两个函数的用法。

假设在 `body` 中有三个段落元素：

```
<p>Apple</p>
<p>Pear</p>
<p>Banana</p>
```

现在，要分别完成以下四种选择元素的任务。

### 选择第一个 p 元素

使用 `select`，参数传入 `p` 即可，如此返回的是第一个 `p` 元素。

```
t("p");
p1.style("color","red");
```

结果如下图，被选择的元素标记为红色。

Apple

Pear

Banana

图片 6.2 选择第一个元素

### 选择三个 p 元素

使用 `selectAll` 选择 `body` 中所有的 `p` 元素。

```
var p = body.selectAll("p");
p.style("color","red");
```

结果如下：

Apple

Pear

Banana

图片 6.3 选择三个元素

## 选择第二个 p 元素

有不少方法，一种比较简单的是给第二个元素添加一个 id 号。

Pear

然后，使用 select 选择元素，注意参数中 id 名称前要加 # 号。

```
var p2 = body.select("#myid");
p2.style("color","red");
```

结果如下：

Apple

Pear

Banana

图片 6.4 选择第二个元素

## 选择后两个 p 元素

给后两个元素添加 class，

```
<p class="myclass">Pear</p>
<p class="myclass">Banana</p>
```

由于需要选择多个元素，要用 selectAll。注意参数，class 名称前要加一个点。

```
var p = body.selectAll(".myclass");
p.style("color","red");
```

结果如下：

Apple

Pear

Banana

图片 6.5 选择后两个元素

关于 `select` 和 `selectAll` 的参数，其实是符合 CSS 选择器的条件的，即用“井号（#）”表示 `id`，用“点（.）”表示 `class`。

此外，对于已经绑定了数据的选择集，还有一种选择元素的方法，那就是灵活运用 `function(d, i)`。我们已经知道参数 `i` 是代表索引号的，于是便可以用条件判定语句来指定执行的元素。

## 插入元素

---

插入元素涉及的函数有两个：

- `append()`：在选择集末尾插入元素
- `insert()`：在选择集前面插入元素

假设有三个段落元素，与上文相同。

### append()

```
body.append("p")
 .text("append p element");
```

在 `body` 的末尾添加一个 `p` 元素，结果为：

```
Apple
Pear
Banana
append p element
```

### insert()

在 `body` 中 `id` 为 `myid` 的元素前添加一个段落元素。

```
body.insert("p", "#myid")
 .text("insert p element");
```

已经指定了 `Pear` 段落的 `id` 为 `myid`，因此结果如下。

```
Apple
insert p element
Pear
Banana
```



## 删除元素

---

删除一个元素时，对于选择的元素，使用 `remove` 即可，例如：

```
var p = body.select("#myid");
p.remove();
```

如此即可删除指定 `id` 的段落元素。

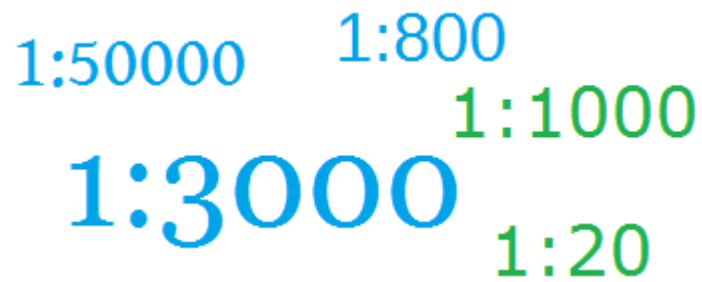


7

## 第六章 比例尺的使用



比例尺是 D3 中很重要的一个概念，上一章里曾经提到过直接用数值的大小来代表像素不是一种好方法，本章正是要解决此问题。



1:50000 1:800  
1:1000  
1:3000  
1:20

图片 7.1 柱形图

## 为什么需要比例尺

---

上一章制作了一个柱形图，当时有一个数组：

```
var dataset = [250 , 210 , 170 , 130 , 90];
```

绘图时，直接使用 250 给矩形的宽度赋值，即矩形的宽度就是 250 个像素。

此方式非常具有局限性，如果数值过大或过小，例如：

```
var dataset_1 = [2.5 , 2.1 , 1.7 , 1.3 , 0.9];
var dataset_2 = [2500 , 2100 , 1700 , 1300 , 900];
```

对以上两个数组，绝不可能用 2.5 个像素来代表矩形的宽度，那样根本看不见；也不可能用 2500 个像素来代表矩形的宽度，因为画布没有那么长。

于是，我们需要一种计算关系，能够：

将某一区域的值映射到另一区域，其大小关系不变。

这就是比例尺（Scale）。

## 有哪些比例尺

比例尺，很像数学中的函数。例如，对于一个一元二次函数，有  $x$  和  $y$  两个未知数，当  $x$  的值确定时， $y$  的值也就确定了。

在数学中， $x$  的范围被称为**定义域**， $y$  的范围被称为**值域**。

D3 中的比例尺，也有定义域和值域，分别被称为 `domain` 和 `range`。开发者需要指定 `domain` 和 `range` 的范围，如此即可得到一个计算关系。

D3 提供了多种比例尺，下面介绍最常用的两种。

### 线性比例尺

线性比例尺，能将一个连续的区间，映射到另一区间。要解决柱形图宽度的问题，就需要线性比例尺。

假设有以下数组：

```
var dataset = [1.2, 2.3, 0.9, 1.5, 3.3];
```

现有要求如下：

将 `dataset` 中最小的值，映射成 0；将最大的值，映射成 300。

代码如下：

```
var min = d3.min(dataset);
var max = d3.max(dataset);

var linear = d3.scale.linear()
 .domain([min, max])
 .range([0, 300]);

linear(0.9); //返回 0
linear(2.3); //返回 175
linear(3.3); //返回 300
```

其中，`d3.scale.linear()` 返回一个线性比例尺。`domain()` 和 `range()` 分别设定比例尺的定义域和值域。在这里还用到了两个函数，它们经常与比例尺一起出现：

- `d3.max()`
- `d3.min()`

这两个函数能够求数组的最大值和最小值，是 D3 提供的。按照以上代码，

比例尺的定义域 domain 为: [0.9, 3.3]

比例尺的值域 range 为: [0, 300]

因此, 当输入 0.9 时, 返回 0; 当输入 3.3 时, 返回 300。当输入 2.3 时呢? 返回 175, 这是按照线性函数的规则计算的。

有一点请大家记住:

d3.scale.linear() 的返回值, 是可以当做函数来使用的。因此, 才有这样的用法: linear(0.9)。

## 序数比例尺

有时候, 定义域和值域不一定是连续的。例如, 有两个数组:

```
var index = [0, 1, 2, 3, 4];
var color = ["red", "blue", "green", "yellow", "black"];
```

我们希望 0 对应颜色 red, 1 对应 blue, 依次类推。

但是, 这些值都是离散的, 线性比例尺不适合, 需要用到序数比例尺。

```
var ordinal = d3.scale.ordinal()
 .domain(index)
 .range(color);

ordinal(0); //返回 red
ordinal(2); //返回 green
ordinal(4); //返回 black
```

用法与线性比例尺是类似的。

## 给柱形图添加比例尺

---

在上一章的基础上，修改一下数组，再定义一个线性比例尺。

```
var dataset = [2.5 , 2.1 , 1.7 , 1.3 , 0.9];

var linear = d3.scale.linear()
 .domain([0, d3.max(dataset)])
 .range([0, 250]);
```

其后，按照上一章的方法添加矩形，在给矩形设置宽度的时候，应用比例尺。

```
var rectHeight = 25; //每个矩形所占的像素高度(包括空白)

svg.selectAll("rect")
 .data(dataset)
 .enter()
 .append("rect")
 .attr("x", 20)
 .attr("y", function(d, i){
 return i * rectHeight;
 })
 .attr("width", function(d){
 return linear(d); //在这里用比例尺
 })
 .attr("height", rectHeight-2)
 .attr("fill", "steelblue");
```

如此一来，所有的数值，都按照同一个线性比例尺的关系来计算宽度，因此数值之间的大小关系不变。

## 源代码

---

下载地址: [rm40.zip](#)

展示地址: <http://www.ourd3js.com/demo/rm/R-4.0/UseScaleInChart.html>

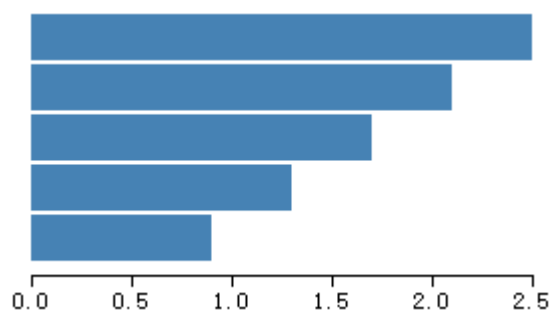




## 第七章 坐标轴



坐标轴，是可视化图表中经常出现的一种图形，由一些列线段和刻度组成。坐标轴在 SVG 中是没有现成的图形元素的，需要用其他的元素组合构成。D3 提供了坐标轴的组件，如此在 SVG 画布中绘制坐标轴变得像添加一个普通元素一样简单。



图片 8.1 柱形图

## 坐标轴由什么构成

在 SVG 画布的预定义元素里，有六种基本图形：

- 矩形
- 圆形
- 椭圆
- 线段
- 折线
- 多边形

另外，还有一种比较特殊，也是功能最强的元素：

- 路径

画布中的所有图形，都是由以上七种元素组成。

显然，这里面没有\*\*坐标轴\*\*这种元素。如果有的话，我们可以采用类似以下的方式定义：

```
<axis x1="" x2="" ...></axis>
```

很可惜，没有这种元素。但是，这种设计是合理的：不可能为每一种图形都配备一个单独的元素，那样 SVG 就会过于庞大。

因此，我们需要用其他元素来组合成坐标轴，最终使其变为类似以下的形式：

```
<g>
<!-- 第一个刻度 -->
<g>
<line></line> <!-- 第一个刻度的直线 -->
<text></text> <!-- 第一个刻度的文字 -->
</g>
<!-- 第二个刻度 -->
<g>
<line></line> <!-- 第二个刻度的直线 -->
<text></text> <!-- 第二个刻度的文字 -->
</g>
...
<!-- 坐标轴的轴线 -->
<path></path>
</g>
```

分组元素，是 SVG 画布中的元素，意思是 group。此元素是将其他元素进行组合的容器，在这里是用于将坐标轴的其他元素分组存放。

如果需要手动添加这些元素就太麻烦了，为此，D3 提供了一个组件：`d3.svg.axis()`。它为我们完成了以上工作。

## 定义坐标轴

---

上一章提到了比例尺的概念，要生成坐标轴，需要用到比例尺，它们二者经常是一起使用的。下面，在上一章的数据和比例尺的基础上，添加一个坐标轴的组件。

```
//数据
var dataset = [2.5 , 2.1 , 1.7 , 1.3 , 0.9];
//定义比例尺
var linear = d3.scale.linear()
 .domain([0, d3.max(dataset)])
 .range([0, 250]);

var axis = d3.svg.axis()
 .scale(linear) //指定比例尺
 .orient("bottom") //指定刻度的方向
 .ticks(7); //指定刻度的数量
```

第 1 - 2 行：定义数组。

第 4 - 7 行：定义比例尺，其中使用了数组 dataset。

第 9 - 12 行：定义坐标轴，其中使用了线性比例尺 linear。其中：

- d3.svg.axis(): D3 中坐标轴的组件，能够在 SVG 中生成组成坐标轴的元素。
- scale(): 指定比例尺。
- orient(): 指定刻度的朝向，bottom 表示在坐标轴的下方显示。
- ticks(): 指定刻度的数量。

## 在 SVG 中添加坐标轴

---

定义了坐标轴之后，只需要在 SVG 中添加一个分组元素，再将坐标轴的其他元素添加到这个 里即可。代码如下：

```
svg.append("g")
 .call(axis);
```

上面有一个 `call()` 函数，其参数是前面定义的坐标轴 `axis`。

在 D3 中，`call()` 的参数是一个函数。调用之后，将当前的选择集作为参数传递给此函数。也就是说，以下两段代码是相等的。

```
function foo(selection) {
 selection
 .attr("name1", "value1")
 .attr("name2", "value2");
}
foo(d3.selectAll("div"))
```

和

```
d3.selectAll("div").call(foo);
```

因此，

```
svg.append("g").call(axis);
```

与

```
axis(svg.append(g));
```

## 设定坐标轴的样式和位置

---

默认的坐标轴样式不太美观，下面提供一个常见的样式：

```
<style>
.axis path,
.axis line{
 fill: none;
 stroke: black;
 shape-rendering: crispEdges;
}

.axis text {
 font-family: sans-serif;
 font-size: 11px;
}
</style>
```

分别定义了类 axis 下的 path、line、text 元素的样式。接下来，只需要将坐标轴的类设定为 axis 即可。

坐标轴的位置，可以通过 transform 属性来设定。

通常在添加元素的时候就一并设定，写成如下形式：

```
svg.append("g")
 .attr("class","axis")
 .attr("transform","translate(20,130)")
 .call(axis);
```

## 源代码

---

下载地址: [rm50.zip](#)

展示地址: <http://www.ourd3js.com/demo/rm/R-5.0/Axis.html>



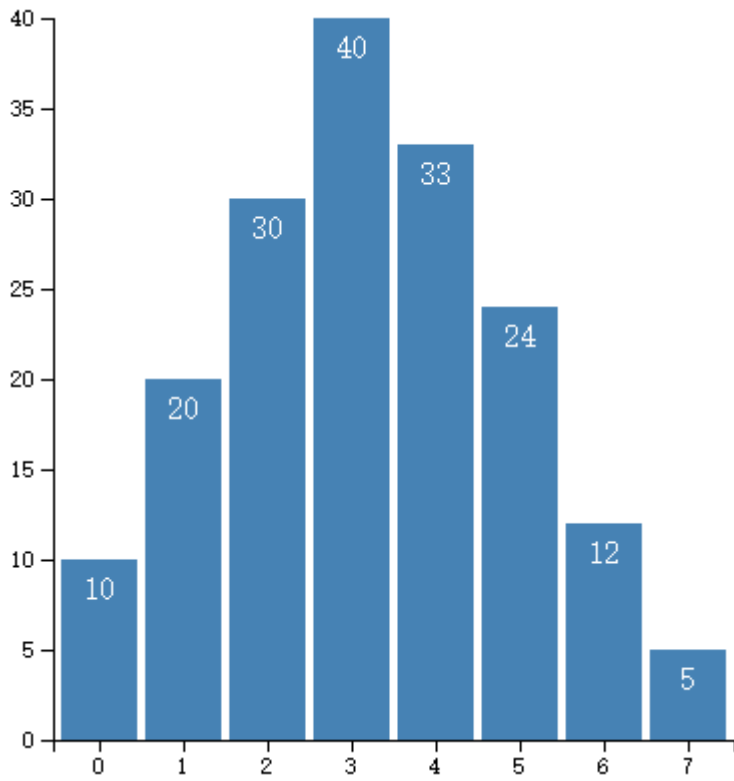


9

## 第八章 完整的柱形图



一个完整的柱形图包含三部分：矩形、文字、坐标轴。本章将对前几章的内容进行综合的运用，制作一个实用的柱形图，内容包括：选择集、数据绑定、比例尺、坐标轴等内容。



图片 9.1 柱形图

## 添加 SVG 画布

---

```
//画布大小
var width = 400;
var height = 400;

//在 body 里添加一个 SVG 画布
var svg = d3.select("body")
 .append("svg")
 .attr("width", width)
 .attr("height", height);

//画布周边的空白
var padding = {left:30, right:30, top:20, bottom:20};
````
```

上面定义了一个 padding，是为了给 SVG 的周边留一个空白，最好不要将图形绘制到边界上。

定义数据和比例尺

```
``javascript
//定义一个数组
var dataset = [10, 20, 30, 40, 33, 24, 12, 5];

//x轴的比例尺
var xScale = d3.scale.ordinal()
  .domain(d3.range(dataset.length))
  .rangeRoundBands([0, width - padding.left - padding.right]);

//y轴的比例尺
var yScale = d3.scale.linear()
  .domain([0, d3.max(dataset)])
  .range([height - padding.top - padding.bottom, 0]);
```

x 轴使用序数比例尺，y 轴使用线性比例尺。要注意两个比例尺值域的范围。

定义坐标轴

```
//定义x轴
var xAxis = d3.svg.axis()
  .scale(xScale)
  .orient("bottom");

//定义y轴
var yAxis = d3.svg.axis()
  .scale(yScale)
  .orient("left");
```

x 轴刻度的方向向下，y 轴的向左。

添加矩形和文字元素

```
//矩形之间的空白
var rectPadding = 4;

//添加矩形元素
var rects = svg.selectAll(".MyRect")
    .data(dataset)
    .enter()
    .append("rect")
    .attr("class", "MyRect")
    .attr("transform", "translate(" + padding.left + "," + padding.top + ")")
    .attr("x", function(d,i){
        return xScale(i) + rectPadding/2;
    })
    .attr("y", function(d){
        return yScale(d);
    })
    .attr("width", xScale.rangeBand() - rectPadding )
    .attr("height", function(d){
        return height - padding.top - padding.bottom - yScale(d);
    });

//添加文字元素
var texts = svg.selectAll(".MyText")
    .data(dataset)
    .enter()
    .append("text")
    .attr("class", "MyText")
    .attr("transform", "translate(" + padding.left + "," + padding.top + ")")
    .attr("x", function(d,i){
        return xScale(i) + rectPadding/2;
    })
    .attr("y", function(d){
        return yScale(d);
    })
    .attr("dx", function(){
        return (xScale.rangeBand() - rectPadding)/2;
    })
    .attr("dy", function(d){
        return 20;
    })
    .text(function(d){
        return d;
    });
```

矩形元素和文字元素的 x 和 y 坐标要特别注意，要结合比例尺给予适当的值。

添加坐标轴的元素

```
//添加x轴
svg.append("g")
  .attr("class","axis")
  .attr("transform","translate(" + padding.left + "," + (height - padding.bottom) + ")")
  .call(xAxis);

//添加y轴
svg.append("g")
  .attr("class","axis")
  .attr("transform","translate(" + padding.left + "," + padding.top + ")")
  .call(yAxis);
```

坐标轴的位置要结合空白 padding 的值来设定。

源代码

下载地址: [rm51.zip](#)

展示地址: <http://www.ourd3js.com/demo/rm/R-5.1/Chart.html>



10

第九章 让图表动起来



D3 支持制作动态的图表。有时候，图表的变化需要缓慢的发生，以便于让用户看清楚变化的过程，也能给用户不小的友好感。

什么是动态效果

前面几章制作的图表是一蹴而就地出现，然后绘制完成后不再发生变化的，这是静态的图表。

动态的图表，是指图表在某一段时间会发生某种变化，可能是形状、颜色、位置等，而且用户是可以看到变化的过程的。

例如，有一个圆，圆心为 (100, 100)。现在我们希望圆的 x 坐标从 100 移到 300，并且移动过程在** 2 秒**的时间内发生。

这种时候就需要用到动态效果，在 D3 里我们称之为过渡（transition）。

实现动态的方法

D3 提供了 4 个方法用于实现图形的过渡：从状态 A **变为状态 B**。

transition()

启动过渡效果。

其前后是图形变化前后的状态（形状、位置、颜色等等），例如：

```
.attr("fill","red")    //初始颜色为红色
.transition()          //启动过渡
.attr("fill","steelblue") //终止颜色为铁蓝色
```

D3 会自动对两种颜色（红色和铁蓝色）之间的颜色值（RGB值）进行插值计算，得到过渡用的颜色值。我们无需知道中间是怎么计算的，只需要享受结果即可。

duration()

指定过渡的持续时间，单位为毫秒。

如 duration(2000)，指持续 2000 毫秒，即 2 秒。

ease()

指定过渡的方式，常用的有：

- linear：普通的线性变化
- circle：慢慢地到达变换的最终状态
- elastic：带有弹跳的到达最终状态
- bounce：在最终状态处弹跳几次

调用时，格式形如：ease(“bounce”)。

delay()

指定延迟的时间，表示一定时间后才开始转变，单位同样为毫秒。此函数可以对整体指定延迟，也可以对个别指定延迟。

例如，对整体指定时：

```
.transition()  
.duration(1000)  
.delay(500)
```

如此，**图形整体**在延迟 500 毫秒后发生变化，变化的时长为 1000 毫秒。因此，过渡的总时长为1500毫秒。

又如，对一个一个的图形（图形上绑定了数据）进行指定时：

```
.transition()  
.duration(1000)  
.delay(function(d,i){  
    return 200*i;  
})
```

如此，假设有 10 个元素，那么第 1 个元素延迟 0 毫秒（因为 $i = 0$ ），第 2 个元素延迟 200 毫秒，第 3 个延迟 400 毫秒，依次类推…。整个过渡的长度为 $200 * 9 + 1000 = 2800$ 毫秒。

实现简单的动态效果

下面将在 SVG 画布里添加三个圆，圆出现之后，立即启动过渡效果。

第一个圆，要求移动 x 坐标。

```
var circle1 = svg.append("circle")
    .attr("cx", 100)
    .attr("cy", 100)
    .attr("r", 45)
    .style("fill", "green");

//在1秒（1000毫秒）内将圆心坐标由100变为300
circle1.transition()
    .duration(1000)
    .attr("cx", 300);
```

第二个圆，要求既移动 x 坐标，又改变颜色。

```
var circle2 = svg.append("circle")... //与第一个圆一样，省略部分代码

//在1.5秒（1500毫秒）内将圆心坐标由100变为300，
//将颜色从绿色变为红色
circle2.transition()
    .duration(1500)
    .attr("cx", 300)
    .style("fill", "red");
```

第三个圆，要求既移动 x 坐标，又改变颜色，还改变半径。

```
var circle3 = svg.append("circle")... //与第一个圆一样，省略部分代码

//在2秒（2000毫秒）内将圆心坐标由100变为300
//将颜色从绿色变为红色
//将半径从45变成25
//过渡方式采用bounce（在终点处弹跳几次）
circle3.transition()
    .duration(2000)
    .ease("bounce")
    .attr("cx", 300)
    .style("fill", "red")
    .attr("r", 25);
```

结果请查看以下链接：

展示地址：<http://www.ourd3js.com/demo/rm/R-6.0/SimpleTransition.html>

给柱形图加上动态效果

在上一章完整柱形图的基础上稍作修改，即可做成一个带动态效果的、有意思的柱形图。

在添加文字元素和矩形元素的时候，启动过渡效果，让各柱形和文字缓慢升至目标高度，并且在目标处跳动几次。

对于文字元素，代码如下：

```
.attr("y",function(d){
    var min = yScale.domain()[0];
    return yScale(min);
})
.transition()
.delay(function(d,i){
    return i * 200;
})
.duration(2000)
.ease("bounce")
.attr("y",function(d){
    return yScale(d);
});
```

文字元素的过渡前后，发生变化的是 y 坐标。其起始状态是在 y 轴等于 0 的位置（但要注意，不能在起始状态直接返回 0，要应用比例尺计算画布中的位置）。终止状态是目标值。

对于矩形元素，思想与文字元素一样，只是在计算起始状态时要稍微复杂一些，请读者自行研读源代码。

展示地址：<http://www.ourd3js.com/demo/rm/R-6.0/ChartTransition.html>

源代码

下载地址: [rm60.zip](#)



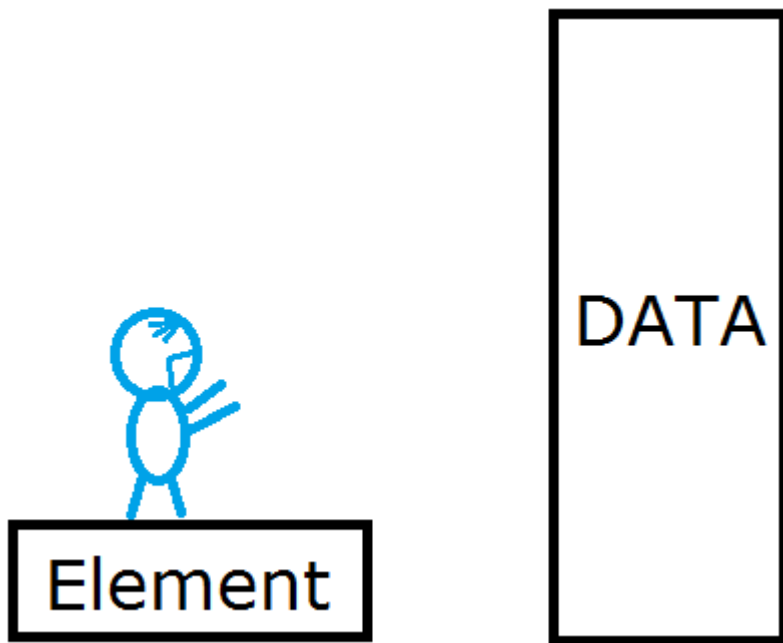
11



第十章 理解 Update、Enter、Exit



Update、Enter、Exit 是 D3 中三个非常重要的概念，它处理的是当选择集和数据的数量关系不确定的情况。



图片 11.1 选择集与数据

什么是 Update、Enter、Exit

前几章里，反复出现了形如以下的代码。

```
svg.selectAll("rect") //选择svg内所有的矩形
  .data(dataset)       //绑定数组
  .enter()             //指定选择集的enter部分
  .append("rect")      //添加足够数量的矩形元素
```

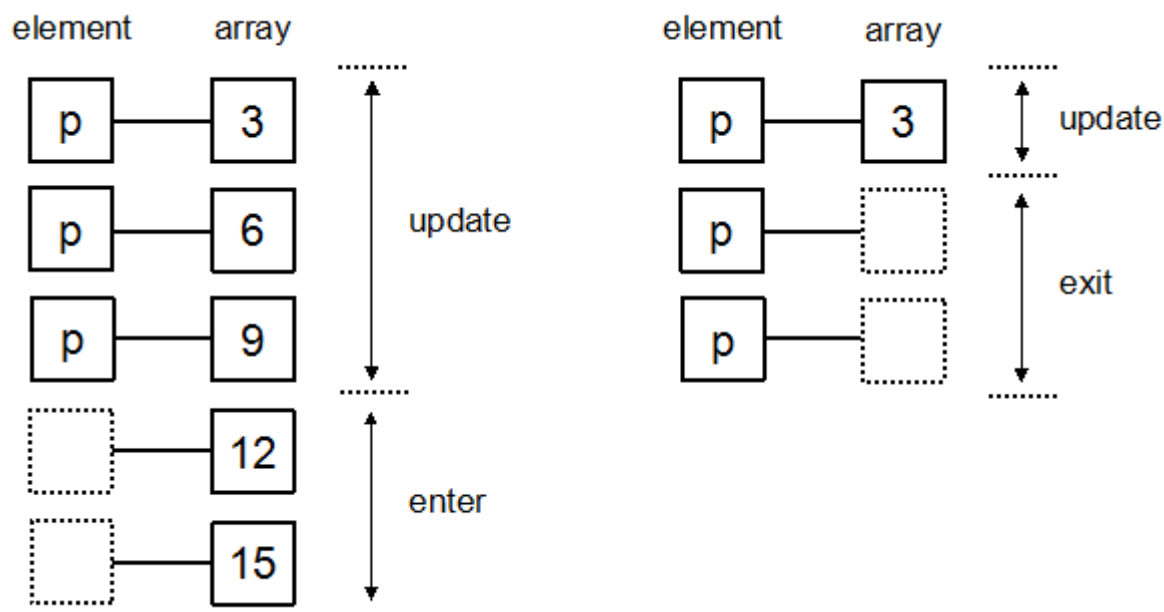
前面提到，这段代码使用的情况是当以下情况出现的时候：

有数据，而没有足够图形元素的时候，使用此方法可以添加足够的元素。

当时并没有深究这段代码是什么意思，本章将对此进行讲解。但是，由于此问题相对复杂，本章只进行最初步的介绍。

假设，在 body 中有三个 p 元素，有一数组 [3, 6, 9]，则可以将数组中的每一项分别与一个 p 元素绑定在一起。但是，有一个问题：当数组的长度与元素数量不一致（数组长度 > 元素数量 or 数组长度 < 元素数量）时呢？这时候就需要理解 Update、Enter、Exit 的概念。

如果数组为 [3, 6, 9, 12, 15]，将此数组绑定到三个 p 元素的选择集上。可以想象，会有三个数据没有元素与之对应，这时候 D3 会建立两个空的元素与数据对应，这一部分就称为 Enter。而有元素与数据对应的部分称为 Update。如果数组为 [3]，则会有两个元素没有数据绑定，那么没有数据绑定的部分被称为 Exit。示意图如下所示。



图片 11.2 update,enter,exit

看到这，我想大家能体会到为什么本节最开始处的代码能够给 SVG 内添加足够数量的元素了吧。它的意思其实是：

此时 SVG 里没有 rect 元素，即元素数量为 0。有一数组 dataset，将数组与元素数量为 0 的选择集绑定后，选择其 Enter 部分（请仔细看上图），然后添加（append）元素，也就是添加足够的元素，使得每一个数据都有元素与之对应。

Update 和 Enter 的使用

当对应的元素不足时（ 绑定数据数量 > 对应元素 ），需要添加元素（ append ）。

现在 body 中有三个 p 元素，要绑定一个长度大于 3 的数组到 p 的选择集上，然后分别处理 update 和 enter 两部分。

```
var dataset = [ 3 , 6 , 9 , 12 , 15 ];

//选择body中的p元素
var p = d3.select("body").selectAll("p");

//获取update部分
var update = p.data(dataset);

//获取enter部分
var enter = update.enter();

//update部分的处理：更新属性值
update.text(function(d){
    return "update " + d;
});

//enter部分的处理：添加元素后赋予属性值
enter.append("p")
    .text(function(d){
        return "enter " + d;
    });
```

结果如下图，update 部分和 enter 部分被绑定的数据很清晰地表示了出来。



```
update 3
update 6
update 9
enter 12
enter 15
```

图片 11.3 update和enter

请大家记住：

- update 部分的处理办法一般是：更新属性值
- enter 部分的处理办法一般是：添加元素后，赋予属性值

Update 和 Exit 的使用

当对应的元素过多时（ 绑定数据数量 < 对应元素 ），需要删掉多余的元素。

现在 body 中有三个 p 元素，要绑定一个长度小于 3 的数组到 p 的选择集上，然后分别处理 update 和 exit 两部分。

```
var dataset = [ 3 ];

//选择body中的p元素
var p = d3.select("body").selectAll("p");

//获取update部分
var update = p.data(dataset);

//获取exit部分
var exit = update.exit();

//update部分的处理：更新属性值
update.text(function(d){
    return "update " + d;
});

//exit部分的处理：修改p元素的属性
exit.text(function(d){
    return "exit";
});

//exit部分的处理通常是删除元素
// exit.remove();
```

结果如下，请大家区分好 update 部分和 exit 部分。这里为了表明哪一部分是 exit，并没有删除掉多余的元素，但实际上 exit 部分的绝大部分操作是删除。



update 3

exit

exit

图片 11.4 update和exit

请大家记住：

- exit 部分的处理办法一般是：删除元素（ remove ）

源代码

下载地址: [rm70.zip](#)

展示地址: <http://www.ourd3js.com/demo/rm/R-7.0/enter.html>

展示地址: <http://www.ourd3js.com/demo/rm/R-7.0/exit.html>

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台



更多信息请访问 

<http://wiki.jikexueyuan.com/project/d3wiki/>