Broadview
www.broadview.com.cn

代码速查

始于功能，终于代码，拿来即用
语法、规范、理念、工具、技巧，尽在超实用代码段中

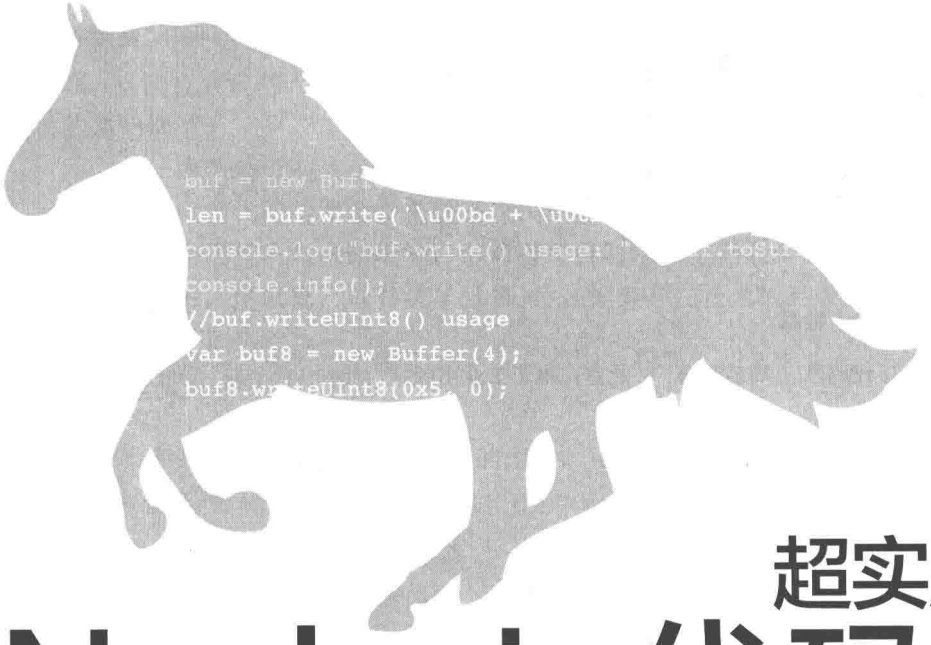
超实用的 Node.js代码段

周敏 编著

基于实践和学以致用的原理，全是干货。涵盖控制台、模块、包、异步I/O、Async流程控制、进程通信、TCP/UDP网络编程、流、Web开发、MySQL、MongoDB……

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

·代码逆袭·



超实用的 Node.js代码段

周敏 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书精选 300 余段 Node.js 代码，涵盖了服务器端脚本开发中的绝大多数要点、技巧与方法，堪称史上最实用的 Node.js 框架开发方面的参考书籍，是网站建设与服务器端开发人员的好帮手。本书的代码跨平台、跨设备、跨浏览器，充分向读者演示了如何使用 Node.js 框架的各项技术。

本书从 Node.js 框架的使用原理与应用场景出发，对最实用的 Node.js 代码段进行了全方位的介绍和演示。全书分为 15 章，包含控制台、模块和包管理、异步 I/O 与 Async 流程控制库、Buffer、进程管理、子进程通信、OS 操作系统、文件系统、路径处理、TCP/UDP 网络编程、流（Stream）、Web 开发、常用工具及 MySQL 与 MongoDB 数据库交互等 Node.js 框架技术的内容，对提高网站建设与服务器端开发人员的 Node.js 技术水平有着非常重要的指导作用。

本书内容简洁明了、代码精练、重点突出、实例丰富、语言通俗易懂、原理清晰明白，是网站建设与服务器端开发人员的良好选择，同时也非常适合大中专院校学生学习阅读。

图书在版编目（CIP）数据

超实用的 Node.js 代码段 / 周敏编著. —北京：电子工业出版社，2015.11
（代码逆袭）
ISBN 978-7-121-27431-2

Ⅰ. ①超… Ⅱ. ①周… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2015）第 249419 号

责任编辑：董　英
印　　刷：北京中新伟业印刷有限公司
装　　订：北京中新伟业印刷有限公司
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：787×1092　1/16　　印张：22.75　　字数：553 千字
版　　次：2015 年 11 月第 1 版
印　　次：2015 年 11 月第 1 次印刷
印　　数：3000 册　　定价：59.00 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到 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

近些年 Node.js 框架作为一种服务器端脚本语言的开发技术，在 IT 圈内可谓是刮起了一股热旋风。因为大家发现，原来仅仅运行于浏览器端的 JavaScript 脚本也可以运行在服务器端，这简直太震撼人心了。于是，学习和掌握 Node.js 框架开发技术成了众多开发设计人员的热切期望。

本书是一本讲解代码实践的书，它为读者全面深入地讲解了针对各种场景的 Node.js 技术。300 余段的代码给读者带来的不仅仅是全面的基础知识，更是为读者提供了设计简洁高效的服务器端代码与网站架构、应对跨平台与跨浏览器兼容、优化服务器性能等切实问题的解决之道。可以说，本书是学习 Node.js 框架开发技术的高效助手。

Node.js 框架中的那些事儿

对于大多数刚刚接触 Node.js 框架的读者，可能一时无从下手，那么 Node.js 技术的难点都有哪些方面呢？

- 跨平台与浏览器的兼容
- 正确理解服务器端脚本使用
- 模块和包管理
- 异步 I/O 与 Async 流程控制库
- 进程与子进程通信
- 文件与路径处理
- Node.js 事件处理机制
- TCP/UDP 网络编程
- HTTP/HTTPS 应用编程
- Node.js 与 AJAX 应用
- Node.js 与数据库交互

以上所有内容在本书的代码中都有讲解，除了这些常见 Node.js 技术难点外，本书还力求将最有用的 Node.js 代码段汇总在一起，提供各种解决实际问题的方案。

Node.js 框架学习方法

11 个字就能帮助我们更好地学习 Node.js。

- **多看、多练**：观摩成功的网页设计，分析并练习网页设计中常用的代码。
- **多想、多问**：思考设计实现的原理，提出自己的问题并通过各种渠道来寻找答案。
- **多总结**：记录前人已经探索出来的 Node.js 技巧，总结实战中碰到的问题及解决方案。

只要真正能做到勤思考、勤动手、勤总结，Node.js 学习定能一马平川。

本书的编写特点

- **独特的 Node.js 切入点**
  与市面上其他 Node.js 相关的书不同，本书从最常见的场景应用角度出发，直接应用 Node.js 代码段实现功能操作，全部是最实用的例子，全部是最透彻的分析！
- **内容丰富，知识全面**
  本书以 Node.js 框架各个模块的场景应用作为基础，立体式、全方位地解释各种场景下的 Node.js 代码段应用，场景丰富、实例丰富，并拥有良好的可扩展性、可复用性。
- **去中心化，分布式学习**
  本书的代码实例都是独立的，读者可以从中间开始学，也可以从头开始学。代码跨平台、跨设备，可以在平板电脑上学，也可以在 PC 上学，甚至，手机上写代码也是完全可能的。
- **Node.js 框架原理与实践相结合**
  Node.js 框架是一个强大的服务端脚本开发库，无论读者具有什么编程背景，都可以通过它来优化、改进自己的服务器网站。本书的代码段基于简单易学的原理，提供了丰富多样的操作特性，这使得本书成为了适用于各类脚本编程的必备工具。
- **涵盖内容全面**
  本书的实例从控制台应用、模块和包管理、异步 I/O 与 Async 流程控制库、Buffer、进程管理、子进程通信、OS 操作系统、文件系统、路径处理、TCP/UDP 网络编程、流（Stream）、Web 开发、常用工具及数据库交互等方面的内容出发，由简入繁、点面结合、配合原理解释，给读者呈现了一场代码盛宴。
- **自发式学习**
  在学习代码前，先让读者练习实际上最基础却最容易做错的 Node.js 框架代码和面试题，激发读者的学习斗志。

本书的设计始于功能、终于代码，是 IT 设计人员的案头必备参考书。

本书的内容安排

本书共 15 章，各章节实现了不同类别的 Node 代码段。

第 1 章　主要介绍了 Node.js 框架的控制台模块，通过该模块的方法可以向操作系统控制台实现各种格式化输入和输出等操作，也就是读者所熟知的“读取-求值-输出”循环

（Read-Eval-Print Loop，简称 REPL）交互式的编程环境。

第 2 章　主要介绍了 Node.js 框架自有的一套模块加载系统，通过该模块可以把各个功能拆分、封装到不同的模块之中，在需要的时候调用该模块。Node.js 框架使用模块和包来组织管理，从性质及加载方式上可以分为以下几类内容：核心模块、文件模块、从文件夹加载、文件夹作为模块和模块缓存。

第 3 章　主要介绍了 Node.js 异步 I/O 编程，Node.js 框架在设计之初就被考虑作为一个高效的 Web 服务器而存在，因此高效的异步机制贯穿于整个 Node.js 框架的编程模型之中。通过本章的介绍，读者可以学到异步 I/O 机制、异步 I/O 应用和 Async 流程控制库应用等 Node.js 框架异步编程的内容。

第 4 章　主要介绍了 Node.js 框架中 Buffer 的概念，它可以理解为是缓冲区或临时存储区，是暂时存放输入、输出数据的一小块内存。如果读者学过 C 语言编程，对指针数组的概念有一定的了解的话，那么学习和掌握 Node.js 框架的 Buffer 就会容易很多。

第 5 章　主要介绍了使用 Node.js 框架中功能强大的进程管理模块（Process）的方法。Process 模块是 Node.js 框架的一个全局内置对象，Node.js 代码可以在任何位置访问该对象，实际上这个对象就是 Node.js 代码宿主的操作系统进程对象。使用 Process 模块可以截获进程的异常、退出等事件，可以获取进程的环境变量、当前目录、内存占用等信息，还可以进行工作目录切换、进程退出等操作。

第 6 章　主要介绍了使用 Node.js 框架的 child_process 模块创建子进程的四个方法，分别是 spawn()、exec()、execFile()和 fork()。其中 spawn()是最原始的创建子进程的方法，其他三个都是通过对 spawn()方法不同程度的进一步封装实现的。使用 child_process 模块提供的这些方法，可以实现多进程任务、操作 shell 和进程通信等操作，实用功能是非常强大的。

第 7 章　主要介绍了 Node.js 框架中的操作系统（OS）模块，该模块提供了一系列与操作系统相关的函数方法，不过这些方法相对简单，实现的功能也十分有限。

第 8 章　主要介绍了 Node.js 框架中的文件系统（File System）模块如何来支持 I/O 操作的方法，这些操作方法是对标准 POSIX 函数的简单封装，它提供了文件的读取、写入、更名、删除、遍历目录、链接等 POSIX 文件系统操作。

第 9 章　主要介绍了 Node.js 框架中的路径处理（Path）模块、url 路径处理（url）模块以及字符串解析（Query String）模块，这些模块提供了一系列与路径解析处理相关的函数方法，这些方法对于处理常规的需求是足够的。

第 10 章　主要介绍了 Node.js 框架中对 TCP/UDP 网络编程方面的支持，Node.js 框架为设计人员提供了网络（Net）模块来支持 TCP 协议应用，数据报套接字（UDP）模块来支持 UDP 协议应用，这两个模块提供了一系列与网络应用相关的函数方法，通过这些方法可以构建基本的网络应用。

第 11 章　主要介绍了 Node.js 框架中的抽象接口流（Stream）模块，流（Stream）模块操作最主要的是.pipe()方法，它可以为开发者提供可以重复使用的统一的接口，通过抽象的流（Stream）接口来控制流（Stream）之间的读写平衡。

第 12 章　主要介绍了应用 Node.js 框架中的 HTTP 模块与 HTTPS 模块开发 Web 应用的方法，这两个模块基于 HTTP 协议与 HTTPS 协议开发，提供了一系列与 Web 应用开

发相关的函数方法，通过这些方法可以构建各种功能复杂且强大的 Web 应用。

第 13 章　主要介绍了 Node.js 框架中的常用工具（Util）模块，该模块是为了解决核心 JavaScript 的功能过于精简而设计的。该模块可以实现对一个原型对象的继承功能，实现对象格式化操作、将任意对象转换为字符串操作、调试输出功能、正则表达式验证等。

第 14 章　主要介绍了 Node.js 框架与 MySQL 数据库交互的方法，主要选用的是目前人气最高的 node-mysql 开源项目作为 Node.js 框架 MySQL 扩展库，该开源项目提供了 MySQL 数据库对 Node.js 框架的完整支持，具有一套与数据库开发相关的函数方法，通过这些方法就可以非常方便地构建 Node.js 数据库应用。

第 15 章　主要介绍了 Node.js 框架与 MongoDB 数据库交互的方法，主要选用的是同名的 mongodb 开源项目作为 Node.js 框架的 MongoDB 扩展库，该扩展库具有一套与数据库开发相关的函数方法，通过这些方法就可以非常方便地构建 Node.js 数据库应用。

本书面对的读者

- Web 服务器设计入门者
- Web 服务器开发入门者
- Node.js 框架学习爱好者
- 由 JavaScript 向 Node.js 框架转型的开发人员
- 中小型企业网站开发者
- 大中专院校的学生
- 各种 IT 培训学校的学生
- 网站后台开发人员
- 网站建设与网页设计的相关威客兼职人员

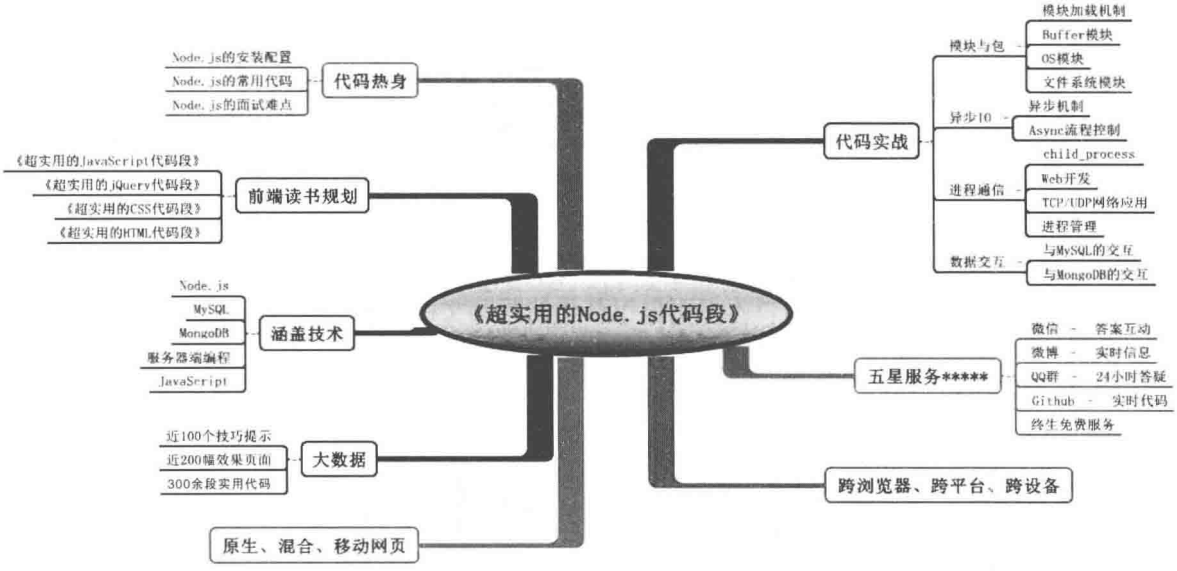
本书的服务

笔者能力有限，如果读者发现我们在写作过程中有什么疏漏，或者您对本书有什么疑问，可通过以下方式与我们沟通。

- QQ 群：296811675，作者在线答疑。
- 扫描封底的微信二维码，时刻参与我们的图书互动。
- 通过新浪微博@博文视点 Broadview，了解我们发布的信息和各种前端流行技术。
- 博文视点官方网站 http://www.broadview.com.cn/，下载本书所有实例源代码。
- Github，https://github.com/kingwjz/Node.js-code，了解代码的实时更新和迭代过程，可以在每章代码下参与讨论，也可以观看其他读者提出的问题，还可以随时随地下载代码。

很多读者在学习过程中苦于无法交流，小故障无法及时解决，加入我们的服务方阵，我们将为您提供终生免费的服务。一本书、一段情、一辈子。

本书的思维导图



编者推荐

本书摒弃传统的说教模式，每段代码都是单独的功能型页面，读者可以从全书的任意一点开始线性阅读。本书的目的就是将最有用的代码与读者分享，包含了服务器设计人员在实战中必须具备的所有技巧和方法，读者可以拿来就用。本书 300 余段代码也许并不是最优的代码，但笔者提供了 Github 地址，与全世界 IT 工程师一起优化了这些代码，并实现了更新迭代，以保证读者始终能看到最好的、最高效的、最实用的 Node 代码段。这是一本市场上绝无仅有的 Node 实战书，是一本值得拥有的 Node 设计书。

序 1　Node.js 的前世今生

这是一本是有显著特点的 Node.js 书，它涉及了脚本编程的技巧、脚本运行性能的优化、快速开发服务器的应用。所以本书不仅仅是你的帮助文档，更是你的良师益友，拥有这些代码，你将会在服务器脚本工作中战无不胜！

Node.js 框架概述

Node.js 是一个基于 Google 公司 Chrome 浏览器的 JavaScript 运行时建立的平台，主要用于搭建响应速度快、方便扩展的服务器 Web 应用。其实，Node 本身就是一个 JavaScript 运行环境，是对 Google V8 引擎的完美封装。Google V8 引擎执行 JavaScript 脚本的速度非常快、性能非常好，Node.js 框架对它进行了优化，提供了相应替代的 API，保证 V8 引擎在非浏览器环境下运行得更好。

Node.js 框架基于事件驱动、非阻塞 I/O 模型，因而得以轻量和高效，非常适合在分布式设备终端上运行数据密集型的实时应用。Node.js 框架采用一套“非阻塞”I/O 库来支持事件循环的方式，更好地为文件系统、数据库之类的资源提供了接口。目前，由于 Web 服务器与富客户端浏览器应用程序之间共享代码的方法只能通过 JavaScript 脚本语言来实现，因此对服务器脚本语言支持最完美的 Node.js 框架发展十分迅猛，实际上已经成为了服务器端的 JavaScript 脚本语言开发平台。

更为重要的是，在 Node.js 框架发布以来的很短时间内，各大开源社区就已经贡献了大量的扩展库（模块），提供了对各种开源软件或数据库驱动的支持。因此，虽然 Node 项目还非常年轻，但众多的软件开发人员都对 Node 项目展现出了极大的热情，并贡献出了自己的一份力量，改进并完善了 Node.js 框架。

Node.js 框架环境搭建

学习 Node.js 框架的第一步，需要先搭建好其工作环境，这样才能运行并测试 Node 脚本文件。下面具体介绍一下。

（1）获取 Node 安装包，可以访问以下 Node.js 框架官方网址：

```
https://nodejs.org/download/
```

在该页面中，可以看到适用于不同操作系统的、各个版本的 Node 安装包源文件，选中我们需要的版本类型直接下载就可以。

在 Windows 操作系统环境下安装 Node.js 框架，直接运行下载好的 Node 安装包源文件即可，目的路径一般选择如下：

```
D:/nodejs/
```

安装完毕后，读者可以到该目录下浏览一下具体内容，这样会对 Node 环境有一个大致的了解。

（2）在 Ubuntu（Linux）环境下需要运行 apt 命令进行安装，具体方法如下：

```
sudo apt-get install nodejs
```

如果想获取最新的 Node 安装源，具体方法如下：

```
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install nodejs
```

一般情况下，node 命令会被自动安装在以下路径之中（视不同的 Ubuntu 系统版本而定）：

```
/usr/local/bin/node
```

大致经过以上这几步的操作，Node.js 框架就安装完毕了，为了检验 Node 环境是否正确，可以使用以下命令来检测 Node 的版本：

```
node –v
```

如果显示出正确的版本号，则说明 Node 环境搭建成功了。

Node.js 框架推荐工具

1．开发工具，平台类

（1）jetBrains 公司的 WebStorm 开发工具

WebStorm 是 jetBrains 公司开发的一款 JavaScript 开发工具，被众多 JavaScript 开发者作为首选的开发平台，完全支持 Windows、Linux、Mac OS X 等操作系统，有“Web 前端开发神器”的美誉。读者可以从下面的网址获取该工具：

```
http://www.jetbrains.com/webstorm/
```

（2）Sublime Text 代码编辑器

Sublime Text 是一款代码编辑器（支持 C、Java、JavaScript、HTML、CSS 等），同时也可以作为文本编辑器使用。Sublime Text 具有漂亮的用户界面和强大的扩展功能，比如支持代码缩略图、各种工具插件、多窗口任务等功能。同时，Sublime Text 也是一款跨平

台的编辑器，完全支持 Windows、Linux、Mac OS X 等操作系统。读者可以从下面的网址获取该工具：

http://www.sublimetext.com/

（3）Eclipse Node.js IDE 插件

对于已经习惯了使用 Eclipse 开发工具的读者来讲，可以使用 Eclipse Node.js IDE 这款插件来开发 Node 程序。读者可以从下面的网址获取该插件工具：

http://www.nodeclipse.org/

（4）Node.js Tools for Visual Studio 插件

如果读者习惯使用微软公司的 Visual Studio 系列开发工具，同样有一款 Node.js Tools for Visual Studio 插件可以提供对开发 Node 程序的支持。读者可以从下面的网址获取该插件工具：

https://nodejstools.codeplex.com

2．测试调试工具类

（1）NodeUnit

NodeUnit 是一款应用于 Node.js 的单元测试框架，基于 assert 模块开发，读者可以从下面的网址获取该测试工具：

https://github.com/caolan/nodeunit

（2）Supervisor

如果希望在 Node 代码修改后立即看到效果，而不是每次都要重启 Node 服务才能看到效果，Supervisor 调试工具可以帮助实现这个功能，它会监视代码的改动并自动重启 Node 服务。读者可以从下面的网址获取该测试工具：

http://supervisord.org

（3）Mocha

Mocha 是一款完整的测试工具，可安装在 NPM 和 CI 服务器之上。读者可以从下面的网址获取该测试工具：

https://github.com/mochajs/mocha

（4）should.js

should.js 模块是 assert 模块的扩展，其语法与日常用的语法几乎一模一样，非常易于使用。读者可以从下面的网址获取该测试工具：

https://github.com/visionmedia/should.js

3．第三方开发包类

（1）Async 异步处理

Async 是一个流程控制工具包，提供了直接而强大的异步功能，该插件基于 JavaScript

语言专门为 Node.js 设计，同时也可以直接在浏览器中使用。读者可以从下面的网址获取该开发工具：

https://github.com/caolan/async

（2）Express 开发包

Express 是一个简洁而灵活的 Node.js Web 应用框架，提供一系列强大特性，帮助我们创建各种 Web 应用。Express 不对 Node.js 已有的特性进行二次抽象，只是在其之上扩展了 Web 应用所需的功能。Express 开发包所具有的丰富的 HTTP 工具以及来自 Connect 框架的中间件可以为审计人员随取随用，并创建出友好的、快速且简单的 API。读者可以从下面的网址获取该开发工具：

https://www.npmjs.com/package/express

（3）node-mysql 开发包

node-mysq 开发包是一个纯 Node.js 的用 JavaScript 实现的 MySQL 客户端程序。node-mysql 封装了 Node.js 框架对 MySQL 的基本操作，拥有 100%的 MIT 公共许可证。读者可以从下面的网址获取该开发工具：

https://github.com/felixge/node-mysql

本书浏览器约定

本书涉及的浏览器测试基准如图 1 所示，可以看到内核和外壳的占有情况，内环为内核，外环为外壳，这也是接下来要提到的浏览器同类项的数据基础。

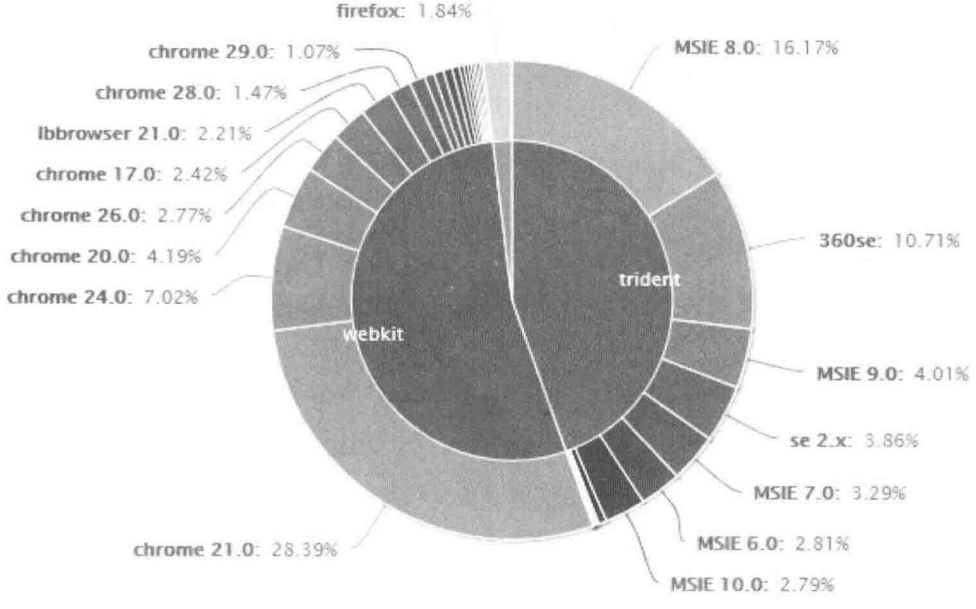


图 1 浏览器基准

不同的浏览器可能会采用相同的内核（渲染引擎）。一方面，外壳实现的差异性会影响整个网站功能，这也是为什么内核无法完全代表所有浏览器的原因。另一方面，某些浏览器具有相同的血统，是可以归为同一类的，同一类中，如果测试通过其中一个，那么其他与之同类的浏览器也基本上可以通过测试。图 1 是根据相同的内核或对 W3C 标准有类似的支持程度对浏览器进行归类的。之所以这样做，是因为测试基准除了要基本反映浏览器的市场占有率，还要考虑到开发测试的成本。因此，测试基准中的浏览器应当具有典型代表性。

浏览器的不同版本的内核也不一样，因此通常要针对不同版本的浏览器做测试，开发者要了解内核对标准的支持，比如 IE 的渲染引擎 Trident 的不同版本差别较大，因此 IE 需要测试 6~10 的所有版本，然而由于 Firefox 和 Chrome 升级覆盖面广，因此基准中只保留其最新版本。

本书涉及的代码会根据需要，对不同浏览器进行有针对性的测试。

序 2　最简单却必须看的 Node 代码

Node 代码基于 JavaScript 代码，却也有所不同，相信通过对本书的学习，读者会对 Node 技术爱不释手！下面介绍一些很简单、却又很有特点的 Node 代码。

1. 控制台运行 Node 代码的方法：

```
01  $ node
02  > console.log('Hello Node!');
03  Hello Node!
```

第 01 行代码运行 Node 环境；第 02 行代码调用 Node.js 框架的打印输出命令；第 03 行代码为控制台打印输出的结果。

2. 控制台运行 Node 脚本的方法：

```
01  $ node hello.js
02  Hello Node!
```

第 01 行代码使用 Node 命令运行 hello.js 脚本文件（脚本文件中代码同第 1 个例子中的第 02 行代码）；第 02 行代码为控制台打印输出的结果。

3. 使用 require 命令加载模块的方法：

```
01  var http = require('http');    // TODO: 加载核心模块‘http’
02  var circle = require('./ch02.module-file-circle.js');
```

第 01 行代码使用 require 命令加载核心模块‘http’；第 02 行代码使用 require 命令加载自定义模块'./ch02.module-file-circle.js'。

4. 使用 exports 命令输出对象的方法：

```
01  exports.area = function(r) {
02      return PI * r * r;
03  };
```

这段代码使用 exports 命令输出一个 area 方法，用于计算圆的面积。

5. 使用 async 流程库的方法：

```
01  async.series([
02      function(callback) {
03          callback(null, 'hello');
```

```
04        },
05        function(callback) {
06            callback(null, 'async');
07        },
08        function(callback) {
09            callback(null, 'series');
10        }
11    ],function(err, results) {
12        console.log(results);
13    });
```

这段代码使用了 async 流程库的 series 方法来控制流程，async 流程库还有其他几种控制方法，读者可以阅读本书的详细内容。

6. I/O 异常捕获的方法：

```
01  try{
02      setTimeout(function(){
03          var data = a/0;      // TODO: 错误的计算
04      },1000);
05  }catch (e){
06      console.log(e);
07  }
```

这段代码演示了使用 try/catch 语法进行 I/O 异常捕获的方法，第 03 行代码输出了一个错误的计算，第 06 行代码输出捕获的错误异常。

7. 初始化 Buffer 对象的方法：

```
01  var buffer = new Buffer("This is Buffer", "utf8");  // TODO: 初始化 Buffer
```

这段代码演示了初始化 Buffer 对象的方法，并定义了“utf8”编码格式。

8. 获取进程信息的方法：

```
01  console.info('当前进程 id:');
02  console.info(process.pid);
03  console.info('当前进程名称:');
04  console.info(process.title);
```

这段代码演示了通过 process 模块获取进程 id 和进程名称的方法。

9. 创建子进程的方法：

```
01  var spawn = require('child_process').spawn;     // TODO: 引入 child_process 模块
02  var ls_var = spawn('ls', ['-lh', '/var']);          // TODO: 定义命令行 'ls -lh /var'
```

这段代码演示了通过 child_process 模块创建子进程的方法，并实现了查看/var 目录的命令。

10. 获取操作系统信息的方法：

```
var type = os.type();
console.info('当前操作系统类型为: ' + type);
var platform = os.platform();
console.info('当前操作系统平台为: ' + platform);
```

这段代码演示了通过操作系统 OS 模块获取操作系统类型和平台信息的方法。

11. 读文件的方法：

```
if(fs.existsSync(file_path)) {
    var file_contents = fs.readFileSync(file_path, 'utf-8');   // TODO: 读文件（同步方式）
    console.info('read txt/readFileSync.txt contents: ');
    console.info(file_contents);                               // TODO: 打印输出文件内容
    console.info();
}
```

这段代码演示了通过文件系统 fs 模块读文件的方法。

12. 规范化字符串路径的方法：

```
var path = require('path');                                    // TODO: 引入路径处理模块
var path_a = "/home//king";
console.info('path.normalize("/home//king") is : ' + path.normalize(path_a));
```

这段代码演示了通过路径 path 模块规范化字符串路径的方法。

13. 创建 TCP 服务器的方法：

```
var net = require('net');                  // TODO: 引入网络(Net)模块
var HOST = '127.0.0.1';                    // TODO: 定义服务器地址
var PORT = 9696;                       // TODO: 定义端口号
net.createServer(function(sock) {
    /**
     * 添加事件处理方法
     */
}).listen(PORT, HOST);
```

这段代码演示了通过网络 net 模块创建 TCP 服务器的方法。

14. 创建 HTTP 服务器的方法：

```
var http = require('http');                // TODO: 引入 http 模块
http.createServer(function(req, res) {
    /**
     * 通过 res.writeHeader()函数方法写 HTTP 文件头
     */
    res.writeHead(200, {'Content-type' : 'text/html'});
    /**
     * 通过 res.write()函数方法写页面内容
```

```
09          */
10          res.write('<h3>Node.js --- HTTP</h3>');
11          /**
12           * 通过 res.end()函数方法发送响应状态码,并通知服务器消息完成
13           */
14          res.end('<p>Create Basic HTTP Server!</p>');
15      }).listen(6868);                        // TODO: 监听 6868 端口号
```

这段代码演示了通过 HTTP 模块创建 Web 服务器的方法。

15. Node.js 框架连接 MySQL 数据库的方法：

```
01  var http = require("http"); // TODO: 引入 HTTP 模块
02  var mysql = require('/usr/local/lib/node_modules/mysql');    // TODO: 引入 mysql 模块
03  var connection = mysql.createConnection({
04      host: "localhost",                  // TODO: 主机地址
05      user: "root",                       // TODO: 数据库用户名
06      password: "root",                   // TODO: 数据库密码
07      database: "nodejs",                 // TODO: 数据库名称
08      port: 3306                          // TODO: 端口号
09  });
10  http.createServer(function (req, res) {
11      res.writeHead(200, { "Content-Type" : "text/html;charset=utf8" });
12      res.write("<h3>测试 Node.js - MySQL 数据库连接!</h3><br/>");
13      connection.connect(function(err) {
14          if(err) {
15              res.end('<p>Error Connected to MySQL!</p>');
16              return;
17          } else {
18              res.end('<p>Connected to MySQL!</p>');
19          }
20      });
21  }).listen(6868);      // TODO: 监听 6868 端口号
```

这段代码演示了 Node.js 框架连接 MySQL 数据库的方法。