



[PACKT]
PUBLISHING

给出常见问题的快速解答及多达70余种创建动态数据可视化的方案

D3.js 数据可视化实战手册

Data Visualization with D3.js Cookbook

[加] Nick Qi Zhu 著
杨锐 刘夏 王超 张沙沙 译

 人民邮电出版社
POSTS & TELECOM PRESS

版权声明

Copyright © Packt Publishing 2013. First published in the English language under the title *Data Visualization with D3.js Cookbook*.

All rights reserved.

本书中文简体字版由 **Packt Publishing** 公司授权人民邮电出版社出版。未经出版者书面许可，对本书的任何部分不得以任何方式或任何手段复制和传播。

版权所有，侵权必究。

-
- ◆ 著 [加拿大] Nick Qi Zhu
译 杨 锐 刘 夏 王 超 张沙沙
责任编辑 王峰松
责任印制 彭志环
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京 印刷
 - ◆ 开本：800×1000 1/16
印张：20
字数：384 千字 2014 年 8 月第 1 版
印数：1- 000 册 2014 年 8 月北京第 1 次印刷
著作权合同登记号 图字：01-2013-9214 号
-

定价： 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京崇工商广字第 0021 号

D3.js 数据可视化实战手册

[加拿大] Nick Qi Zhu 著

杨 锐 刘 夏 王 超 张沙沙 译

人 民 邮 电 出 版 社

北 京

内容提要

如今这个互联网时代，人们每天都生产海量的数据，如果直接面对这些数据，可能让人无从下手。将数据可视化，用形象立体的形式将其展现，有利于分析其中的关联，攫取可能存在的商业机会。本书意图通过大量的示例和代码，向读者讲述如何利用 D3.js 来实现数据可视化。只要读者了解 JavaScript，就能完全掌握本书的内容。

本书一共 13 章，从如何搭建 D3.js 的开发环境开始，逐步介绍 D3 中的各种操作，包括选集、数据的初步处理、数据映射、坐标轴组件、动画过渡效果、SVG 相关介绍、绘制图表、安排布局、可视化交互、力学模拟、制作地图和测试驱动。本书包含大量的示例和代码，可以帮助读者充分理解书中讲述的每一个概念。本书最后的附录部分，介绍了另外两个 JavaScript 库，主要是关于三维制图和多维图表的。希望本书的内容能对读者理解和学习数据可视化有所帮助。

推荐序

ThoughtWorks 公司从 2010 年开始，每年都会发布至少一份引领行业技术潮流和走向的“技术雷达”，在这样的“技术雷达”中，或平台，或工具，或语言，被技术专家们贴上不同的标签——“采用”、“试用”、“尝试”和“停用”，以表明 ThoughtWorks 根据自己的技术经验和见解，向整个 IT 行业从业者推介某个条目的程度。如果你是“技术雷达”的长期关注者，你会发现这个舞台上真是瞬息万变，伴随着行业领域的变化和技术趋势的新走向，“技术雷达”也在不断地推陈出新。“你方唱罢我登场”，“技术雷达”上鲜有长期的优胜者，但本书的主角 D3.js 却是个另类。

D3.js 第一次出现在“技术雷达”，是在 2012 年 10 月的工具篇中，我们推荐人们“试用”它。而在接下来连着两年的“技术雷达”中，我们都推荐人们在自己的工作和项目中直接“采用”D3.js。D3.js 作为遵守 Web 标准的 JavaScript 库，凭借其轻量级和易于学习的特性，已经成为新兴的数据可视化技术领域中的重要力量，甚至有人将其比作通用 Web 开发中的 jQuery。

数据可视化是一个逐渐受到重视的领域。人们每天都在生产海量的数据，但海量的数据中依旧蕴含着无限的可能，越是大规模的数据，越是存在可以被挖掘的商业机会，我们可以从中分析出公司的库存走向、用户的行为习惯，甚至新兴行业机会的萌动。这些被分析后得出的数据，成为我们改进和拓展新领域的数据支撑。而数据可视化，会让这个过程更加明晰和具备可说服力。

D3.js 作为数据可视化技术领域的佼佼者，相对于其他工具，拥有庞大的用户基数、更加友好开放的社区以及大量的学习资料。我们还注意到，越来越多的非 IT 从业者，比如业务人员，甚至是记者，也在借助 D3.js 的强大能力。

很荣幸的是，D3.js 以及本书的原作者 Nick Qi Zhu，还有中文版的译者杨锐、刘夏、王超、张沙沙，都是我目前在 ThoughtWorks 的同事。他们在本书中为读者呈现了大量的示例，以帮助读者最快速地上手 D3.js。

谢谢我的同事们。

张凯峰

高级咨询师

ThoughtWorks 中国

前言

D3.js 是一个 JavaScript 库，它主要用于对数据的动态图表展示。通过 HTML、SVG 以及 CSS，D3 可以让数据展现得更加鲜活。D3 使得数字的图形化展示变得异常简单，可以说，它是当下最强大的基于网络的数据可视化技术。

本书理论与实践结合，力图向读者全方位地展示 D3 数字可视化技术，帮助读者快速利用 D3 创建可视化程序。学习完本书后，快速高效地创建叹为观止的数据可视化程序，对读者来说将是小菜一碟！

本书由浅入深，首先介绍了一些 D3 数字可视化编程中的基本概念，继而通过一些代码样例，对 D3 的其他特性做逐一展示。

在这里，读者将会学习到数据可视化的基本概念，JavaScript 的函数式编程，以及 D3 的基础概念，例如元素选取、数据绑定、动画以及 SVG 生成。除此之外，读者还会领略到 D3 的一些高级特性，例如自定义插值、自定义中间帧、定时器、布局管理、力，等等。本书还提供了许多预生成的图表和代码，帮助读者更快起步。

本书包括

第 1 章，D3 入门指南，是 D3.js 预热。它涵盖了一些基本概念，诸如 D3.js 是什么，以及如何搭建一个适用于 D3.js 数据可视化程序的开发环境等。

第 2 章，精挑细选，向你介绍了 D3 数字可视化中最基本的一项操作——选集。选集可以帮助读者定位页面上的元素。

第 3 章，与数据同行，探索了任何数据可视化程序中都会涉及的基础问题——如何通过程序构造、可视化效果展示数据。

第 4 章，张弛有“度”，介绍了数字可视化中非常重要的一个子领域。作为一个数字可视化的开发人员，如何将数据映射为可视化元素，是一个每天都要面对的问题，本章就此问题做了深入探索。

第 5 章，玩转坐标轴，介绍了坐标轴组件，以及基于笛卡尔坐标系的可视化程序的相关技术。

第 6 章，优雅变换，介绍了过渡相关的概念。“一图胜千言”正是对数字可视化的最好总结。这一章涵盖了 D3 库中过渡以及动画的相关概念。

第 7 章，图形之美，介绍了 SVG 相关的概念。SVG 是一个广泛用于数字可视化程序的 W3C（World Wide Web Consortium）标准。

第 8 章，图表美化，探索了数据可视化中最为人知的组件——图表。图表是定义良好的且易于理解的数据可视化展示方式。

第 9 章，井然有序，本章集中讲述 D3 的布局。D3 的布局是一种算法，用于计算和生成元素的位置信息，这些元素可用于生成复杂又有趣的可视化程序。

第 10 章，可视化交互，本章集中讲述 D3 对可视化交互的支持，换句话说，即如何向你的可视化程序添加控制能力。

第 11 章，使用“原力”，介绍了 D3 中又一神奇的功能——力学。力模拟是数字可视化程序中最炫的一项技术。

第 12 章，地图的奥秘，介绍了 D3 基本的地图可视化技术，以及如何在 D3 中实现一个全功能的可视化地图。

第 13 章，测试驱动，帮助你使用 TDD 的方式来实现可视化程序。

附录，Building Interactive Analytics in Minutes，介绍了三维制图 Crossfilter.js 和 dc.js 技术。

准备工作

- ◆ 文本编辑器：编辑、创建 HTML、CSS、JavaScript 文件。
- ◆ 浏览器：浏览器（火狐 3、IE9、Chrome、Safari 3.2 及以上版本浏览器）。
- ◆ 一个本地的 HTTP 服务器：本书中的一些章节需要 HTTP 服务器来存储数据文件。在第 1 章，我们将对如何搭建一个简单的 Node 或者 Python 的 HTTP 服务器做出介绍。

◆ Git 客户端（可选）：如果你想从 Git 目录迁入本书的源码，你需要在你的机器上安装一个 Git 客户端。

本书面向对象

如果你是一个开发人员，或者是一个 HTML、CSS、JavaScript 的狂热爱好者，你希望了解 D3 的大部分知识，那么这本书非常适合你。这本书还可以作为资深的 D3 数字可视化程序开发人员的快速参考指南。

读者反馈

我们非常期待读者的反馈。让我们知道你对此书的观点——你喜欢什么，不喜欢什么，这很重要。读者反馈能够帮助我们调整此书内容，更好地为读者介绍所需要的知识。

你可以通过向 feedback@packtpub.com 发送邮件来表达自己的观点，请在主题栏中注明本书的标题。

如果书中包含了你非常擅长的话题，或者你有兴趣编写的部分，请在 www.packtpub.com/authors 查看我们的指南。

客户支持

作为 Packt 书籍的拥有者，读者享有如下权利。

下载样例代码

你可以使用你的账户在 <http://www.packtpub.com> 下载所购买书籍中的样例代码。如果是在其他地方购买的此书，请通过 <http://www.packtpub.com/support> 注册获取源代码。

勘误表

虽然我们尽全力保证书中内容的准确性，但是错误还是在所难免。如果读者在本书中找到错误，可能是文章的错误，也可能是代码的错误，请告知我们。这样，其他读者就不会被错误内容困扰，更好地帮助我们在后续的发行中提升本书质量。如果你发现书中的勘

误，请通过 <http://www.packtpub.com/submit-errata> 来报告，在页面选取你的书籍，然后单击 **errata submission form** 链接，并键入错误的详细内容。读者指出的勘误一旦被验证后，就会被上传到我们的网站，或者添加到已存勘误的列表中。读者可以通过 <http://www.packtpub.com/support> 进入相关书目标题，查看已知勘误。

版权

版权材料在互联网中的盗版现象几乎是所有媒体中存在的通病。对于 Packt，我们严格保护版权和许可证。如果读者在互联网中，无论以任何形式见到我们作品的复制品，请向我们提供地址或者网站名称。

对于涉嫌盗版的材料，请将链接发送至 copyright@packtpub.com。

非常感谢你对我们的作者以及本书内容的厚爱！

问题

任何关于本书的疑问，请通过 questions@packtpub.com 联系我们，我们将尽全力帮助你解惑。

审核人员简介

Andrew Berls 是一名 Ruby 和 JavaScript 的开发人员，居住在加拿大的圣芭芭拉。他从知晓 HTML 标签开始，就爱上了构建网站，现在是一名全栈工程师，网站前台、后端无所不知。他目前在 Causes.com 公司实习，使用 D3.js 为虚拟社交网络搭建数据看板。Andrew 即将完成加利福尼亚大学圣芭芭拉分校计算机科学专业的学位。除了编程，他还酷爱烹饪（虽然做得不好）和远足。

Kevin Coughlin 拥有着新泽西大学的计算机科学和经济学的双学位。他拥有两年的 JavaScript 开发经验。Kevin 结合 HTML5 标准，使用尖端的客户端和服务端技术，诸如 Angular.js、Backbone.js 以及 Node.js 等，为开放网络提供了很多高效的现代化解决方案。

Kevin 还是一名博客活跃分子，他会定期在他的博客上更新新兴技术，有兴趣请查阅 <http://kevintcoughlin.com>。

Ismeni Lourentzou 拥有工商管理学学士学位，长期任职于希腊国家银行。对 Java 的极大热忱和对新知识的渴望，促使她又在雅典经济与商务大学（AUEB, Athens University of Economics and Business）开始了她的计算机科学学位之路。本科阶段，其作为由 Michalis Vazirgiannis 教授领衔的 AUEB 数据和网络挖掘组的成员之一，参与了 2012 年知识探索与数据挖掘大赛，研究题目为“在线广告代码自动生成 (Automated Snippet Generation of Online Advertising)”，并在 2013 年信息知识管理大会（CIKM）上得以发表。同时她也是 AUEB 信息恢复组的成员，该小组由 Theodore Kalamboukis 教授领队，参与了 ImageClef 2013 大赛。他们的参赛作品在基于图像的文字检索大赛中荣获第 2，在基于图像的视觉检索大赛中获得第 5。她对研究和编程的热爱，促成了她的职业转型。目前她是伊利诺伊大学 Urbana - Champaign 分校（University of Illinois at Urbana-Champaign）的一名博士生，研究方向为智能信息系统中的机器学习与信息搜索，该技术主要针对文本信息挖掘，以减少用户对文字信息的手动搜索、组织、信息理解的量，来改善用户体验。她希望在完成博士学业后，

可以继续研究，每天学习更多的知识。

我要特别感谢我的家人对我的支持和帮助，他们总是激励我，我的妈妈更是在我非常繁忙的时候来照顾我，我的姐姐总是能够理解我，我的父亲亦会在困难的时候支持我。还有，我的男朋友，他用他永恒的耐心 and 爱来陪伴我，我的朋友也在这个过程中为我提供了很多意见。

Pablo Navarro 是来自智利的数据可视化咨询师。他在法国巴黎高等矿业德·圣-艾蒂安大学（École des Mines de Saint-Etienne, France）获得应用数学硕士学位。在搜索以及数据分析方面工作数年后，他决定专攻网络平台的数据展示，这也正是他目前工作的主要领域。在业余时间，他喜欢水彩插画、跑步以及阅读人类进化论方面的东西。如果你对他的工作有兴趣，请关注 <http://pnavarrc.github.io>。

第 1 章

D3.js 入门指南

本章涵盖以下内容：

- ◆ 搭建简易的 D3 开发环境
- ◆ 搭建基于 NPM（Node Packaged Modules 是 Node.js 的套件管理工具）的开发环境
- ◆ 理解 D3 风格的 JavaScript

1.1 简介

本章旨在帮助读者初步认识并且运行 D3.js。其中包含一些基本知识，比如什么是 D3.js，如何搭建一个典型的 D3.js 数据可视化（data visualization）环境。还有一个专门的章节，解释了一些 JavaScript 中鲜为人知而 D3.js 又甚为倚重的特性。

什么是 D3？D3 是指数据驱动文档（Data-Driven Documents），根据 D3 的官方定义：

D3.js 是一个 JavaScript 库，它可以通过数据来操作文档。D3 可以通过使用 HTML、SVG 和 CSS 把数据鲜活形象地展现出来。D3 严格遵循 Web 标准，因而可以让你的程序轻松兼容现代主流浏览器并避免对特定框架的依赖。同时，它提供了强大的可视化组件，可以让使用者以数据驱动的方式去操作 DOM。

D3 维基（2013 年 8 月）

总的来说，D3 是这样一个特殊的 JavaScript 库，它利用现有的 Web 标准，通过更简单的（数据驱动）方式来制作炫目的可视化效果。D3.js 由 Mike Bostock（<http://bost.ocks.org/mike/>）制作。之前他制作过一个叫 Protovis 的数据可视化 JavaScript 库，如今它已

经被 D3.js 取代。如果了解更多诸如 D3.js 制作过程、影响 Protovis 和 D3.js 的相关理论这类的信息，可以看看下面的链接。而本书将着眼于如何使用 D3.js 来增强可视化。D3 使用 JavaScript 实现数据可视化的方式比较特别，因此刚开始时可能会让人觉得有些难懂。我希望通过本书中的大量实例，其中有基础的，也有高级的话题，能够帮助大家更好更高效地使用 D3。一旦理解了原理，使用 D3 就可以让数据可视化的效率和丰富程度产生指数化的增长。

更多有关制作 D3 的创意，可以参考 Mike Bostock 于

2010 年在 IEEE InfoVis 发表的论文 Declarative Language

Design for Interactive Visualization ,链接 :<http://vis.stanford.edu/papers/protovis-design>。



如果对于 D3 是如何制作的感兴趣，建议看看 Mike Bostock 于 2011 年在 IEEE InfoVis 发表的论文 D3:

Data-Driven Document ,链接 :<http://vis.stanford.edu/papers/d3>。

Protovis ,D3.js 的前辈，是 Mike Bostock 和斯坦福可视

化组的 Jeff Heer 制做的，相关链接：[http://](http://mbostock.github.io/protovis/)

mbostock.github.io/protovis/。

1.2 搭建一个简易的 D3 开发环境

在开始使用 D3 之前，我们要做的第一件事是搭建一个开发环境。这节里，我们将告诉你如何在几分钟内搭建一个简单的 D3 开发环境。

1.2.1 准备阶段

在我们开始前，请确保你已经安装好一个文本编辑器。

1.2.2 搭建环境

我们先要下载 D3.js。

1. 我们可以在 <http://d3js.org/> 下载最新版本的 D3.js，也可以在 <https://github.com/mbostock/d3/tags> 下载之前的版本。另外，如果你对开发中的最新 D3 版本感兴趣，可以 fork 以下的代码库 <https://github.com/mbostock/d3>。

2. 下载并且解压缩后，我们得到了 3 个文件：d3.v3.js、d3.v3.min.js 和许可证文件。在开发过程中，建议使用 d3.v3.js，它可以帮你深入到 D3 库中跟踪调试 JavaScript 代码。把 d3.v3.js 和新建的 index.html 放在同一个文件夹里，index.html 应该包含下面的代码。


```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Simple D3 Dev Env</title>
  <script type="text/javascript" src="d3.v3.js"></script>
</head>
<body>

</body>
</html>
```



如果是直接下载源代码或者 tagged 版本，JavaScript 文件的名称会略微不同，叫做 d3.js，而非 d3.v3.js。


我们已经搭建了一个最简单的 D3 数据可视化开发环境。现在可以用我们最喜欢的文本编辑器编辑那个 html 文件，还可以用浏览器打开它来检查可视化的效果。

 可以在这里下载这个例子的源码：<https://github.com/NickQiZhu/d3-cookbook/tree/master/src/chapter1/simple-dev-env>。

1.2.3 工作原理

D3 是个相当独立的程序库。它不依赖于特定浏览器提供的功能以及其他 JavaScript 库。实际上，你甚至可以通过简单的配置，让 D3 脱离浏览器而在诸如 Node.js 这样的环境中运行起来（后面的章节会更详细地介绍这点）。


如果你的浏览器是 IE9，建议使用 Aight 兼容库和 Sizzle

 selector engine。其中前者可以在这里下载：<https://github.com/shawnbot/aight>，而后者可以在这里下载：<http://sizzlejs.com/>。

html 文件的头部信息中必须包含如下的编码格式声明。

```
<meta charset="utf-8">
```

上述声明会告诉浏览器和验证器在渲染页面的时候使用哪个字符集，否则浏览器就不会正确加载 D3 了，因为 D3 使用了一些 utf-8 里面的特殊字符，比如 π 。

 D3 是完全开源的。它使用了它的作者 Michael Bostock 自己定制的开源许可证。该许可证跟流行的 MIT 许可证十分类似，仅有一点不同，就是其明确声明了 Michael Bostock 的名字未经允许不可用来标记此软件的派生

品，或者用以扩大此软件的衍生品的影响。

1.2.4 更多内容

本书提供了大量的代码示例。所有的源码均可在 GitHub 上下载到 (<https://github.com>)。

如何获取源码

最简单的方式就是直接克隆 Git 上的代码库 (<https://github.com.NickQiZhu/d3-cookbook>)。如果你不打算构建开发环境，跳过这步即可。

如果你不熟悉 Git，克隆 (clone) 很类似于其他的版本



控制软件中的签出 (checking out)。不过克隆所做的并

非简单的签出。它把所有分支和历史拷贝到了你的本

地机器，也就是把整个代码库都拷贝到了本地机器中，

所以你完全可以使用这个拷贝过来的代码库，在本地

环境中离线工作。

如果是首次安装 Git，可以在这里找到 Git 客户端下载列表：<http://git-scm.com/downloads>，在这里还有针对不同操作系统的安装向导：<http://git-scm.com/book/en/Getting-StartedInstalling-Git>。



另一个使用 Git 和 GitHub 的方式是安装 GitHub 客户端，

它提供了比 Git 更多样的功能。在作者编写本书时，GitHub

仅仅提供了 Windows 版 (<http://windows.github.com>)。

com/) 和 Mac OS 版 (<http://mac.github.com/>)。

安装完 Git 客户端后, 运行下面的命令就可以把本书中所有的代码都下载到你的机器里。

```
> git clone git://github.com/NickQiZhu/d3-cookbook.git
```

如果你打算使用 GitHub 客户端 (目前还没有中文版),



在 <https://github.com/NickQiZhu/d3-cookbook>

页面点击 Fork 按钮 (在页面的右上角), 之后你就可以

在 GitHub 客户端中看到这个代码库了。

1.3 搭建一个基于 NPM 的开发环境

如果你所在的项目是一个略复杂的数据展示项目, 并且使用了为数不少的 JavaScript 库, 那我们之前讨论的那个简单的解决方案可能就显得有些褚小杯大, 不能胜任了。在这一节当中, 我们将展示一个使用了 NPM (Node Packaged Modules, 实际上就是 JavaScript 库的代码库管理系统) 的更加强大的系统。如果你像我一样没有耐心, 想更快地尝试本书最带劲儿的部分, 想学点秘传招式, 完全可以跳过这部分, 如果想搭建一个产品开发环境, 再回来看看这部分也不迟。

1.3.1 准备阶段

我们开始前要先确保 NPM 已经安装好了。在安装 Node.js 时, NPM 作为其中一部分也被安装了。可以在 <http://nodejs.org/download/> 下载 Node.js。选择适合你操作系统的 Node.js, 安装完毕后, 在终端窗口运行如下 npm 命令。

```
> npm -v  
1.2.14
```

如果前面的命令输出了 NPM 客户端的版本号, 就说明安装成功了。

1.3.2 搭建环境

安装完 NPM 后，让我们创建一个包描述符文件，以便自动化一些手动安装的过程。

1. 首先，在工程文件夹下，创建名为 `package.json` 的文件，包含下面的代码。

```
{
  "name": "d3-project-template",
  "version": "0.1.0",
  "description": "Ready to go d3 data visualization project template",
  "keywords": [
    "data visualization",
    "d3"
  ],
  "homepage": "<project home page>",
  "author": {
    "name": "<your name>",
    "url": "<your url>"
  },
  "repository": {
    "type": "git",
    "url": "<source repo url>"
  },
  "dependencies": {
    "d3": "3.x"
  },
  "devDependencies": {
    "uglify-js": "2.x"
  }
}
```

2. 定义 `package.json` 文件后，运行下面的命令。

```
> npm install
```

1.3.3 工作原理

`package.json` 文件中绝大部分字段仅仅用于提供信息，比如 `name`、`description`、`homepage`、`author` 和 `repository`。如果你打算把自己的库发布到 NPM 的代码库中，就要用

到 `name` 和 `version` 字段。现在，我们在意的是 `dependencies` 和 `devDependencies` 字段。

- ◆ `dependencies` 字段描述了你的工程所用到的库在运行时的依赖，它们可以使你的工程在浏览器中正常运行。在这个简单的例子中，对于 `d3` 我们只有一个依赖。`d3` 是 `D3` 在 NPM 库中发布时使用的名字。版本号 `3.x` 标明该工程可以兼容任意大版本为 3 的发行版，并且 NPM 应该获取大版本为 3 的最新的稳定发布版本来满足依赖。

D3 是个自满足的库，运行时对外部是零依赖。然而这



并不意味着它不能跟其他 JavaScript 库一起运行。作者

平时也使用一些其他的库以便让自己的工作容易些，

比如 JQuery、Zepto.js、Underscore.js 和 Backbone.js。

- ◆ `devDependencies` 字段描述了库在开发时（编译时）的依赖。就是说，这个字段内罗列出来的库文件仅仅在构建工程时会用到，运行 JavaScript 工程时用不到它们。



关于 NPM 包 JSON 文件更详细的文档，可以参考这里

<https://npmjs.org/doc/json.html>。

执行 `npm install` 可以自动触发 NPM 去下载你工程中所引用的所有依赖，包括递归的下载依赖的依赖。所有的依赖库文件会被下载到 `node_modules` 文件夹中，该文件夹位于工程文件夹中的根目录里。这些完成以后，就可以创建一个 HTML 文件（跟我们之前创建的那个一样），HTML 文件直接从 `node_modules/d3/d3.js` 来引用 `D3` 的 JavaScript 库。

本节中的代码可以在此处下载，里面包含了自动构建脚本。

<https://github.com/NickQiZhu/d3-cookbook/tree/master/src/chapter1/npm-dev-env>

工程中会有一些麻烦的地方，比如手动下载 JavaScript 库以时刻保持它们的版本更新。为了避免这些麻烦，使用 NPM 是个行之有效的方式，可以拯救你于水火之中。当然，一些聪明的读者可能已经发现了，使用这个方法可以把我们的“搭建环境”过程直接提升一个

档次。想象一下，你正在构建一个巨大的可视化工程，其中包含了上千行的 JavaScript 代码，很明显我们这里所描述的简单的搭建方式满足不了这种情形。因为“模块化的 JavaScript 开发”这个话题足够写本书了，所以这里我们就不再讨论这方面的话题了，我们将把注意力放在数据可视化和 D3 上。如果你对这方面感兴趣，可以看看这一节的代码，里面演示了如果实现一个模块化的工程，当然也包含了自动构建脚本。在后面单元测试相关的章节中，我们会针对这个话题多讲一些，演示一下，可以在某些方面增强环境，以便运行自动化单元测试。

1.3.4 更多内容

前面提到过，你可以通过浏览器直接打开 HTML 文件来查看可视化的结果，不过这种方式有一些局限性。但我们需要从其他文件中加载数据（后面的章节中我们就要这么做了，这也很像你平时工作中的情形），由于浏览器内建的安全机制，这种方式就不好用了。为了绕开这个安全限制，强烈建议搭建一个本地的 HTTP 服务器，用该服务器来维护 HTML 页面和数据文件，而非直接从文件系统加载。

搭建本地 HTTP 服务器

由于使用的操作系统不同，HTTP 服务器的软件包不同，搭建 HTTP 服务器的方法也很不同。这里说几种比较流行的搭建方式。

Python 简易 HTTP 服务器

这是我最喜欢的方式。如果你的机器上已经安装了 Python，通常 UNIX/Linux/Mac OS 发行版上都有，直接运行下面的命令即可。

```
> python -m SimpleHTTPServer 8888
```

如果你拥有更新版本的 Python，那么你也可以运行如下命令。

```
> python -m http.server
```

这个 python 小程序可以启动一个 HTTP 服务器，然后你就可以访问该程序所在文件夹中的所有文件。这是目前所有操作系统中运行 HTTP 服务器最简单的方式。



如果你的机器没有安装 python，可以在这里下载

<http://www.python.org/getit/>。现在所有的操

作系统，诸如 Windows、Linux，还有 Mac，都支持它。

Node.js HTTP 服务器

安装 Node.js 之后（前面的小节中所做的搭建开发环境练习，包含了相应的内容），就可以搭建 http-server 模块了。与 Python 简易 HTTP 服务器类似，通过该模块，你可以从任意的文件夹快速启动一个轻量级的 HTTP 服务器。

安装 http-server 模块。

```
> npm install http-server -g
```

命令中的 -g 参数会把 http-server 模块设置为全局的，这样你就可以在命令行里直接使用 http-server 命令。完成此步后，可以通过下面的命令，在任意文件夹内启动一个服务器。

```
> http-server .
```

该命令可以启动一个 Node.js 驱动的 HTTP 服务器，默认端口号是 8080，也可以用 -p 参数指定一个端口号。



如果是在 Linux/UNIX/Mac 的操作系统中运行该命令，

需要用 sudo 模式或者 root 权限来运行。

1.4 理解 D3 风格的 JavaScript

对于那些习惯了过程式或者面向对象式的 JavaScript 风格的人来说，他们会感觉 D3 使用函数式的 JavaScript 编程风格有一些奇怪。本节会涵盖一些 JavaScript 中函数式编程最根本的功能性概念，以便对 D3 有个基本的理解，将来可以用 D3 的风格来编写可视化工程的代码。

1.4.1 准备阶段

在浏览器中打开下面文件的本地副本。

<https://github.com/NickQiZhu/d3-cookbook/blob/master/src/chapter1/functional-js.html>

1.4.2 开始编程

现在让我们尝试更深入一点，了解一下 JavaScript 函数式方面的内容。请看下面的代码段，

```
function SimpleWidget(spec) {
  var instance = {}; // <-- A

  var headline, description; // <-- B

  instance.render = function () {
    var div = d3.select('body').append("div");

    div.append("h3").text(headline); // <-- C

    div.attr("class", "box")
      .attr("style", "color:" + spec.color) // <-- D
      .append("p")
      .text(description); // <-- E

    return instance; // <-- F
  };

  instance.headline = function (h) {
    if (!arguments.length) h; // <-- G
    headline = h;
    return instance; // <-- H
  };
}
```

```
instance.description = function (d) {  
  if (!arguments.length) d;  
  description = d;  
  return instance;  
};  
  
return instance; // <-- I  
}  
  
var widget = SimpleWidget({color: "#6495ed"})  
  .headline("Simple Widget")  
  .description("This is a simple widget demonstrating functional javascript.");  
widget.render();
```

这段代码在页面上生成了下面图片中的示例。



函数式 JavaScript 简单示例

1.4.3 工作原理

尽管非常简单，但是不可否认，这段示例中的代码跟 D3 风格的 JavaScript 非常相似。这不是巧合，在 JavaScript 编程范型中，这叫做函数对象。跟很多有趣的话题一样，这个话题也能写一本书。不过在本节中，我会尝试尽量多讲一些这种特殊范型的东西，好让不理解 D3 语法的读者也能创建这种风格的库文件。正如 D3 的维基页面上所讲的那样，这种函数式编程风格给 D3 带来了很大的便利性。

D3 的函数风格，使得多种组件插件之间的代码重用成为现实。

D3 维基（2013 年 8 月）

函数即对象

JavaScript 中的函数是对象。跟其他的对象一样，函数只是键值对的一个集合。函数对

象跟普通对象的区别就是，函数可以执行，函数还带有两个隐藏的属性，即函数上下文和函数代码。这两个隐藏属性有时候会给你一个大大的“意外惊喜”，如果你有着很深的过程式编程背景，这点可能更明显。不过这也是我们最需要注意的关键点了，要了解一下 D3 使用函数的奇怪方式。

JavaScript 的大部分特性都显得有些不够“面向对象”，不



过在函数对象这方面，JavaScript 跟其他语言相比较应

该更胜一筹。

现在我们心里有了这样的概念，那就再看一遍这段代码。

```
var instance = {}; // <-- A

var headline, description; // <-- B

instance.render = function () {
  var div = d3.select('body').append("div");

  div.append("h3").text(headline); // <-- C

  div.attr("class", "box")
    .attr("style", "color:" + spec.color) // <-- D
    .append("p")
    .text(description); // <-- E

  return instance; // <-- F
};
```

在 A、B 和 C 行，我们可以看到 instance、headline 和 description 都是 SimpleWidget 这个函数对象的内部私有变量。可是 render 函数却是 instance 对象的一个方法，并且被定义为对象字面量。函数本身也是对象，它可以存储在对象/函数、其他变量、数组里，也可以作为函数参数。运行 SimpleWidget 的结果就是 I 行所写的，返回一个 instance 对象。

```
function SimpleWidget(spec) {
  ...
  return instance; // <-- I
}
```

`render` 函数中用到了一些我们还没讲过的 D3 中的函



数，不过我们现在先不管它们，后面的章节中我们会详细讲解的。它们也只是渲染了一些可视化的东西，

跟我们目前的话题没有太多的关系。

下载示例代码



可以从 <http://www.packtpub.com> 下载你购买的所有 Packt 图书的示例代码。如果是在其他地方购买的话，可以在 <http://www.packtpub.com/support> 注

册一下，代码会通过邮件发送给你。

静态变量作用域

好奇的读者可能会问，这个示例中的变量作用域到底是怎样的啊？看上去好奇怪，`render` 函数不仅访问了 `instance`、`headline` 和 `description`，还访问了从 `SimpleWidget` 传进来的 `spec` 变量。这个怪异的变量作用域其实是由一个简单的静态作用域规则来决定的。可以把这个规则想象成这样：当查找一个变量引用时，该变量先被当成是一个本地变量。如果没有找到变量声明（比如 C 行中的 `headline`），就继续在父对象里找（本例中，`SimpleWidget` 函数就是静态的父对象，`headline` 变量的声明在 B 行）。如果还是没有找到，就不断地重复这个过程，递归地去父对象里查找，一直到全局变量的定义那层。如果最后还是没有找到，就针对该变量生成一个引用错误。这样的作用域行为与大多数流行语言（诸如 Java、C#）中的变量处理方式大不相同，可能需要一段时间适应，要是你觉得不习惯的话，也不用担心，练得多了，就习惯了。

对于有 Java 和 C#背景的人，需要再提醒一下，



JavaScript 没有实现块作用域 (block scoping)。我们这
里描述的静态作用域规则，仅仅适用于函数/对象级别，
不适用于块级别。

```
for(var i = 0; i < 10; i++){  
  for(var i = 0; i < 2; i++){  
    console.log(i);  
  }  
}
```

对于上面这段代码，你可能觉得它会打印 20 个数字。



其实在 JavaScript 里，这段代码会陷入到无限循环。因
为 JavaScript 没有实现块级别的作用域，所以里面那层
循环的 i 跟外面那层循环的 i 是同一个变量。于是里面
的循环改变了 i 的值，导致外面的循环永远不会结束。

跟流行的原型编程中的伪类模式相比，这样的模式通常被称作函数模式。函数模式的优点是提供了更好的信息隐藏和封装。因为只能通过静态作用域规则里限定的那些嵌套定义的函数来访问私有变量（我们示例中的 `headline` 和 `description`），所以 `SimpleWidget` 函数返回的对象就更加灵活，也更加健壮。

如果我们用函数式风格创建对象，并且该对象所有的方法都没有用 `this`，那

这个对象就是持久 (durable) 的^①。持久对象就是许多功能行为的集合了。

(Crockfort D. 2008 年)

可变参数函数

在 G 行有些奇怪的东西。

```
instance.headline = function (h) {  
  if (!arguments.length) h; // <-- G  
  headline = h;  
  return instance; // <-- H  
};
```

你可能会问, G 行的这个 `arguments` 从哪里来的? 这个例子中从来没有定义过它。其实这个 `arguments` 是内建的隐藏参数, 并可在函数执行时直接使用。`arguments` 是一个数组, 它包含了所在函数的全部参数。

实际上, `arguments` 本身并不是 JavaScript 的数组对象。



虽然它有 `length` 属性, 并可以用索引下标访问每个元素, 但是它没有 JavaScript 中数组对象的那么多方法(比如 `slice`、`concat`)。如果你想在 `arguments` 上使用 JavaScript 数组对象的方法, 需要这样:

```
var newArgs = Array.prototype.slice.apply  
(arguments);
```

把这个隐藏的参数和 JavaScript 可以在函数声明时省略参数的功能结合起来, 就可以写出 `instance.headline` 这种不需要指定参数个数的函数。在本例中, 我们可以传一个参数 `h`, 也可以不传。因为如果没有传进来参数, `arguments.length` 就返回 0, `headline` 函数就返回 `h`, 如果 `h` 有值, 就变成了一个赋值操作。为了说清楚, 我们看看下面这段代码。

^① 译者注: 如果一个对象不包含外部可见的成员数据, 并且其方法不会使用 `this`, 那么这个对象就是持久的。

```
var widget = SimpleWidget({color: "#6495ed"})
  .headline("Simple Widget"); // 给 headline 赋值
console.log(widget.headline()); // 输出"Simple Widget"
```

这里你可以看到 `headline` 在参数不同的情况下，可以分别作为 `setter` 和 `getter`（赋值操作和取值操作）。

函数级联调用

这个例子另一个有趣的地方是函数的级联调用。这也是 D3 库提供的一个主要的函数调用方式，因为 D3 库中的大多数函数都设计成了这种链式的结构，以便能提供简洁的、上下文连贯的编程接口。如果你理解可变参数函数的概念，就很好理解这个了。可变参数函数（比如 `headline` 函数）能同时作为 `setter` 和 `getter`，当其作为 `setter` 时，返回 `instance` 对象，这就使得你可以在返回的 `instance` 上立即执行另一个函数，此即链式调用。

看下面这段代码。

```
var widget = SimpleWidget({color: "#6495ed"})
  .headline("Simple Widget")
  .description("This is ...")
  .render();
```

这个例子中，`SimpleWidget` 函数返回了 `instance` 对象（如 I 行所示）。那么，`headline` 函数在这里是一个 `setter`，同时也返回一个 `instance` 对象（如 H 行所示）。`description` 函数执行后也返回 `instance` 对象。最后调用了 `render` 函数。

现在我们已经大概了解了 JavaScript 的函数式风格，有了一个可工作的 D3 开发环境，也准备好了使用 D3 提供的丰富功能来一试身手。在开始前，我还想讲几个比较重点的事情，即如何寻找、分享代码以及有困难时如何获取帮助。

1.4.4 更多内容

先让我们看几个有用的东西。

寻找、分享代码

在 D3 众多值得称赞的亮点中，有一个是比其他可视化工具提供了更加丰富的示例和教程，你可以从中汲取灵感。当我创建自己的开源可视化图表项目，还有写作这本书的时候，我在那些资源中获得了大量的灵感。我会在那些最棒的例子里，挑出一些列个清单出

来。这个清单虽然不是百科全书，但却是个不错的入门参考。

- ◆ D3 gallery (<https://github.com/mbostock/d3/wiki/Gallery>), 这里有不少有趣的例子, 可以帮你在线查找 D3 的使用方法, 有各种各样的图表、特定的技术, 还有一些跟其他工具一起实现的示例。
- ◆ BioVisualize (<http://biovisualize.github.io/d3visualization>), 算是一个分门别类的 D3 gallery, 可以帮你快速地在在线查找你想要的例子。
- ◆ D3 教程 (<https://github.com/mbostock/d3/wiki/Tutorials>), 有很多朋友不断更新提供的教程、讨论和文档, 给你细致地演示了 D3 的使用方法。
- ◆ D3 插件 (<https://github.com/d3/d3-plugins>), 可能有些你需要的功能是 D3 没有的。在你自己实践之前, 先查查 D3 的插件库, 它提供了各种常用的、不常用的功能。
- ◆ D3 API (<https://github.com/mbostock/d3/wiki/API-Reference>) 是很不错的文档。这里你能找到 D3 所提供的全部功能、属性的详细说明。
- ◆ Mike Bostok's Blocks (<http://bl.ocks.org/mbostock>) 是个 D3 示例站点, 作者是 Mike Bostock, 这个站点里有很多有趣的例子。
- ◆ JS Bin (<http://jsbin.com/ugacud/1/edit>) 是个在线的 D3 测试、实验环境。你可以很容易地通过该工具跟其他人分享一些简单的代码。
- ◆ JS Fiddle (<http://jsfiddle.net/qAHC2/>) 跟 JS Bin 差不多, 也是一个在线 JavaScript 代码分享平台。

如何获取帮助

即便有了这些例子、教程, 还有书, 你在实践的过程里仍然会遇到问题。不过 D3 有数目庞大并且活跃度不错的社区。一般情况, 简单地 “google” 一下, 就能找到满意的答案。要是没有也不用担心, D3 还有强大的社区支持。

- ◆ StackOverFlow 上的 D3.js (<http://stackoverflow.com/questions/tagged/d3.js>): StackOverflow 是最著名的免费技术主题问答社区站点, D3 在 StackOverflow 上有专门的页面, 可以帮你找到专家, 快速地解决你的问题。
- ◆ D3 Google 讨论组 (<https://groups.google.com/forum/?fromgroups#!forum/d3-js>): 这是个官方的用户组, 不单单有 D3, 还有一些其他相关的库。