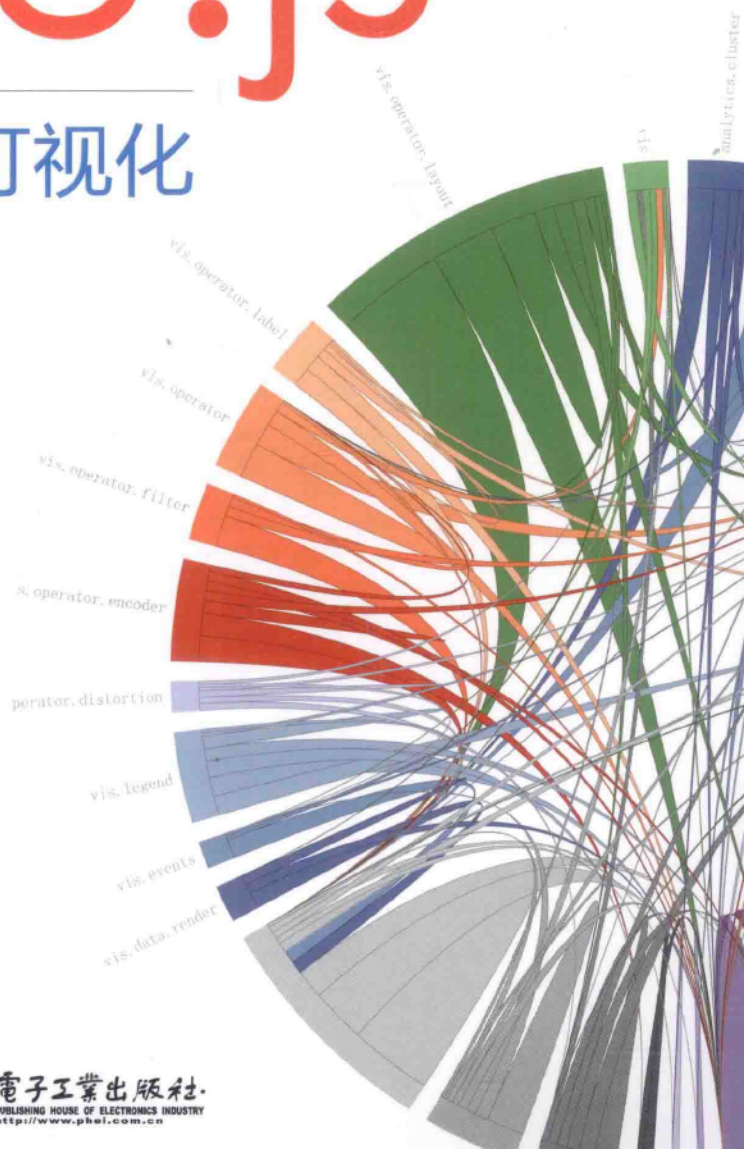


精通 D3.js

• 吕之华 著 •

交互式数据可视化 高级编程

大数据时代需要可视化工具，
D3是世界最流行的可视化函数库。



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

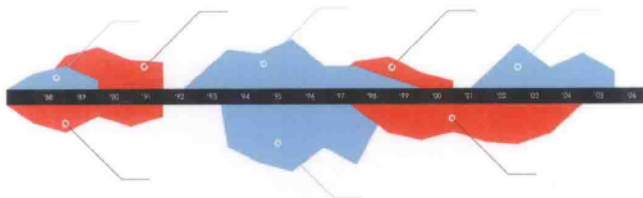
提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ1779903665.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签索引和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我QQ1779903665。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，与本人无关，我们仅仅只是帮助你寻找到你要的pdf而已。



大数据时代需要可视化工具，D3是世界最流行的可视化函数库。本书手把手教你学会D3，从零讲起直到高级应用，既是教程，又可作为参考手册，查阅D3各种方法的使用。内容图文并茂，示例丰富，帮助你轻松生成各种漂亮图形。

—— 阮一峰

• 内容简介 •

本书以当前流行的数据可视化技术D3.js为主要内容，分为三大部分，共计13章。第一部分讲述基础知识，第二部分学习制作各种常见图表，第三部分讲解交互式图表及地图的进阶应用。本书是一个相对完整的D3.js教程，讲解此技术所有重要的知识点，既有基础入门知识，又有相对深入的内容。笔者秉持以下原则：由易到难，循序渐进，图文并茂，清晰易懂。

↓ 书中源代码下载网址：

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

<http://broadview.com.cn/26776>



博文视点Broadview



@博文视点Broadview

上架建议：计算机 > Web前端

ISBN 978-7-121-26776-5



9 787121 267765 >

定价：79.00元



责任编辑：付 睿
封面设计：侯士卿

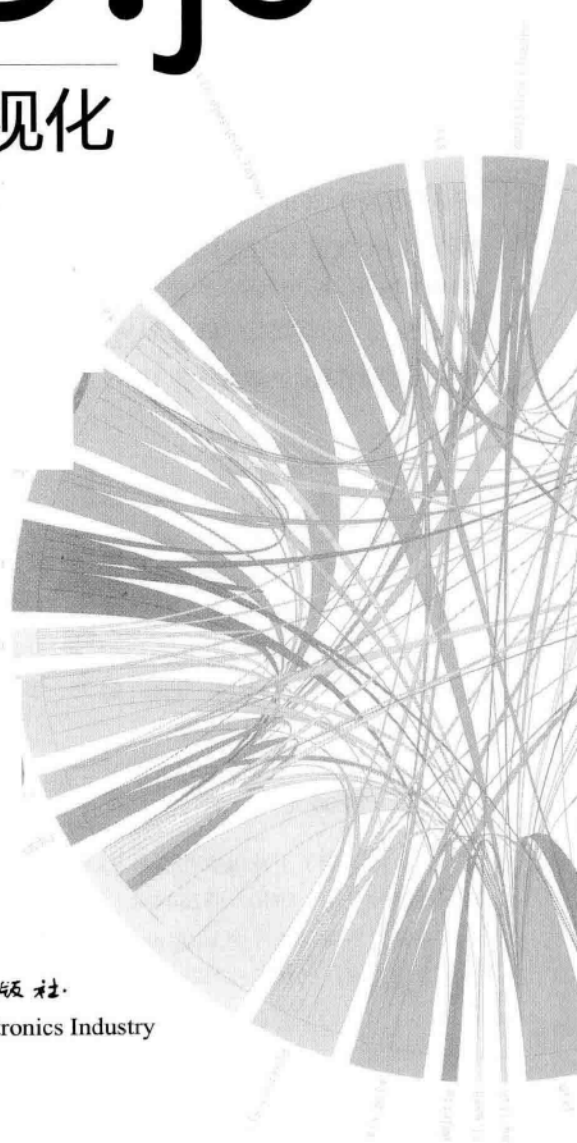
精通D3.js

• 吕之华 著 •

交互式数据可视化
高级编程

電子工業出版社

Publishing House of Electronics Industry



内 容 简 介

本书以当前流行的数据可视化技术 D3.js 为主要内容,分为三大部分,共计 13 章。第一部分讲述基础知识,第二部分学习制作各种常见图表,第三部分讲解交互式图表及地图的进阶应用。本书作为一个相对完整的 D3.js 教程,讲解此技术所有重要的知识点,既有基础入门知识,又有相对深入的内容。笔者秉持以下原则:由易到难,循序渐进,图文并茂,清晰易懂。

本书适合有一定计算机基础的读者,需要熟悉 C、C++、Java、JavaScript 等至少一门编程语言,能够理解基础的数据结构和算法。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

精通 D3.js: 交互式数据可视化高级编程/吕之华著. —北京: 电子工业出版社, 2015.8

ISBN 978-7-121-26776-5

I. ①精... II. ①吕... III. ①JAVA 语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2015)第 169494 号

责任编辑: 付睿

印 刷: 北京丰源印刷厂

装 订: 三河市华成印务有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×980 1/16 印张: 25.25 字数: 562 千字 彩插: 9

版 次: 2015 年 8 月第 1 版

印 次: 2015 年 8 月第 1 次印刷

定 价: 79.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

目 录

第 1 章 D3 简介	1
1.1 D3 是什么	1
1.1.1 D3 简史	2
1.1.2 D3 的优势	2
1.1.3 D3 的适用范围	3
1.2 数据可视化是什么	3
1.2.1 目的	4
1.2.2 构成要素	4
1.2.3 相关概念	6
1.3 图表种类	6
1.4 学习方法	11
第 2 章 Web 前端开发基础	13
2.1 浏览器和服务器	14
2.1.1 浏览器	14
2.1.2 服务器	15
2.2 HTML&CSS	16
2.2.1 HTML 元素	17
2.2.2 CSS 选择器	20
2.2.3 综合运用	23
2.3 JavaScript	25
2.3.1 在 HTML 中使用 JavaScript	26
2.3.2 语法	26
2.3.3 变量	27
2.3.4 数据类型	28

2.3.5	操作符	30
2.3.6	语句	32
2.3.7	函数	34
2.3.8	对象	34
2.3.9	数组	35
2.4	DOM	36
2.4.1	结构	37
2.4.2	访问和修改 HTML 元素	37
2.4.3	添加和删除节点	38
2.4.4	事件	39
2.5	SVG	40
2.5.1	位图和矢量图	40
2.5.2	图形元素	41
2.5.3	文字	46
2.5.4	样式	47
2.5.5	标记	48
2.5.6	滤镜	50
2.5.7	渐变	51
第 3 章	安装和使用	53
3.1	安装	53
3.1.1	下载文件	54
3.1.2	网络引用	54
3.2	搭建服务器	54
3.3	Hello, World	57
3.4	绘制矢量图	58
3.5	调试	59
第 4 章	选择集与数据	61
4.1	选择元素	61
4.2	选择集	63
4.2.1	查看状态	63
4.2.2	设定和获取属性	63
4.3	添加、插入和删除	66
4.4	数据绑定	67

4.4.1	datum()的工作过程	68
4.4.2	data()的工作过程	71
4.4.3	绑定的顺序	74
4.5	选择集的处理	76
4.5.1	enter 的处理方法	76
4.5.2	exit 的处理方法	77
4.5.3	处理模板	78
4.5.4	过滤器	79
4.5.5	选择集的顺序	79
4.5.6	each()的应用	80
4.5.7	call()的应用	80
4.6	数组的处理	81
4.6.1	排序	81
4.6.2	求值	82
4.6.3	操作数组	86
4.6.4	映射 (Map)	89
4.6.5	集合 (Set)	91
4.6.6	嵌套结构 (Nest)	92
4.7	柱形图的制作	96
4.7.1	矩形和文字	97
4.7.2	更新数据	101
第 5 章	比例尺和坐标轴	105
5.1	定量比例尺	105
5.1.1	线性比例尺	106
5.1.2	指数和对数比例尺	109
5.1.3	量子 and 分位比例尺	110
5.1.4	阈值比例尺	112
5.2	序数比例尺	113
5.3	坐标轴	118
5.3.1	绘制方法	119
5.3.2	刻度	121
5.3.3	各比例尺的坐标轴	122
5.4	柱形图的坐标轴	123

5.5 散点图的制作	125
第 6 章 绘制	128
6.1 颜色	128
6.1.1 RGB	129
6.1.2 HSL	130
6.1.3 插值	131
6.2 线段生成器	132
6.3 区域生成器	136
6.4 弧生成器	137
6.5 符号生成器	140
6.6 弦生成器	142
6.7 对角线生成器	144
6.8 折线图的制作	145
第 7 章 动画	151
7.1 过渡效果	151
7.1.1 过渡的启动	152
7.1.2 过渡的属性	155
7.1.3 子元素	158
7.1.4 each()和 call()	160
7.1.5 过渡样式	162
7.2 定时器	163
7.2.1 setInterval 和 setTimeout	163
7.2.2 d3.timer	164
7.3 应用过渡的场合	165
7.4 简单的动画制作	171
7.4.1 时钟	171
7.4.2 小球运动	172
第 8 章 交互	174
8.1 交互式入门	174
8.1.1 鼠标	176
8.1.2 键盘	178
8.1.3 触屏	180

8.2	事件	182
8.3	行为	183
8.3.1	拖曳	184
8.3.2	缩放	186
第 9 章	导入和导出	191
9.1	文件导入	191
9.1.1	JSON	192
9.1.2	CSV	194
9.1.3	XML	198
9.1.4	TEXT	199
9.2	文件导出	200
9.2.1	导出为 SVG 文件	200
9.2.2	编辑矢量图	203
第 10 章	布局	206
10.1	布局是什么	206
10.2	饼状图	207
10.3	力导向图	213
10.4	弦图	221
10.5	树状图	228
10.6	集群图	234
10.7	捆图	238
10.8	打包图	245
10.9	直方图	248
10.10	分区图	255
10.11	堆栈图	261
10.12	矩阵树图	268
第 11 章	地图	274
11.1	地图的数据	274
11.1.1	获取数据	275
11.1.2	简化数据	278
11.1.3	GeoJSON	280
11.1.4	TopoJSON	284

11.2	中国地图	285
11.2.1	基于 GeoJSON	285
11.2.2	基于 TopoJSON	289
11.3	地理路径	297
11.3.1	地理路径生成器	297
11.3.2	形状生成器	301
11.4	投影	306
11.5	球面数学	315
第 12 章	友好的交互	317
12.1	提示框	317
12.1.1	饼状图的提示框	318
12.1.2	提示框的样式	321
12.2	坐标系中的焦点	323
12.2.1	折线图的焦点	323
12.2.2	为折线图添加提示框	329
12.3	元素组合	334
12.3.1	饼状图的拖曳	335
12.3.2	移入和移出	336
12.3.3	合并	345
12.4	区域选择	347
12.4.1	在 SVG 画板里选择一块区域	348
12.4.2	散点图的区域选择	350
12.5	开关	353
12.5.1	思维导图的构造思路	353
12.5.2	思维导图的制作	356
第 13 章	地图进阶	363
13.1	值域的颜色	363
13.2	标注	368
13.2.1	标注地点	368
13.2.2	夜光图	370
13.3	标线	373
13.3.1	带有箭头的标线	373
13.3.2	球状地图的标线	377

13.4 拖动和缩放	378
13.4.1 平面地图	378
13.4.2 球面地图	381
13.5 力导向地图	383
13.5.1 Voronoi 图和 Delaunay 三角剖分	383
13.5.2 力导向的中国地图	387
附录 A 彩色插图	393
附录 B 参考文献	410

第 1 章

D3 简介

本章内容包括：

D3 是什么

- 数据可视化是什么
- 常见可视化图表的种类
- 学习 D3 的方法

笔者第一次接触到 D3，首先是被绚丽多彩的图表吸引，然后陶醉于这些图表的可交互特性，而且动画流畅简洁、赏心悦目。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“一张图片价值相当于一千个字”，的确是名副其实。对于信息爆炸式增长的今天，我们没有时间一条一条地阅读，希望的是一眼就能找到自己想要的信息。图片不仅容易理解，而且容易记忆，是值得推广的信息传达方式。

1.1 D3 是什么

D3 的全称是 Data-Driven Documents，直译为“数据驱动的文档”。听名字有点抽象，简单概括为一句话：

D3 是一个 JavaScript 的函数库，是用来做数据可视化的。

文档指 DOM，即文档对象模型（Document Object Model）。D3 允许用户绑定任意数据到

DOM, 然后根据数据来操作文档, 创建可交互式的图表。

JavaScript 文件的后缀名通常为.js, 故 D3 也常称为 D3.js。D3 提供了各种简单易用的函数, 大大简化了 JavaScript 操作数据生成图表的难度。由于它本质上是 JavaScript, 所以用 JavaScript 也是可以实现所有功能的, 但 D3 能大大减轻你的工作量, 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。

1.1.1 D3 简史

大数据时代蓬勃发展的今天, 每天都有惊人的数据产生, 怎么提取并显示有用的信息, 变得越来越重要。在 Web 浏览器中进行可视化也成为迫切的需求, 有许多项目以此为目标。2009 年, Mike Bostock、Jeff Heer、Vadim Ogievetsky 共同开发了 Protovis, 可以算是 D3 的前身。2011 年, 他们停止了 Protovis, 使用 JavaScript 开发了一个新的项目, 这就是 D3。

JavaScript 诞生于 1995 年, 经过近 20 年的发展, 如今已经成为 Web 浏览器上事实上的标准语言, 其应用范围扩展到服务器端、移动领域等, 并且越来越完善。因此, D3 采用 JavaScript 作为开发语言。

2011 年 2 月 18 日, Mike Bostock 发布了 v1.0 版本。

2011 年 8 月 25 日, v2.0 版本发布, 功能大幅增强, 应用逐渐增多。

2012 年 12 月 22 日, v3.0 版本发布, 修复了大量 bug, 功能更加稳健。

笔者写作时, 最新版本为 v3.4.13。v3.0 版之后差别不大, 本书以 v3.x 版为标准。

1.1.2 D3 的优势

可视化的库有很多, 基于 JavaScript 开发的库也有很多, D3 有什么优势呢?

1. 数据能够与 DOM 绑定在一起

D3 能够将数据与 DOM 绑定在一起, 使得数据与图形成为一个整体, 即图形中有数据、数据中有图形。那么在生成图形或更改图形时, 就可以方便地根据数据进行操作。并且, 当数据更改之后, 图形的更新也会很方便。

2. 数据转换和绘制是独立的

将数据变成图表, 需要不少数学算法, 一些可视化库的做法是: 提供一个函数 drawPie(), 输入数据, 直接绘制出饼状图。

D3 的做法是: 提供一个函数 computePie(), 可将数据转换成饼状图的数据, 然后开发者使用自己喜欢的方式来绘制饼状图。

看起来,好像 D3 使问题变麻烦了,但是在图表比较复杂的时候,直接绘制的饼状图往往达不到要求,细微的部分没有办法更改。将两者分开,就极大地提高了自由度,以至于开发者甚至可以使用其他的图形库来显示 D3 计算的数据。

3. 代码简洁

jQuery 是网页开发中很常用的库,其链式语法被很多人喜爱。D3 也采用了这一语法,能够一个函数套一个函数,使得代码很简洁。

4. 大量布局

饼状图、树形图、打包图、矩阵树图等,D3 将大量复杂的算法封装成一个个“布局”,能够适用于各种图表的制作。

5. 基于 SVG, 缩放不会损失精度

SVG,是可缩放的矢量图形。D3 大部分是在 SVG 上绘制的,并且提供了大量的图形生成器,使得在 SVG 上绘制图形变得简单。另外,由于 SVG 是矢量图,所以放大缩小不会有精度损失。

1.1.3 D3 的适用范围

D3 开发的应用是显示在网页上的。因此,开发者需将数据置于服务器端,并在网页文件(HTML)中插入 D3 代码。用户通过浏览器请求此网页文件,就会看见开发者希望让用户看到的可视化内容。

Ben Fry 在他的著作《Visualizing Data》中将数据可视化的过程分为七个步骤。

- (1) 获取——Acquire
- (2) 分析——Parse
- (3) 过滤——Filter
- (4) 挖掘——Mine
- (5) 表现——Represent
- (6) 改善——Refine
- (7) 交互——Interact

前四步不属于 D3 的处理范围,更多的是处理后三步,即表现、改善、交互。

1.2 数据可视化是什么

数据可视化(Data Visualization)起源于 18 世纪,William Playfair 在出版的书籍《The Commercial and Political Atlas》中第一次使用了柱形图和折线图,当时是为了表示国家的进出口

量，在今天依然这么使用。19 世纪初，出版了《Statistical Breviary》一书，里面第一次使用了饼状图，这三种图形都是至今最常用的最著名的可视化图形。19 世纪中叶，数据可视化主要被用于军事用途，用来表示军队死亡原因、军队的分布图等。进入 20 世纪，数据可视化有了飞跃性的发展。1990 年，在人机界面学会上，作为信息可视化原型的技术被发表。1995 年，IEEE Information Visualization 正式创立，信息可视化作为独立的学科被正式确立。近年，随着大数据时代的到来，数据可视化作为大量数据的呈现方式，成为当前重要的课题。

1.2.1 目的

The main goal of data visualization is its ability to visualize data, communicating information clearly and effectively.

数据可视化的目的，是要对数据进行可视化处理，以使得能够明确地、有效地传递信息。

— Vitaly Friedman

比起枯燥乏味的数值，人类对于大小、位置、浓淡、颜色、形状等能够有更好、更快的认识。经过可视化之后的数据能够加深人对于数据的理解和记忆。

例如有以下的数据，请找出最大值：

[321, 564, 1391, 245, 641, 798, 871]

数据量比较小，用肉眼也能找出来，但更好的办法是将数据进行可视化处理，如图 1-1 所示。

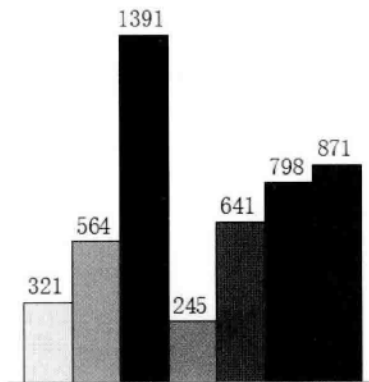


图 1-1 可视化后的柱形图

如图 1-1 所示，很明显，经过可视化之后，数据变得容易理解了。

1.2.2 构成要素

数据可视化的手法很多，其中有一些共通的视觉要素。

- 坐标。数值的位置被对应到直角坐标系或极坐标系上，如图 1-2 所示。

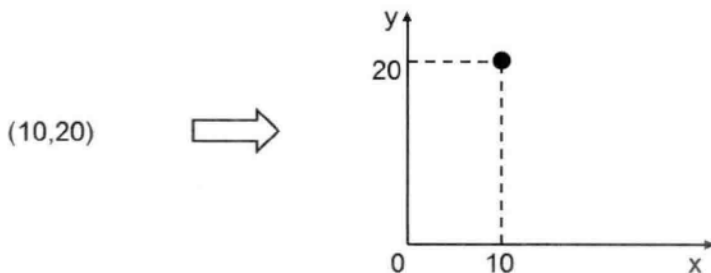


图 1-2 对应到坐标系上

- 大小。数值的大小被对应到图形的大小上，如图 1-3 所示。

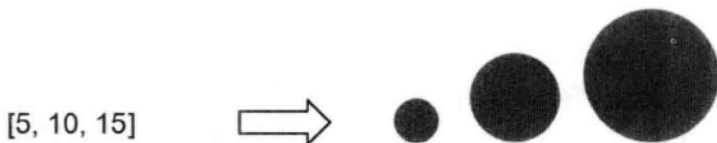


图 1-3 对应到大小上

- 色彩。数值的分类和界限等对应到不同的颜色上，如图 1-4 所示。

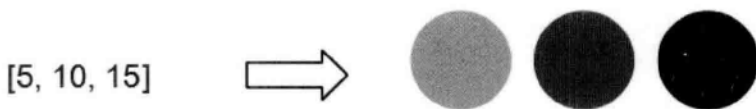


图 1-4 对应到颜色上

- 标签。数值的特征用标签来标记，如图 1-5 所示。



图 1-5 对应到标签上

- 关联。数值之间的联系，用关联线条等连接起来，如图 1-6 所示。

如图 1-2 至图 1-6 所示，列举了一些常见的将数据对应到视觉要素的方式。这些方式经过多数人的使用，是最容易被人理解也是最容易制作的，但是视觉要素并非局限于此。

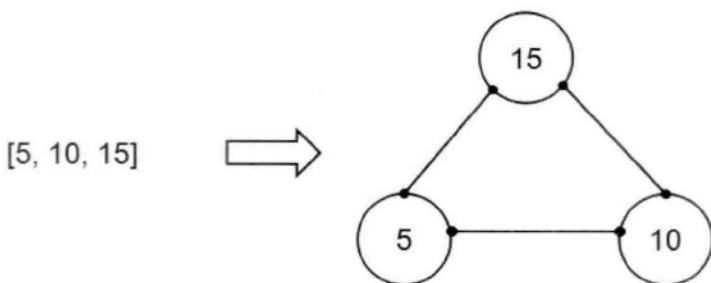


图 1-6 对应到关联线条上

1.2.3 相关概念

数据可视化（Data Visualization）和信息可视化（Information Visualization）很相近，有时几乎可以等同。但严格来说它们是不同的，它们的不同可以总结为一句话：

数据可视化是对数字信息进行可视化，信息可视化是对**数字信息**和**非数字信息**进行可视化。

1.3 图表种类

用于数据可视化的图表种类相当多，这里列举一些常用的图表。实际应用时可单独使用，也可以多种联动。

一般来说，图表要尽可能简单，能用简单的就用简单的。有的人可能会觉得简单的图表太古老、不大气，而追求复杂的图表，这反而有点本末倒置。数据可视化的目的，是要使数据明确地、有效地传递，而简单的图表是能够最快被人认可的。凭感觉自创图表也是可以的，但要注意此图表是否比原来的更简单易懂。

下面列举部分可视化图表。要注意，一种图表对应的汉语名称可能有多种，在本节中使用的名称将作为本书的标准。

1. 柱形图

柱形图是最常见、最容易理解的图表，使用矩形的长短来表示数据的大小。数据类型一般是形如“时间—销售额”这样的二维数据集，图表要表现的是“随着时间的变化，销售额的变化情况”（如图 1-7 所示）。

如将图 1-7 的 x 轴和 y 轴替换，得到横向的柱形图，有时也称为条形图，但本质是一样的，都是用柱形的长短来表示数据的大小。此外，还可矩形的宽窄来表示第三维的数据，使得一个图里的信息量更大，但会加深理解的难度。

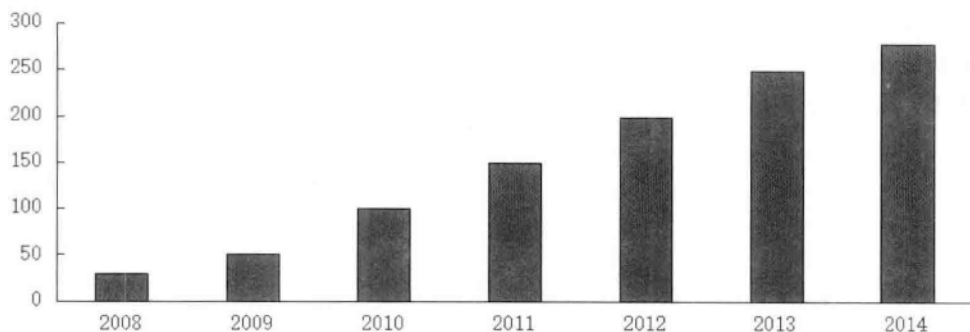


图 1-7 柱形图

2. 散点图

散点图使用三维数据集，将其中的二维数据分别对应到 x 轴和 y 轴，再将第三维用点表示，而第三维数据是对应前二维的。其直观表现为在 x 轴和 y 轴的坐标系中分布着很多点，如图 1-8 所示。

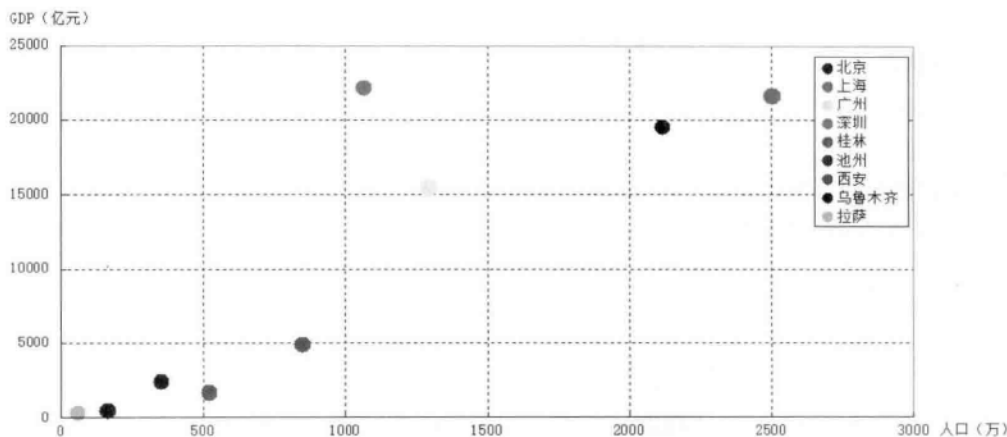


图 1-8 散点图

图 1-8 中 x 轴表示人口， y 轴表示 GDP，第三维数据为城市，所表示的内容为各城市在人口和 GDP 的二维坐标系中的分布。

3. 折线图

折线图的目的与柱形图类似，也适合表示在二维数据集中，某一维相对于另一维的变化趋势，不同的是：

- 折线图较适合连续的数据，柱形图较适合离散的数据。

- 折线图较适合大量的数据，柱形图较适合少量的数据。
- 折线图用于表示多个数据集之间的比较时，效果较好。

图 1-9 表示的是中国和日本从 2005 年至 2013 年 GDP 的变化趋势。

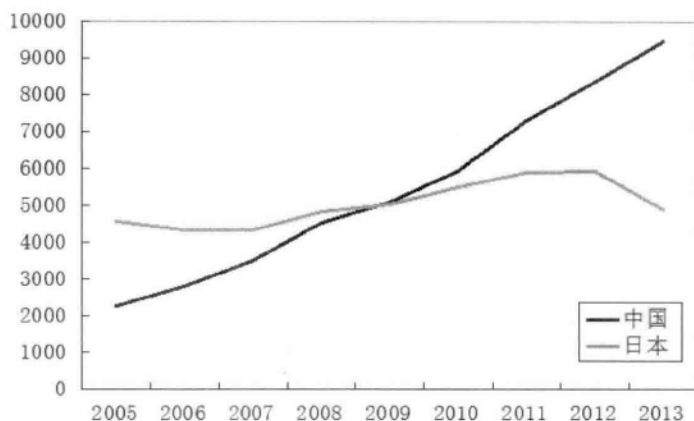


图 1-9 折线图

4. 饼状图

饼状图可以用于比较数值的大小，但是有一个缺点：如果数值之间差距不大，肉眼很难分辨。因此，最好用于表示某一个值占全体值的百分比，比如 2014 年各浏览器占市场份额的百分比、各操作系统的份额百分比、各编程语言的使用比等。

如图 1-10 所示，饼状图的每一块都用标签表示出来，也可以用线连接到外部表示。另外，饼状图还有一些变种，如各扇形的半径不同，该半径可表示另一个数据量。

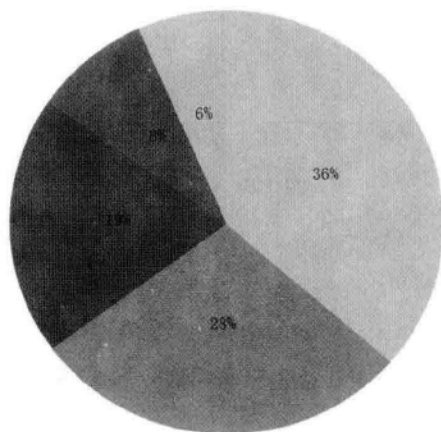


图 1-10 饼状图

5. 弦图

弦图，主要用于表示节点之间的联系。两点之间的连线表示哪两个节点具有联系，线的粗细表示权重。如图 1-11 所示，表示各城市的人口来自于哪些城市，后面章节会有详细介绍，这里只要有个印象即可。

6. 力导向图

力导向图适合描述大量顶点之间的关系，各顶点之间具有相互的作用力。如图 1-12 所示，各顶点之间用线相连，相连的顶点表示具有一定的关系。实际应用时可以赋予顶点和连线各种意义，如可做成人物关系图、力导向地图等，具有很大的扩展性。

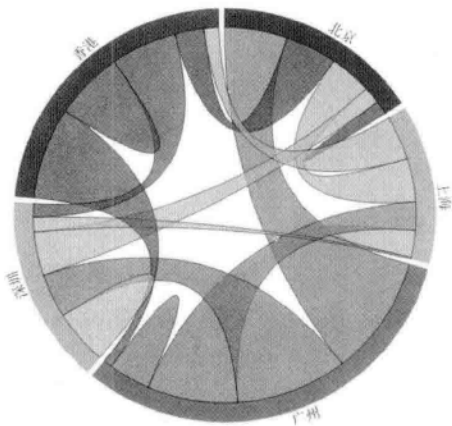


图 1-11 弦图

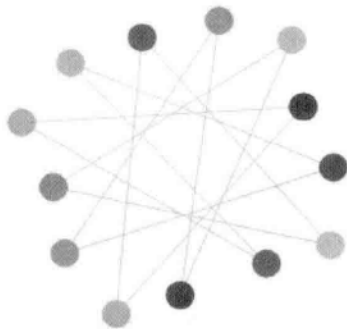


图 1-12 力导向图

7. 树状图

树状图用于表示层级、上下级、包含与被包含关系，与之类似的还有集群图。如图 1-13 所示，像树枝一样展现出省份和城市的包含关系。

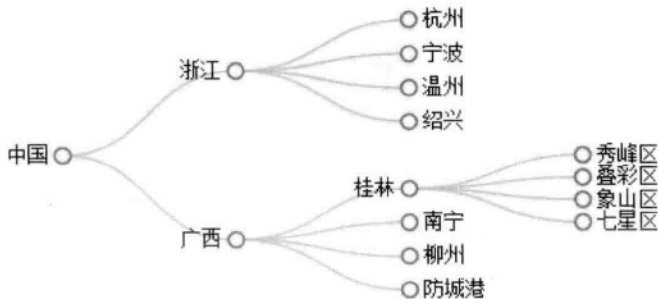


图 1-13 树状图

8. 打包图

打包图，用于表示包含与被包含的关系，也可表示各对象的权重。如图 1-14 所示，圆内套圆表示节点的关系，圆的大小表示节点的权重。

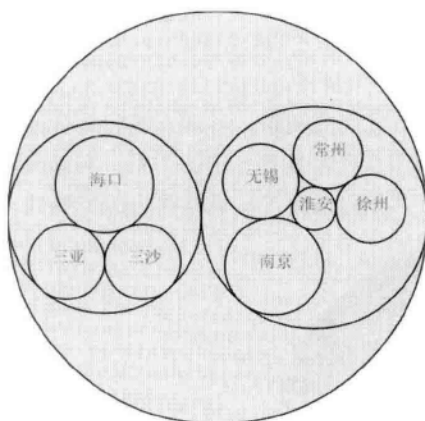


图 1-14 打包图

9. 分区图

分区图用于表示包含与被包含关系，其表现形式很像将硬盘分区，如图 1-15 和图 1-16 所示。

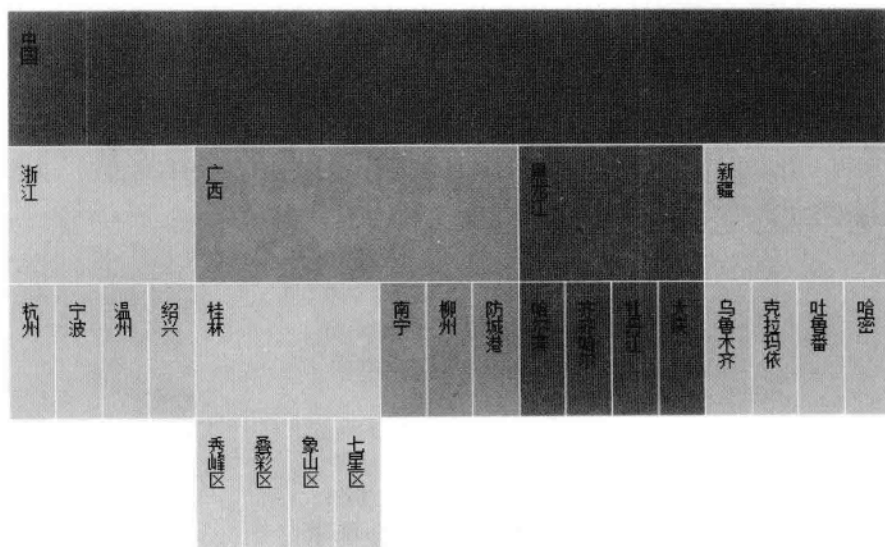


图 1-15 矩形分区图

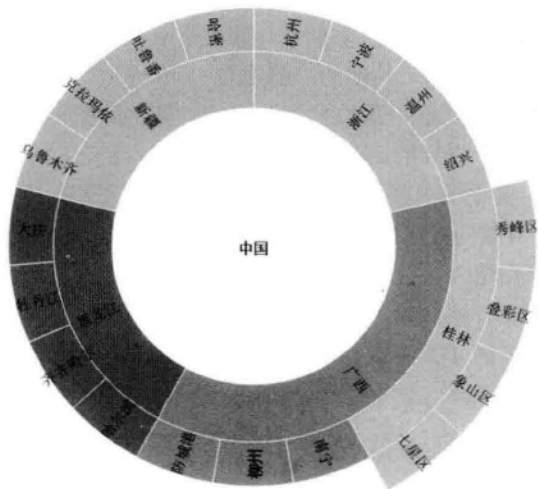


图 1-16 圆形分区图

1.4 学习方法

首先，不要做以下两件事。

- 看到某一个图表，感觉很好，立即复制下来，希望通过简单地更改数据或参数，就能达到自己使用的目的。
- 在完全不了解 JavaScript 的情况下学习 D3。D3 编程不一定用得到高深语法，但基础知识是必要的。

那么，D3 难学吗？

有不少人认为 D3 挺难学的，原因有三。

1. 官方文档写得比较难

官网上提供了 API 文档，还有大量的例子。但是，大部分例子只有代码，没有文字说明。API 虽有说明，但是却没有太多针对性的例子，使初学者感觉头大。

2. 不好理解数据转换和绘制分开的模式

一个函数，`drawPie()`，输入数据，输出绘制图形，一般人的思维模式是这样的。但是，D3 偏偏将两者分开了，分开之后能带来极大的自由度，但是也使得初学者理解它有些困难。

3. 外语不好

对大部分国人来说，看英文文档还是挺头疼的，而中文资料相对不够丰富。

乍看上去, D3 有些难学, 但是一旦掌握了, 就能适应各种图表的制作, 自由度大, 功能极强。有人说, D3 就像是 Photoshop, 其他的库就像是 Windows 画图板: 前者需要一定的时间学习, 学成后在图像处理上所向披靡; 后者不需要学习时间, 会 and 不会没有太大的价值。这么比喻可能有点夸张, 笔者有一个更好的比喻 (灵感来源于辜鸿铭先生的文章)。

D3 就像是写毛笔字, 其他的可视化库就像是写钢笔字。钢笔字上手容易, 下笔简单、快捷, 写出来的东西叫作**文章**。毛笔字需要长期磨炼, 上手较难, 但是一旦掌握了, 便能行云流水, 心随念想, 可进可退, 只在笔尖, 写出来的东西叫作**艺术**。

建议初学者从简单图表开始做起, 尤其是柱形图、散点图、折线图这三种最基础的图表。通过反复练习基础图表, 掌握 D3 各大功能模块的运用方法, 把基础打好。如此, 在制作复杂图表时才能起到事半功倍的效果。如果急于求成, 一开始就着眼于“思维导图”这样的图表, 会步履维艰, 断不可为。

下面是一些学习 D3 的网站:

- <http://d3js.org/>

D3 的官方网站, 含有 API 和大量示例。

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

D3 创始人制作的, 有很多说明文档。

- <https://www.dashingd3js.com/table-of-contents>,

非常简单易懂的教程, 文字解释、图片都十分清晰。此站开设的目的就是为了让人们迅速而高效地掌握 D3。

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

笔者站点, 有 D3 的一系列教程。

本书分为基础知识、常用图表的制作、深度应用三个部分。难度上由浅入深, 建议依顺序阅读, 在可以选择跳过的部分会有提示。

第 2 章

Web 前端开发基础

本章内容包括：

- 浏览器和服务端
- HTML 和 CSS 基础
- JavaScript 基础
- DOM 基础
- SVG 基础

本章简单介绍 Web 前端开发的基础，是学习 D3 的预备知识，主要针对没有前端基础的读者。有一定基础的读者可选择性阅读，或遇到不明问题时再查询即可。

第 1 节，介绍浏览器和服务端之间是如何进行交互的。理解了这一点，才知道要在什么地方使用 D3。

第 2 节，讲述 HTML 和 CSS 的基础知识。HTML 是用于描述网页内容的，CSS 是用于定义网页样式的，它们相互独立却常一起出现。

第 3 节，学习 JavaScript，一种直译式脚本语言，用于设定网页的行为。D3 就是基于 JavaScript 开发的。

第 4 节，介绍 DOM，即文档对象模型。它是针对结构化文档的一个接口，使用它可以动态地访问和修改 HTML 文档。

第 5 节，学习 SVG（可缩放矢量图形）。SVG 是一种画图板，绘制出来的是矢量图，D3 的图形大部分是绘制在 SVG 上的。

2.1 浏览器和服务

浏览器 (Browser) 对我们来说太重要了, 每天早上看新闻要用到, 工作时查询资料要用到, 看网络视频要用到, 休闲娱乐也要用到。如果要评选每天使用得最多的软件, 恐怕就是浏览器了。那么, 当输入网址进入网站后, 见到的图片文字是从哪来的, 怎么传输过来的, 这些信息是存储在哪里的, 什么人准备的呢?

如图 2-1 所示, 管理员在服务器 (Server) 存储各种信息, 包括主页、图片、音乐、视频等, 服务器与因特网 (Internet) 相连。很多与因特网相连的用户, 通过个人电脑中的浏览器, 发送请求给服务器, 服务器就将其需要的信息 (文字图片等) 传送给用户, 用户就能够在浏览器上享受到各种服务。

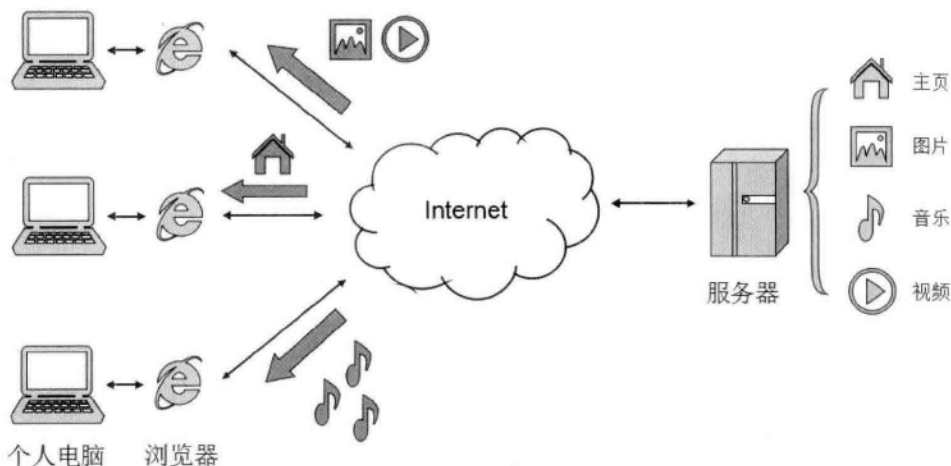


图 2-1 浏览器与服务器的交互

当然, 浏览器与服务器的交互实际上要复杂得多, 但理解本书的内容不需要知道那么多细节, 有上述内容已经足够。

2.1.1 浏览器

浏览器软件有很多, 常见的有:

(1) Internet Explorer (简称 IE) 是最常见的浏览器, 由 Windows 系统自带, 发行于 1995 年, 使用人数最多。目前的最新版本为 IE 12, 还在使用的最古老的版本可能是 IE 6。IE 系列占浏览器市场份额的一半以上, 远超其同行。制作网页时必须考虑到用户所使用的浏览器版本,

有些遗憾的是, D3 对 IE 8 及以下版本支持不好。有人为使 D3 兼容 IE 8 以下版本, 做了各种努力, 效果却差强人意。不过, 使用 IE 8 以下版本的用户目前已不足 17%, 并且还在不断下降, 旧版本被淘汰是大势所趋, 因此不必过于担心此问题。

(2) Firefox (火狐) 浏览器是相当受欢迎的一款浏览器, 目前已占有全球市场份额的 13%, 在某些国家的占有率超过 80%。Firefox 是 Mozilla 基金会开发的开源浏览器, 以其稳定的性能、良好的安全性、丰富的组件著称。由于其开源性, 大量的程序员、研究员、工程师在上面开发扩展组件, 使其几乎无所不能。

(3) Chrome (谷歌) 浏览器是由 Google 公司开发的浏览器。其特点是简单、快速, 并且不易崩溃。其界面相当简洁, 几乎所有空间都用于显示网页, 而且不仅浏览网页的速度快, 打开软件的速度也快, 能明显感觉到与别的浏览器之间的区别。本书中的所有代码, 都是用 Chrome 进行测试的。

IE、Firefox、Chrome 三款浏览器所占市场份额之和超过 90%, 因此使用这三款对网页程序进行测试最重要。其他还有在苹果计算机的系统 Mac OS X 中使用的 Safari 浏览器, Opera 浏览器, 符合国人用户习惯的 360、搜狗、傲游等浏览器, 种类很多, 如图 2-2 所示, 有些浏览器的内核是一样的, 只是根据特定用户的习惯做了组件的扩展。D3 可运行于 IE 9+、Firefox、Chrome、Safari、Opera 等浏览器。



图 2-2 各种浏览器的图标

2.1.2 服务器

服务器是一个管理资源并为用户提供服务的计算机。有各种规模的服务器, 最小的可能与个人电脑配置差不多, 但在处理能力、稳定性、安全性等方面还是有差异的。近年来出现了云服务器, 只需要支付相当低的价格, 即可拥有服务器, 并且其管理方式比物理服务器更加简单高效。

要使服务器能够提供各种服务, 需要安装服务器软件。如果要提供 Web 信息浏览服务, 需要安装 Web 服务器软件, 其主要需支持 HTTP 协议, 能处理用户发送过来的 HTTP 请求。常见

的 Web 服务器软件有 Apache、Tomcat、IIS 等。

(1) Apache HTTP Server (简称 Apache) 是 Apache 软件基金会的一个开放源码的 Web 服务器软件, 可以在大多数计算机操作系统中运行。由于其多平台和安全性被广泛使用, 是最流行的 Web 服务器端软件之一。

(2) Tomcat 也是 Apache 软件基金会一个著名的服务器软件, 其与 Apache 的区别和联系如下。

- Apache 只支持 HTML 静态网页, 通过插件可支持 PHP; Tomcat 支持 ASP、JSP、PHP、CGI 等动态网页。
- Apache 是用 C 语言实现的; Tomcat 使用 Java 实现的, 更好地支持 Servlet 和 JSP。
- Apache 的稳定性较好。
- Apache 对于静态页面的解析速度比 Tomcat 快。
- Apache 比 Tomcat 早, 本质上来说 Tomcat 的功能可以替代 Apache。

(3) Internet Information Services (IIS) 是微软公司提供的服务器软件, 除了提供 Web 服务之外, 还提供 FTP 服务、SMTP 服务等。其使用比较简单, 早期版本漏洞较多, 但从 IIS 6.0 版本开始做了大量修复, 安全性有了大幅增长。

Web 服务器软件并非只是服务器计算机才能安装, 在个人电脑上也能安装, 这对于开发者测试是十分方便的。

本书中的示例, 都使用 **Apache HTTP Server** 进行测试。安装 Apache 的方法在第 3 章有详细介绍。

2.2 HTML&CSS

HTML (Hyper Text Markup Language) 指的是超文本标记语言, CSS (Cascading Style Sheets) 指的是层叠样式表。用浏览器打开任意一个网页, 右键单击页面, 选择“查看网页源代码”命令, 即可看到用于描述网页内容的源码信息。最常见到的结构有两种, 第一种形如:

```
<!DOCTYPE html>
<html>
  <head>
    <title>宋词</title>
  </head>
  <body>
    <h1>一剪梅</h1>
    <p>红藕香残玉簟秋, 轻解罗裳, 独上兰舟</p>
    <p>云中谁寄锦书来, 雁字回时, 月满西楼</p>
  </body>
</html>
```

另一种代码的样式形如：

```
p {  
    color:red;  
    font-family:simsun;  
    font-size:20px;  
}
```

前者就是 HTML 代码，用于定义文档的结构和内容，例如显示什么文字、用段落显示还是表格显示等。后者是 CSS 代码，用于定义 HTML 元素的样式，如字体大小、背景颜色、布局等。

要使用 HTML 和 CSS 来制作网页，只需要在记事本程序里编写即可。新建一个后缀为.txt 的文本文件，然后将后缀改为.html，再用记事本程序打开后编写代码。编写完保存之后，再使用浏览器打开此文件，即可浏览效果。但是为了方便，一般使用功能更强大的记事本软件（例如 Notepad++、Sublime Text），有代码高亮的功能，能加快开发速度。

2.2.1 HTML 元素

HTML 是一种标记语言，每个元素都有一个标记标签（Markup Tag）。上面的代码中，已经使用了<html>、<body>、<p>等标签，都是用一对尖括号包围的。标签有成对出现的，如<p></p>；也有不成对出现的，如
。对于成对出现的标签，前一个叫作开始标签，不带斜杠；后一个叫作结束标签，带斜杠。

1. 文档声明

在上面的代码中首先出现的是<!DOCTYPE>，这是一个声明，不是 HTML 标签，其主要目的是告诉浏览器 HTML 的版本信息。此声明需要在 HTML 的第一行书写。

旧版的 HTML 4.01 中，此声明有三种形式：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
    "http://www.w3.org/TR/html4/loose.dtd">  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
    "http://www.w3.org/TR/html4/strict.dtd">  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"  
    "http://www.w3.org/TR/html4/frameset.dtd">
```

其中，第一种是过渡模式；第二种是严格模式；第三种等同于第一种，但允许使用框架集内容。在 HTML5 版本中，此声明写作：

```
<!DOCTYPE html>
```

2014年10月29日,万维网联盟宣布HTML5标准最终制定完成,在这之前大部分浏览器就已经支持了某些HTML5的特性。HTML5的应用必然会越来越广泛,因此,DOCTYPE的声明尽可能写成HTML5的形式,而且此形式也最简洁。

2. 头部

`<head>`是头部元素的标签,其包含的信息用于告诉浏览器文档的标题、页面的编码、脚本文件和样式文件的引用地址等。

- title

定义文档的标题,此标题包括:浏览器标签中的标题、收藏页面时的默认标题、显示在搜索引擎结果中的标题。使用方法为:

```
<title>宋词</title>
```

- meta

定义元数据的信息。`<meta>`没有结束标签,其用法是通过 `http-equiv` 或 `name` 指定信息的种类,再用 `content` 来定义此种类的内容,常见的有:

```
<meta http-equiv="content-type"
      content="text/html; charset=gb2312" />

<meta name="description" content="HTML Document" />

<meta name="keywords" content="HTML, CSS, JavaScript, D3" />
```

第一行告诉浏览器文档的类型为HTML,使用的编码为GB2312。第二行告诉搜索引擎网页的主要内容是什么。第三行告诉搜索引擎网页的关键词是什么。

- link

用于引用外部资源,如引用样式表:

```
<link rel="stylesheet" type="text/css" href="style.css" />
```

- style

用于在文档中定义CSS样式,如果数量很多,则建议写在外部文件中,再用`<link>`引用。如果数量较少,有时为了便于查看HTML元素和CSS样式的关系,可将其在`<style>`中定义。

- script

用于定义客户端脚本,最常见的是JavaScript脚本,目前已经成为事实上的标准语言。

3. 属性

标签可以拥有属性,表示如下:

```
name="value"
```

属性名称不加引号，属性值通常加双引号，也可以加单引号。前面出现的 `rel="stylesheet"` 和 `name="keywords"` 都是属性值。

4. 主体

主体由 `<body>` 标签定义，`<body>` 里包含着各种元素标签，用于表示文档的内容。大多数 HTML 元素标签被分为块级元素和内联元素。

块级元素在显示时会以新行表示，常见的如下所示。

- `h1`、`h2`、`h3`、`h4`、`h5`、`h6`

定义标题，`h1` 是最大的标题，`h6` 是最小的标题。

- `p`

定义段落，末尾会自动换行。

- `ul`、`ol`、`li`

定义列表，`ul` 用于无序列表，`ol` 用于有序列表，内部再嵌套 `li`，每行使用一个。无序列表使用小黑圆圈表示，有序列表使用数字表示。定义一个两行的无序列表如下：

```
<ul>
<li>张三</li>
<li>李四</li>
</ul>
```

- `table`

表格，其中每行由 `<tr>` 定义，每列由 `<td>` 定义。定义一个两行两列的表格如下：

```
<table border="1">
  <tr>
    <td>第一行，第一列</td>
    <td>第一行，第二列</td>
  </tr>
  <tr>
    <td>第二行，第一列</td>
    <td>第二行，第二列</td>
  </tr>
</table>
```

- `div`

定义文档中的分区或节，内部包含任意其他元素。

内联元素在显示时不以新行表示，常见的如下所示。

- `a`

超链接的标签，点击后跳转到目标页面，通过 `href` 来指定 URL 路径。

```
<a href="http://www.ourd3js.com/">数据可视化</a>
```

➤ img

图片，通过 src 来指定图片 URL 地址，alt 设定当图片无法加载时显示的文字。

```

```

➤ span

用于组合行内元素，添加在行内元素之间。

5. 注释

注释标签在浏览器中不会显示，程序员自己可适当添加以便阅读。添加的方法为使用<!-- 和-->将内容包含在里面。

```
<!-- <p>This is a dog. -->
```

2.2.2 CSS 选择器

如果要给两个段落定义相同的样式，该如何做呢？

```
<p style="color:red;background-color:yellow;font-size:22px;">  
  红藕香残玉簟秋，轻解罗裳，独上兰舟</p>  
  
<p style="color:red;background-color:yellow;font-size:22px;">  
  云中谁寄锦书来，雁字回时，月满西楼</p>
```

上述代码将两个段落都设置成红色字体、黄色背景、22px 大小的字体。目的虽然达到了，但十分冗长。如果有更多的段落需设置成这样呢？这样的代码不仅不利于阅读，而且当希望将字体颜色改变成黑色时，需要改很多地方，非常不便。

定义一个 CSS 选择器，然后在段落标签中应用该选择器，即可达到目的。

```
.pstyle {  
  color: red;  
  background-color: yellow;  
  font-size: 22px;  
}
```

在<p>元素中应用选择器。

```
<p class="pstyle">红藕香残玉簟秋，轻解罗裳，独上兰舟</p>  
<p class="pstyle">云中谁寄锦书来，雁字回时，月满西楼</p>
```

是否感觉心情愉快了许多，将来如果想要修改字体、颜色等，只需在选择器中修改即可，

不需要改动 HTML 代码，极大地提高了效率。要记住，HTML 只应用来描述网页的内容，如“标题是什么”、“段落是什么”；而如何表现这些内容，如“字体颜色”、“字体大小”等都使用 CSS。除非必要，否则不要在 HTML 元素中添加大量的属性来描述如何表现元素。

1. 语法

CSS 由两部分组成：选择器的名称和“属性名称-属性值”。

```
selector {  
    name1: value1;  
    name2: value3;  
    name3: value3;  
}
```

属性名称与属性值之间用冒号隔开，每一对值之间加分号隔开。冒号之后建议加一空格，有利于阅读，空格不会影响效果。如果属性值中间有空格，需要给值加引号：

```
h1 { font-family: "sans serif"; }
```

2. 选择器

一个 CSS 选择器能对应一个元素，也能对应多个元素，也可能多个选择器对应一个元素。要达到不同的对应方式，选择器有如下数种类型。

- 元素选择器

以 HTML 元素的标签作为名称。例如：

```
p { color: blue; }
```

则所有段落的文字都会变为蓝色。

- 选择器分组

如果几个选择器的样式相同，可用逗号分隔：

```
h1, h2, p {  
    color: blue;  
}
```

则<h1>、<h2>、<p>三个标签都将文字变为蓝色。

- 类选择器

在选择器名称前加一个点(.)，表示是类选择器：

```
.important { color: red; }
```

要应用此类，在元素标签里添加一个 class 属性，

```
<p class="important">段落</pan>
```

如果此类选择器会应用在很多标签里，而你只希望在段落的标签里才应用此选择器，可：

```
p.important { color: red; }
```

要注意，`p` 和 `important` 之间没有空格，如果加了空格则变为派生选择器。

- ID 选择器

如果希望某个特定的元素具有某种样式，可以使用 ID 选择器。与类选择器不同的是，ID 选择器在文档中只使用一个，但是即便多次使用，很多浏览器也能解读。如此，就变得很像类选择器，建议不要使用多次。

```
#index { font-weight: bold; }  
<p id="index">Index 1</p>
```

在选择器名称前加 `#` 号，在元素属性中添加 `id` 属性即可，属性值不需要加井号 (`#`)。

- 派生选择器

派生选择器又分为三种。如果希望 `p` 中的 `span` 元素应用样式，可在 `p` 与 `span` 之间加一空格，这个称为后代选择器。

```
p span { color: red; }
```

如果只希望选择 `p` 的直系子元素 `span`，则在 `p` 与 `span` 之间加一个大于号 (`>`)，这个称为子元素选择器。

```
p > span { color: red; }
```

如果希望选择紧接在 `h1` 元素之后的一个 `p` 元素，且 `h1` 和 `p` 拥有共同的父元素，则在 `h1` 和 `p` 之间加一个加号 (`+`)，这个称为相邻兄弟选择器。

```
h1 + p { color: red; }
```

3. 属性名称和属性值

属性名称和属性值是成对出现的，CSS 中有很多属性名称，最常见的属性如下。

- 尺寸

`width`: 元素宽度。

`height`: 元素高度。

- 背景

`background-color`: 背景颜色。

`background-image`: 背景图像。

`background-position`: 背景图像的起始位置。

`background-repeat`: 背景图像是否及如何重复。

- 文本

`color`: 文本颜色。

`line-height`: 行高。

text-align: 对齐元素中的文本。

- 字体

font-family: 字体。

font-size: 字体尺寸。

font-style: 字体风格。

font-weight: 字体粗细。

- 边框

padding: 内边距。

border: 边框。

margin: 外边距。

- 定位

position: 元素是静态的、相对的、绝对的，还是固定的。

top: 到上边界的距离。

right: 到右边界的距离。

bottom: 到下边界的距离。

left: 到左边界的距离。

float: 浮动。

clear: 清除浮动。

2.2.3 综合运用

HTML 用于描述“内容是什么”，CSS 用于描述“如何表现此内容”。下面举一个例子，综合使用 HTML 和 CSS，来实践前面的内容。制作网页时，首先要明确所需的网页布局，假设布局如图 2-3 所示。

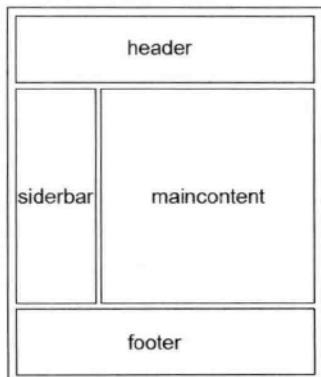


图 2-3 网页布局（各区域名称是 div 的 id 号）

网页被划分为四块区域。可用 div 来定义各区域，HTML 结构如下：

```
<div id="container">
  <div id="header">
  </div>
  <div id="mainbody">
    <div id="siderbar">
    </div>
    <div id="maincontent">
    </div>
  </div>
  <div id="footer">
  </div>
</div>
```

分别用 id 属性来标示各区域，其中 siderbar 和 maincontent 被包含于 mainbody 中，最后再将所有区域块都放入 container 区域块里，方便操控。区域块明确之后，首先在 HTML 里添加 h1、p、ul 等内容标签，然后再为各区域块定义样式。如图 2-4 所示，左图是未添加 CSS 样式之前的模样，可以看到这是纯粹的内容，区域块也没有如图 2-3 所示那样布局。右图是添加了 CSS 样式之后，已经变成了目标布局，文字也美观了不少。

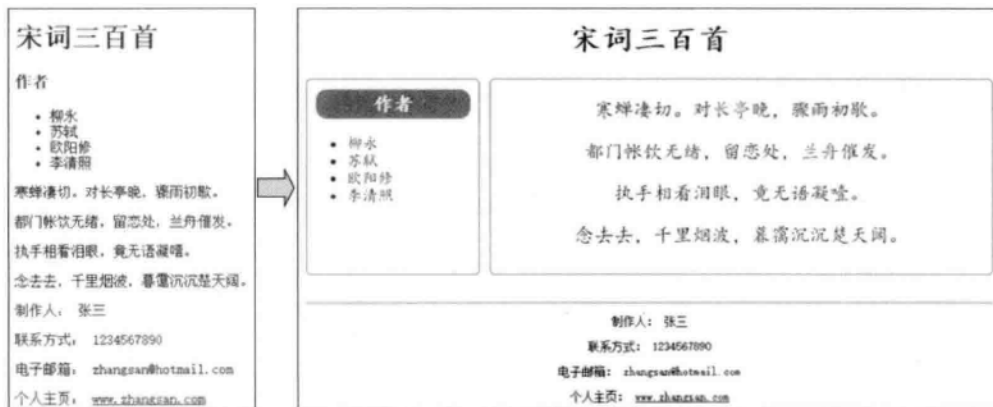


图 2-4 HTML 的内容（左）加上 CSS 样式之后的效果（右）

图 2-4 的布局中，siderbar 和 maincontent 是使用浮动属性布局，这种布局被应用在很多网站上。为 HTML 元素添加内容后，代码如下：

```
<div id="siderbar">
<h3 class="category">作者</h3>
  <ul>
    <li>柳永</li>
    <li>苏轼</li>
```

```
<li>欧阳修</li>
<li>李清照</li>
</ul>
</div>
<div id="maincontent">
  <p>寒蝉凄切。对<span class="important">长亭</span>晚，骤雨初歇。</p>
  <p>都门帐饮无绪，留恋处，<span class="important">兰舟</span>催发。</p>
  <p>执手相看泪眼，竟无语凝噎。</p>
  <p>念去去，千里烟波，暮霭沉沉楚天阔。</p>
</div>
```

使用 h3、ul、li、p、span 添加了内容，要注意这些标签中都没有包含样式属性，样式都要在 CSS 选择器中定义。接下来讲解 sidebar 和 maincontent 这两个 id 的样式。

```
#sidebar{
  width: 25%;
  height: 100%;
  float: left;
  clear: left;
  border: 1px solid gray;
  border-radius: 5px;
}

#maincontent{
  width: 73%;
  height: 100%;
  float: right;
  clear: right;
  text-align: center;
  border: 1px solid gray;
  border-radius: 5px;
  font-size: 20px;
}
```

width 和 height 是区域块的宽度和高度，这里使用百分比，表示父元素宽高的百分之多少。float 分别设定为 left 和 right，表示前者向左浮动，右者向右浮动。clear 表示清除浮动。border 用于定义区域块的边框，三个值依次为宽度、样式、颜色。border-radius 用于定义圆角边框，值为圆角的半径。font-size 用于设定字体大小。

2.3 JavaScript

JavaScript 是 D3 的开发语言，使用 D3 时会涉及很多 JavaScript 的概念，因此基础知识的掌

握是必要的。JavaScript 的语法并不复杂，与 C/C++、Java 很相似，只是记住语法可能只需要半个月，但想要得心应手地使用可能需要数年。本节仅介绍学习 D3 所必需的基础知识。

2.3.1 在 HTML 中使用 JavaScript

在 HTML 文档中通过<script>标签来使用 JavaScript。如果要在 HTML 里写 JavaScript 代码，则：

```
<script type="text/javascript">
console.log("Hello, World");
</script>
```

属性 type 用于设定脚本语言的类型，除了 text/javascript 之外，还可设定为 text/ecmascript、text/vbscript 等脚本语言。但 JavaScript 已成为事实上 Web 浏览器的标准语言，因此可不设定这个值。如果忽略，则默认为 JavaScript 语言。上面的代码输出“Hello, World”字符串，函数 console.log() 能够将文本输出到控制台。

如果 JavaScript 是写在一个单独的文件里的，那么引用方法为：

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

通过 src 属性来指向外部链接文件，此处指向了 d3.v3.min.js 的源文件，D3 的所有代码就是写在这个文件里的。有一点要注意，浏览器会按照<script>出现的顺序依次加载，一个完了再加载另一个。如果<script>写在 HTML 的<head>里，加载的文件过多过大，则在加载的时间里用户看到的是一片空白。因此，<script>也可以放到<body>中所有的内容元素之后，让浏览器先显示内容，再加载脚本。JavaScript 文件的后缀名通常带有.js 扩展名，但并不是必需的，浏览器不会检查扩展名。

2.3.2 语法

1. 区分大小写

变量名、函数名是区分大小写的，因此 name 和 Name 是不同的。第一个字符必须是字母、下划线（_）或美元符号，后面的字符可以有数字。

2. 注释

注释分为单行注释和多行注释，注释方法与 C/C++ 一样：

```
// var name = "zhangsan";      单行注释
```

```
/*                      多行注释
    var a = 10;
    var b = a;
*/
```

3. 分号

每条语句结尾添加分号 (;), 虽然不添加浏览器也能解读, 但容易造成意想不到的错误, 因此建议都加上分号。

4. 花括号

花括号的方法建议写成:

```
function getName(person){
    return person.name;
}
```

而不是像 C 语言一样写成:

```
function getName(person)
{
    return person.name;
}
```

虽然两种都不报错, 但每种语言都有一些不成文的规范, 只有符合规范代码才能被更多人接受。

2.3.3 变量

定义变量时使用 `var` 操作符, 后跟变量名, 多个变量名之间用逗号隔开。如下所示:

```
var a;           //undefined
var b,c,d;       //undefined
```

如果定义变量时不为其赋值, 则默认值为 `undefined`, 表示该值未定义。为防止出现意外后果, 通常会在定义时为其赋值。可以在一个 `var` 后对多个变量赋值, 各变量间用逗号隔开:

```
var a = 10;
var b = 20,
    c = 25,
    d = 30;
```

JavaScript 的变量是松散类型的, 所有变量的定义都是使用 `var`, 既可以是数值, 也可以是布尔型 (`true` 或 `false`), 还可以是字符串。并且, 即便一个变量初始值为数值, 仍然可以将其他

类型的值赋给它:

```
var a = 26;
var b = false;
var c = "message";
a = "box";
```

a 初始值为 26, 但仍可用字符串 box 为其赋值, 这与 C 语言不同。虽然如此, 变量的类型最好自始至终都是确定的, 不建议轻易更改。

2.3.4 数据类型

JavaScript 有五种基本数据类型: undefined、null、boolean、number、string。还有一种复杂数据类型: object。变量属于哪一种数据类型可用 typeof 查看, 如下所示:

```
var a = 26;
console.log(typeof a); //number
```

a 变量中存储的是数值 26, 属于数值(number)类型, 因此使用 typeof 测定后, 输出了 number。typeof 是 JavaScript 的一个关键字。下面分别介绍六种数据类型。

1. undefined

未初始化的变量默认值都是 undefined, 表示该值未定义。如果某变量将来要为其赋值为其他基本数据类型之一, 那么将其初始化为 undefined 较好:

```
var c = undefined;    //初始为undefined
c = 10;               //将来将其赋值为number(数值)类型的10
```

2. null

与 undefined 十分类似, 不同之处在于: null 表示一个空对象、undefined 表示一个未定义的基本类型的变量。如果某变量将来要为其赋值为 object 类型, 将其初始化为 null 较好。

```
var o = null;
o = { name:"zhangsan", age:19 };
```

如果对一个赋值为 null 的变量用 typeof 检测, 结果是 object:

```
var o = null;
console.log(typeof o); //object
```

这其实也证明了 null 适合用于表示空对象, 因此, 当不需要使用一个对象, 将其清空时, 可赋值为 null。赋值之后, 其类型仍然为 object, 表示这个变量还是对象, 只是被清空了而已。


```
var e = { name:"lisi", age:20 };
e = null;
console.log(typeof e); //object
```

3. boolean

布尔类型有两个值：`true` 和 `false`，表示“真”和“假”两种状态。在条件语句（如 `if`）中，其他类型的值会自动转换为布尔型。有五种值会转换为 `false`：`0`、`NaN`、`undefined`、`null`、`""`（空字符串），其他值均转换为 `true`。那么在进行条件判断时，即可使用如下语句：

```
var a = 10;
if(a){
    console.log("数值不是0或NaN");
}
```

4. number

数值可使用十进制数、八进制数、十六进制数，八进制数以零（0）开头，十六进制数以 `0x` 开头：

```
var num_10 = 120;           //十进制数120
var num_8 = 020;            //八进制数20，相当于十进制数16
var num_16 = 0x3A;          //十六进制数3A，相当于十进制数58
```

浮点数值用小数点来表示，较大或较小的数可用指数来表示：

```
var f = 3.1415;             //浮点数
var m = 3e5;                //指数形式，表示  $3 * 10^5$ 
```

JavaScript 的数值类型有范围。最大数值为 `Number.MAX_VALUE`，这个值通常为 `1.7976931348623157e+308`；最小数值为 `Number.MIN_VALUE`，这个值为 `5e-324`。如果超出此范围，正数则返回 `Infinity`（正无穷），负数则返回 `-Infinity`（负无穷）。另外，正数除以 `0` 返回正无穷，负数除以 `0` 返回负无穷。

数值类型还有一个特殊的值 `NaN`，是 `Not a Number` 的意思，表示不是一个数。

5. string

字符串类型可由双引号或单引号表示：

```
var str_1 = "China";
var str_2 = 'America';
```

字符串的长度可用 `length` 得到：

```
console.log(str_1.length); //5
```

两个字符串拼接可用加号 (+) 实现:

```
console.log(str_1 + str_2);    //ChinaAmerica
```

6. object

对象 object 是拥有属性和方法的数据类型。属性是与对象相关的值，方法是在对象上执行的动作。现实生活中，楼房可以是对象，人也可以是对象。下面的做法就创建了一个对象。

```
var person = new Object();
```

但是，这个对象没有属性，也没有方法。下面为其添加属性和方法：

```
person.name = "WangWu";
person.age = 20;
person.growUp = function(){
    this.age += 1;    //年龄增加1岁
}
```

这段代码增加了两个属性 name 和 age，还有一个方法 growUp()。接下来试着访问这些属性和方法：

```
console.log( person.name );    //WangWu
console.log( person.age );    //20
person.growUp();
console.log( person.age );    //21
```

给方法命名的方式通常有两种。一种是空格之后单词的首字母大写，即 growUp()；另一种是空格用下划线代替 (_)，字母全用小写，即 grow_up()。

2.3.5 操作符

操作符用于数值的运算、比较、判断等操作，具体有以下几类。

1. 算术操作符

算术操作符即加减乘除等操作，是最常用的操作符。

```
var num1 = 5 + 3;    //加法，结果为8
var num2 = 5 - 3;    //减法，结果为2
var num3 = 5 * 3;    //乘法，结果为15
var num4 = 5 / 3;    //除法，结果为1.6666666666666667
var num5 = 5 % 3;    //求余，结果为2
num1++;              //递增，结果为9
num2--;              //递减，结果为1
```

2. 赋值操作符

算术运算符和等号配合使用，能减少代码的书写量：

```
var a = 18;           //赋值，使用等号(=)
a += 3;               //等同于 a = a + 3; 结果为21
a -= 2;               //等同于 a = a - 2; 结果为19
a *= 4;               //等同于 a = a * 4; 结果为76
a /= 2;               //等同于 a = a / 2; 结果为38
a %= 7;               //等同于 a = a % 7; 结果为3
```

3. 布尔操作符

布尔操作符共有三个：非 (!)、与 (&&)、或 (||)。

```
var isDog = false;
var isAnimal = true;
console.log( !isDog );           //true
console.log( isDog && isAnimal ); //false
console.log( isDog || isAnimal ); //true
```

4. 关系操作符

关系操作符即大于 (>)、小于 (<)、大于等于 (>=)、小于等于 (<=)、相等 (==)、全等 (===)、不相等 (!=)、不全等 (!==) 这八种。

```
console.log( 8 > 7 );           //true
console.log( 8 >= 7 );          //true
console.log( 8 < 7 );           //false
console.log( 8 <= 7 );          //false
console.log( "8" == 8 );        //两者转换后相等，结果为true
console.log( "8" === 8 );       //两者转换后不相等，false
console.log( "7" != 7 );        //两者转换后相等，false
console.log( "7" !== 7 );       //两者转换后不相等，true
```

特别要注意“相等”和“全等”的区别，相等操作符会自动对数据类型进行转换，全等则不会。

5. 条件操作符

条件操作符只有一个：

```
var result = 5 > 3 ? true : false; //true
```

如果问号前的条件为 true，则返回冒号之前的值；如果为 false，则返回冒号之后的值。

2.3.6 语句

语句用于流程控制，实现条件判断、循环等功能。

1. if-else 语句

最常见的条件判断语句，可以单独使用 if，也可以添加 else if 和 else。

```
if( 20 > 33 ){
    console.log("大于");
}else if( 20 === 33 ){
    console.log("等于");
}else{
    console.log("小于");    //符合这个条件
}
```

2. while 和 do-while 语句

while 语句是先测试条件再循环，do-while 语句是先循环再判断条件。

```
var i = 0;
while(i<5){    //先判断
    i++;        //再循环
}

var j = 0;
do{
    j++;        //先循环
}while(j<10)   //再判断
```

3. for 和 for-in 语句

for 是一种功能更强大的循环语句，紧跟的括号里用两个分号分隔成三个部分，第一部分用于值的初始化，第二部分用于条件判断，第三部分用于求新值：

```
for(var i=0;i<5;i++){
    console.log(i);    //0,1,2,3,4
}
```

for-in 主要用于枚举对象属性和方法，如对于以下对象：

```
person.name = "WangWu";
person.age = 20;
person.growUp = function(){
    this.age += 1;    //年龄增加1岁
}
```

```
}
```

则可写代码如下:

```
for(var prop in person){  
    console.log(prop);    //输出name,age,growup  
}
```

4. switch 语句

与 if 类似,用于条件判断。switch 适于判断的情况是,如果某变量是这个值的情况下,执行某操作,如果是那个值,执行另一操作。每种情况最后要用 break 结尾,防止继续往下判断。

```
var i = 20;  
switch(i){  
    case 10:    //判断过此值,不符合  
        console.log(10);  
        break;    //用break跳出switch  
    case 20:    //判断过此值,符合  
        console.log(20);    //输出20  
        break;    //用break跳出switch  
    case 30:    //没有判断过此值  
        console.log(30);  
        break;  
    default:    //如果都不符合,默认怎么处理  
        console.log("都不符合");  
}
```

5. break、continue 和 label 语句

break 语句可退出循环,不再判定下次执行循环的条件;continue 只是退出本次循环,还会继续判定下次是否执行循环:

```
for(var i=0;i<3;i++){  
    if( i === 1)  
        break;  
    console.log(i);    //只输出0  
}  
  
for(var i=0;i<3;i++){  
    if( i === 1)  
        continue;  
    console.log(i);    //输出0和2  
}
```

但是, break 和 continue 都只能跳出当前花括号的循环。如果要跳出多重循环,会比较麻烦。

因此，可以配合 `label` 使用，用于跳出多重循环。`label` 标记在循环之前：

```
fori: for(var i=0;i<3;i++){
    for(var j=0;j<3;j++){
        if( i == 1 && j == 0 )
            break fori;
        console.log(i+"\t"+j);
    }
}
```

`fori` 即最外层循环的 `label` 名称，在 `break` 之后加上此名称即可指定跳出哪个循环。

2.3.7 函数

JavaScript 的函数使用 `function` 关键字来声明，后跟参数及函数体。与 C/C++ 或 Java 等不同，JavaScript 中函数的参数不需要指定数据类型。其基本语法如下：

```
function add(num1, num2){
    return num1 + num2;
}
```

上面定义了一个函数名为 `add` 的函数，函数体里执行的是两个数相加并返回。然后即可使用此函数名来调用函数：

```
var a = add(3,5);           //a的值为8
```

还有一种定义函数的方法：

```
var add = function(num1,num2){
    return num1 + num2;
}
```

表面上看是将一个匿名函数 `function` 赋值给了一个变量 `add`，这种方式在后续 D3 的学习中会经常见到。

2.3.8 对象

在 2.3.4 节里说过一些关于对象的内容，提供了一种创建对象的方式。除此之外，还有一些常见的创建对象的方法：

```
var person = {
    name: "WangWu",           //两个属性之间要添加逗号
    age: 20,
```

```
growUp: function(){
    this.age += 1;
}           //如果之后还有属性，这里也要添加逗号
}
```

还有一种方法，可制作一个构造函数，然后即可使用 `new` 来创建对象，对于习惯了 C 或 Java 的人可能会感到很开心：

```
function Person(name,age){
    this.name = name;
    this.age = age;
    this.growUp = function(){
        this.age += 1;
    }
}
var wangwu = new Person("WangWu",18);
```

调用对象的属性有两种方法，都很常用：

```
console.log(wangwu.name);           //小圆点
console.log(wangwu["name"]);        //方括号
```

通常使用小圆点的方式，但方括号的方式也很重要，有一些场合只能使用这种方式。另外，使用方括号的方式调用函数也是可以的：

```
wangwu["growUp"]();
```

2.3.9 数组

数组是常用的数据结构，用一对方括号即可定义。JavaScript 的数组其实是一种对象。使用 `typeof` 检测，可知数组是 `object` 类型：

```
var a = [1,2,3,4,5];
console.log(typeof a); //object
```

同一数组里还可以是不同类型的值：

```
var a = [1,"apple",false,4,5];
```

还有一种使用 `Array` 构造函数创建数组的方法：

```
var a = new Array(1,2,3,4);    //长度为4，数组项分别为1,2,3,4的数组
var b = new Array(20);         //长度为20，数组项还没有赋值的数组
```

通过这种方式，更能清楚地看到数组是对象。因此可以想象，数组里的各项其实就是这个

Array 对象里的属性，属性的名称分别为 0、1、2、3（数组的序号）等。那么能否使用如下方式调用呢？

```
console.log(a.1); //不能得到结果
```

结果出错。前面说过，JavaScript 的变量名是不能以数字开头的，因此数组里的属性名称 0、1、2、3 等可看作是特例，是数组专用的。第 2.3.8 节中提到调用对象的方式除了使用小圆点之外，还可使用方括号的方式：

```
console.log(a["1"]); //得到正确的结果
```

这样就证明了数组项其实是对象的属性，数组项的序号实际上是属性的名称。这一点对于理解数组是什么非常重要。

另外，数组的长度可用 `length` 来获取。`length` 属性是可以赋值的，赋值之后，超出此长度的部分就不能使用了。

```
var city = ["Beijing", "Shanghai", "Guangzhou"];
console.log(city.length); //3
console.log(city); //["Beijing", "Shanghai", "Guangzhou"]
city.length = 1; //将长度赋值为1
console.log(city); //["Beijing"]
```

给数组添加项，可以直接给指定序号的项赋值：

```
city[10] = "GuiLin";
console.log(city.length); //11
```

其实，从序号 3~9 都是没有项的，但是 `length` 的值却为 11。`length` 总是比最大的序号大 1（数组序号从 0 开始），与中间有多少项无关。

给数组添加和删除项的方法，还可使用以下函数。

- `push`: 在末尾添加项。
- `pop`: 将末尾项删除并返回。
- `shift`: 将第一项删除并返回。
- `unshift`: 从最前面推入项。

2.4 DOM

DOM，指文档对象模型（Document Object Model），是针对结构化文档的一个接口，它允许程序和脚本动态地访问和修改文档。其中，针对 HTML 的模型称为 HTML DOM。换言之，使用这套模型即可任意访问和修改 HTML 元素。D3 中的函数大量脱胎于 DOM，因此了解 DOM

的基本原理对于理解 D3 有很大的帮助。

2.4.1 结构

DOM 是以树形结构来描述 HTML 文档的，其被称为节点树。每个 HTML 元素都是树上的一个节点，节点之间的关系如图 2-5 所示。

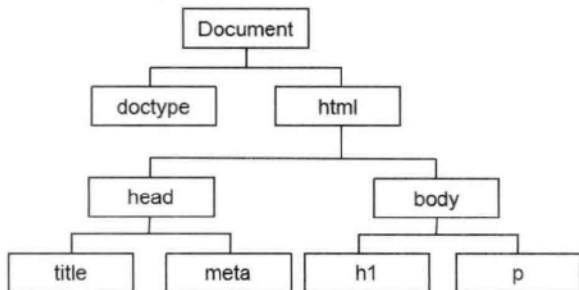


图 2-5 HTML 文档的树形结构

html 是 body 的父节点 (parent)，body 是 html 的子节点 (child)，body 是 head 的同胞节点 (sibling)。

2.4.2 访问和修改 HTML 元素

DOM 的文档对象保存在 document 中，调用其方法或属性，即可访问 HTML 文档中的任意元素。访问方式有：

```
document.getElementById("myid");           //返回id为myid的元素
document.getElementsByTagName("p");         //返回所有标签为p的元素
document.getElementsByClassName("myclass");//返回类为myclass的元素
```

这三种方法返回所指定的元素。下面以 `getElementsByTagName` 为例来说明其用法：

```
<p>Apple</p>
<p>Banana</p>
<script>
  var para = document.getElementsByTagName("p");
  for( var i=0; i<para.length; i++){
    console.log( para[i].innerHTML );
  }
</script>
```

上面的代码使用 `getElementsByTagName` 获取了文档中所有的 `p` 元素，返回值保存在变量 `para` 中。`para` 是一个数组，保存了所有的 `p` 节点。然后，在 `for` 循环中输出了 `para` 的内容，最终结果是在控制台中输出 `Apple` 和 `Banana` 两行文字。这里使用了 `innerHTML` 属性，这个属性是能够对其赋值的。例如将上述 `for` 循环改成：

```
for( var i=0; i<para.length; i++ ){
    para[i].innerHTML = "Pear";
}
```

则两个 `p` 元素的内容均被修改为 `Pear` 了。在 HTML DOM 中，常用的属性如下。

- `innerHTML`: 元素标签内部的文本，包括 HTML 标签。
- `innerText`: 元素标签内部的文本，但不包括 HTML 标签。
- `outerHTML`: 包括元素标签自身在内的文本，也包括内部的 HTML 标签。
- `outerText`: 包括元素标签自身在内的文本，但不包括 HTML 标签。
- `nodeName`: 节点名称。
- `parentNode`: 父节点。
- `childNodes`: 子节点。
- `nextSibling`: 下一个同胞节点。
- `previousSibling`: 上一个同胞节点。
- `style`: 元素的样式。

使用 `style` 属性，即可修改元素的样式，如下代码所示：

```
var body = document.getElementsByTagName("body");
console.log(body);
body[0].style.backgroundColor = "blue";
```

这段代码将 `body` 元素的背景颜色修改成了蓝色。要注意，属性名称 `backgroundColor` 与 CSS 中的名称 `background-color` 是不同的，因为变量名中不允许拥有减号 (-)。故其对应名称是去掉减号后，将下一个单词的首字母改成大写即可。

2.4.3 添加和删除节点

给文档中添加节点可使用 `appendChild()` 方法，每个元素节点中都有这个方法，能在该元素的末尾添加子节点。为此，首先要创建一个节点，然后才能添加。请看如下代码：

```
var para = document.createElement("p");           //创建节点p
para.innerHTML = "Hello";                          //给p的内容赋值
var body = document.getElementById("mybody");      //获取body节点
body.appendChild(para);                            //给body节点添加子节点
```

还可以删除文档中的节点:

```
<body id="mybody">
  <p id="mypara">Apple</p>
  <p>Banana</p>
  <script>
    var para = document.getElementById("mypara");
    var body = document.getElementById("mybody");
    body.removeChild(para);
  </script>
</body>
```

上面的代码中, 首先获取了 id 为 mypara 的元素 p 节点, 然后获取了 body 元素节点, 再使用 body 中的 removeChild() 方法删除掉 p 节点。最终结果是页面中只有一个段落, 文字为 Banana。

2.4.4 事件

HTML DOM 通过事件能够与用户进行交互, 如鼠标单击、鼠标移入、页面加载等。先看如下代码:

```
<p id="mypara">Click Here</p>
<script>
  var para = document.getElementById("mypara");
  para.onclick = function(){
    this.innerHTML = "Thank you";
  }
</script>
```

上述代码为选择的 p 元素定义了一个事件 onclick, 当用户单击该元素时, 便调用 onclick 中的匿名函数 function。该函数的内容是将段落文字更改为 Thank you。像这样的事件有很多, 常用的如下。

- onload: 页面或图片加载完成时。
- onclick: 鼠标单击。
- ondblclick: 鼠标双击。
- onkeydown: 键盘某个按键按下。
- onkeypress: 键盘某个按键按下并松开。
- onkeyup: 键盘某个按键松开。
- onmousedown: 鼠标按钮按下。
- onmousemove: 鼠标移动。
- onmouseout: 鼠标从某元素移开。

提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ1779903665.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签索引和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我QQ1779903665。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，与本人无关，我们仅仅只是帮助你寻找到你要的pdf而已。