

Node.js权威指南

作者：陆凌牛 著

实战

Node.js权威指南

陆凌牛 著

ISBN: 978-7-111-46078-7

本书纸版由机械工业出版社于2014年出版，电子版由华章分社（北京华章图文信息有限公司）全球范围内制作与发行。

版权所有，侵权必究

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @研发书局

腾讯微博 @yanfabook

目录	
前言	
第1章 Node.js介绍	
1.1 Node.js概述	
1.1.1 使用Node.js能够解决什么问题	
1.1.2 实现高性能服务器	
1.1.3 非阻塞型I/O及事件环机制	
1.1.4 Node.js适合开发的应用程序	
1.2 安装Node.js	
1.3 Node.js中的模块	
1.4 一个简单的示例应用程序	
1.5 小结	
第2章 Node.js中的交互式运行环境——REPL	
2.1 REPL运行环境概述	
2.2 在REPL运行环境中操作变量	
2.3 在REPL运行环境中使用下划线字符	
2.4 在REPL运行环境中直接运行函数	
2.5 在REPL运行环境中定义并启动服务器	
2.6 REPL运行环境中的上下文对象	
2.7 REPL运行环境中的基础命令	
2.8 小结	
第3章 Node.js基础知识	
3.1 Node.js中的控制台	
3.1.1 console.log方法	
3.1.2 console.error方法	
3.1.3 console.dir方法	
3.1.4 console.time方法与console.timeEnd方法	
3.1.5 console.trace方法	
3.1.6 console.assert方法	
3.2 Node.js中的全局作用域及全局函数	
3.2.1 console.log方法	
3.2.2 setTimeout函数与clearTimeout函数	
3.2.3 setInterval函数与clearInterval函数	
3.2.4 定时器对象的unref方法与ref方法	
3.2.5 与模块相关的全局函数及对象	
3.3 __filename变量与__dirname变量	
3.3.1 __filename变量	
3.3.2 __dirname变量	
3.4 事件处理机制及事件环机制	
3.4.1 EventEmitter类	
3.4.2 EventEmitter类的各个方法	
3.4.3 获取指定事件的事件处理函数的数量	
3.4.4 EventEmitter类自身所拥有的事件	
3.4.5 事件环机制	
3.5 在Node.js中使用调试器	
3.5.1 在命令行窗口中使用调试器	
3.5.2 观察变量值或表达式的执行结果	
3.5.3 设置与取消断点	
3.5.4 调试器中可以使用的其他实用命令	
3.5.5 使用node-inspector调试工具	
3.6 小结	
第4章 模块与npm包管理工具	
4.1 核心模块与文件模块	
4.2 从模块外部访问模块内的成员	
4.2.1 使用exports对象	
4.2.2 将模块定义为类	
4.2.3 为模块类定义类变量或类函数	
4.3 组织与管理模块	
4.3.1 从node_modules目录中加载模块	
4.3.2 使用目录来管理模块	

- [4.3.3 从全局目录中加载模块](#)
- [4.4 模块对象的属性](#)
- [4.5 包与npm包管理工具](#)
 - [4.5.1 Node.js中的包](#)
 - [4.5.2 npm包管理工具](#)
- [4.6 小结](#)
- [第5章 使用Buffer类处理二进制数据](#)
 - [5.1 创建Buffer对象](#)
 - [5.2 字符串的长度与缓存区的长度](#)
 - [5.3 Buffer对象与字符串对象之间的相互转换](#)
 - [5.3.1 Buffer对象的toString方法](#)
 - [5.3.2 Buffer对象的write方法](#)
 - [5.3.3 StringDecoder对象](#)
 - [5.4 Buffer对象与数值对象之间的相互转换](#)
 - [5.5 Buffer对象与JSON对象之间的相互转换](#)
 - [5.6 复制缓存数据](#)
 - [5.7 Buffer类的类方法](#)
 - [5.7.1 isBuffer方法](#)
 - [5.7.2 byteLength方法](#)
 - [5.7.3 concat方法](#)
 - [5.7.4 isEncoding方法](#)
 - [5.8 小结](#)
- [第6章 在Node.js中操作文件系统](#)
 - [6.1 同步方法与异步方法](#)
 - [6.2 对文件执行读写操作](#)
 - [6.2.1 文件的完整读写](#)
 - [6.2.2 从指定位置处开始读写文件](#)
 - [6.3 创建与读取目录](#)
 - [6.3.1 创建目录](#)
 - [6.3.2 读取目录](#)
 - [6.4 查看与修改文件或目录的信息](#)
 - [6.4.1 查看文件或目录的信息](#)
 - [6.4.2 检查文件或目录是否存在](#)
 - [6.4.3 获取文件或目录的绝对路径](#)
 - [6.4.4 修改文件访问时间及修改时间](#)
 - [6.4.5 修改文件或目录的读写权限](#)
 - [6.5 可以对文件或目录执行的其他操作](#)
 - [6.5.1 移动文件或目录](#)
 - [6.5.2 创建与删除文件的硬链接](#)
 - [6.5.3 创建与查看符号链接](#)
 - [6.5.4 截断文件](#)
 - [6.5.5 删除空目录](#)
 - [6.5.6 监视文件或目录](#)
 - [6.6 使用文件流](#)
 - [6.6.1 流的基本概念](#)
 - [6.6.2 使用ReadStream对象读取文件](#)
 - [6.6.3 使用WriteStream对象写入文件](#)
 - [6.7 对路径进行操作](#)
 - [6.8 小结](#)
- [第7章 实现基于TCP与UDP的数据通信](#)
 - [7.1 使用net模块实现基于TCP的数据通信](#)
 - [7.1.1 创建TCP服务器](#)
 - [7.1.2 socket端口对象](#)
 - [7.1.3 创建TCP客户端](#)
 - [7.1.4 net模块中的类方法](#)
 - [7.2 使用dgram模块实现基于UDP的数据通信](#)
 - [7.2.1 创建UDP服务器与客户端](#)
 - [7.2.2 实现广播与组播](#)
 - [7.3 小结](#)
- [第8章 创建HTTP与HTTPS服务器及客户端](#)

- 8.1 HTTP服务器
 - 8.1.1 创建HTTP服务器
 - 8.1.2 获取客户端请求信息
 - 8.1.3 转换URL字符串与查询字符串
 - 8.1.4 发送服务器端响应流
- 8.2 HTTP客户端
 - 8.2.1 向其他网站请求数据
 - 8.2.2 向本地服务器请求数据
 - 8.2.3 制作代理服务器
- 8.3 创建HTTPS服务器与客户端
 - 8.3.1 创建HTTPS服务器
 - 8.3.2 创建HTTPS客户端
- 8.4 小结
- 第9章 进程与子进程
 - 9.1 Node.js中的进程
 - 9.1.1 进程对象的属性
 - 9.1.2 进程对象的方法与事件
 - 9.2 创建多进程应用程序
 - 9.2.1 使用spawn方法开启子进程
 - 9.2.2 使用fork方法开启子进程
 - 9.2.3 使用exec方法开启子进程
 - 9.2.4 使用execFile方法开启子进程
 - 9.3 在多个子进程中运行Node.js应用程序
 - 9.3.1 使用fork方法创建worker对象
 - 9.3.2 worker对象的方法与事件
 - 9.4 小结
- 第10章 Node.js中的错误处理与断言处理
 - 10.1 使用domain模块处理错误
 - 10.1.1 domain模块概述
 - 10.1.2 创建并使用Domain对象
 - 10.1.3 隐式绑定与显式绑定
 - 10.1.4 绑定回调函数与拦截回调函数
 - 10.1.5 domain堆栈的弹出与推入
 - 10.1.6 Domain对象的销毁
 - 10.2 Node.js中的断言处理
 - 10.2.1 equal方法与notEqual方法
 - 10.2.2 strictEqual方法与notStrictEqual方法
 - 10.2.3 assert方法与ok方法
 - 10.2.4 deepEqual方法与notDeepEqual方法
 - 10.2.5 throws方法与doesNotThrow方法
 - 10.3 小结
- 第11章 加密与压缩
 - 11.1 加密与解密处理
 - 11.1.1 crypto模块概述
 - 11.1.2 散列算法
 - 11.1.3 HMAC算法
 - 11.1.4 公钥加密
 - 11.2 压缩与解压缩处理
 - 11.2.1 创建各种用于压缩及解压缩的对象
 - 11.2.2 zlib模块中的各种方法
 - 11.3 小结
- 第12章 Node.js中的其他模块
 - 12.1 使用dns模块解析域名
 - 12.1.1 使用resolve方法将域名解析为DNS记录
 - 12.1.2 使用lookup方法查询IP地址
 - 12.1.3 使用reverse方法反向解析IP地址
 - 12.1.4 dns模块中的各种错误代码
 - 12.2 使用punycode模块转换punycode编码
 - 12.3 使用os模块获取操作系统信息
 - 12.4 使用readline模块逐行读取流数据

- [12.4.1 创建Interface对象](#)
- [12.4.2 Interface对象所拥有的各种方法与事件](#)
- [12.5 使用util模块中提供的一些实用方法](#)
- [12.6 使用vm模块改变脚本运行环境](#)
 - [12.6.1 在独立环境中运行JavaScript代码](#)
 - [12.6.2 创建并使用Script对象](#)
- [12.7 自定义REPL运行环境](#)
- [12.8 小结](#)
- [第13章 数据库访问](#)
 - [13.1 在MongoDB数据库中存取数据](#)
 - [13.1.1 MongoDB概述](#)
 - [13.1.2 安装MongoDB数据库](#)
 - [13.1.3 安装MongoDB包](#)
 - [13.1.4 连接MongoDB数据库](#)
 - [13.1.5 在MongoDB数据库中插入数据](#)
 - [13.1.6 在MongoDB数据库中查询数据](#)
 - [13.1.7 在MongoDB数据库中更新与删除数据](#)
 - [13.1.8 使用Mongoose类库](#)
 - [13.2 在MySQL数据库中存取数据](#)
 - [13.2.1 建立连接与关闭连接](#)
 - [13.2.2 执行数据的基本处理](#)
 - [13.2.3 执行存储过程](#)
 - [13.2.4 执行多表结合查询](#)
 - [13.2.5 以数据流的方式处理查询数据](#)
 - [13.2.6 创建连接池](#)
 - [13.3 小结](#)
- [第14章 使用Express构建Web应用程序](#)
 - [14.1 Express概述](#)
 - [14.1.1 安装Express](#)
 - [14.1.2 使用Express开发一个简单的示例应用程序](#)
 - [14.2 设置路由](#)
 - [14.3 使用各种提交数据或请求数据的方法](#)
 - [14.3.1 使用post方法接收客户端提交的POST请求](#)
 - [14.3.2 使用put方法接收客户端提交的PUT请求](#)
 - [14.3.3 使用delete方法接收客户端提交的DELETE请求](#)
 - [14.3.4 使用all方法接收客户端提交的各种请求](#)
 - [14.4 中间件](#)
 - [14.4.1 中间件概述](#)
 - [14.4.2 Express框架中内置的中间件](#)
 - [14.4.3 basicAuth中间件](#)
 - [14.4.4 bodyParser中间件](#)
 - [14.4.5 cookieParser中间件](#)
 - [14.4.6 logger中间件](#)
 - [14.4.7 methodOverride中间件](#)
 - [14.4.8 responseTime中间件](#)
 - [14.4.9 router中间件](#)
 - [14.4.10 session中间件](#)
 - [14.4.11 static中间件](#)
 - [14.4.12 directory中间件](#)
 - [14.4.13 Express 3中的异常处理机制](#)
 - [14.4.14 limit中间件函数](#)
 - [14.4.15 配置应用程序](#)
 - [14.5 模板引擎](#)
 - [14.5.1 模板引擎概述](#)
 - [14.5.2 Jade模板引擎的使用方法](#)
 - [14.5.3 EJS模板引擎的使用方法](#)
 - [14.6 小结](#)
- [第15章 使用Socket.IO类库实现WebSocket通信](#)
 - [15.1 Socket.IO概述](#)
 - [15.2 Socket.IO的使用方法](#)

[15.3 在Express框架中使用Socket.IO](#)

[15.4 在服务器端保存用户数据](#)

[15.5 广播消息](#)

[15.6 使用命名空间](#)

[15.7 小结](#)

[第16章 综合案例介绍](#)

[16.1 创建简单聊天室应用程序](#)

[16.1.1 案例概述](#)

[16.1.2 页面显示效果](#)

[16.1.3 HTML页面代码及CSS样式代码](#)

[16.1.4 JavaScript脚本代码部分](#)

[16.1.5 服务器端代码](#)

[16.2 创建Web应用程序](#)

[16.2.1 案例概述](#)

[16.2.2 页面展示效果](#)

[16.2.3 订单检索页面](#)

[16.2.4 订单编辑页面](#)

[16.3 小结](#)

前言

为何写作本书

近年来，随着智能手机与HTML 5的不断普及，JavaScript脚本语言的重要性也随之不断提升，IT业界涌现了大量学习与善用JavaScript脚本语言的工程师，其中许多工程师对任何服务器端开发语言均一无所知。很多工程师提出，如果能够让服务器端与客户端均使用一种脚本语言，则无疑可以减少服务器端的开发难度，提高服务器端的开发效率。另一方面，由于近几年许多JavaScript引擎中均内置了JIT（Just In Time）编译器，使JavaScript引擎的处理速度得到了大幅度提高，JavaScript脚本语言的运行速度不会逊色于任何服务器端开发语言。

据此现状，2009年8月，IT业界制定了CommonJS标准，用于标准化服务器端JavaScript脚本语言，制定服务器端JavaScript脚本语言中所需要实现的处理。

同年，美国人Ryan Dahl推出了第一个遵循CommonJS标准的服务器端JavaScript脚本语言开发框架——Node.js。在Node.js内部，运行的是Google开发的高性能V8JavaScript脚本语言，该语言可以运行在服务器端。Node.js的一个最重要的特性是通过单线程实现异步处理环境。通常，提及异步处理，开发者们首先会联想到的是服务器端多线程环境，在Node.js中，通过事件环与非阻塞型I/O机制实现服务器端的异步处理。

为了帮助国内的Web开发者更好地学习Node.js开发框架，笔者特此推出本书，希望国内的Web开发者们能够通过学习本书尽早地运用Node.js框架开发出高效的Web服务器以及运行于该Web服务器中的Web应用程序。

读者对象

根据不同使用需要，本书适用于如下读者：

- 对Web网站或Web应用程序的开发技术感兴趣或者打算从事Web网站或Web应用程序开发的技术人员。
- Web网站或Web应用程序的开发者（包括Web前端开发工程师及后端开发工程师）。
- 有关Web网站或Web应用程序开发项目的项目管理人员。
- 开设相关课程的大专院校及培训机构。

如何阅读本书

本书内容分三大部分展开。

第一部分：第1章详细阐述什么是Node.js框架，为什么要选择Node.js框架进行服务器端的开发，使用Node.js框架能够解决什么问题，Node.js框架适合用于开发哪些应用程序，如何下载及使用Node.js框架，Node.js框架的主要特性，使用Node.js框架时必须了解的基础知识。第2章～第12章针对Node.js v0.10版中的各模块进行展开阐述，详细阐述这些模块的作用，如何使用这些模块，这些模块中所提供的各对象、属性、方法及事件。

第二部分：第13章～第15章分别阐述在使用Node.js框架进行Web服务器端的开发时极有可能利用到的第三方开发包，包括如何在Node.js应用程序中使用关系型数据库及NoSQL型数据库，如何使用Express框架开发Web应用程序，如何使用Socket.io类库实现WebSocket通信。

第三部分：第16章介绍两个综合案例，在第一个案例中，我们讲述如何结合使用Node.js与Socket.io类库制作一个聊天室应用程序的服务器端及客户端，在第二个综合案例中，我们讲述如何结合使用Node.js与

Express框架制作一个Web应用程序的服务器端及客户端。

在本书的每一章中，每一个正在阐述的理论点均使用代码实例进行具体形象地说明，每个实例中所涉及的理论知识都以通俗易懂的语言进行阐述，大部分实例均使用图片来形象说明该实例的运行效果。本书所有实例代码都经笔者亲自测试运行成功，提供给读者学习使用。每个实例的详细代码及其使用到的脚本文件、各种资源文件都可在华章公司网站（www.hzbook.com）的本书页面上下载。读者可以对这些代码进行修改，以便观察各种不同效果，加深对实例代码的理解。

勘误和支持

由于时间及水平方面的原因，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。笔者QQ号码为240824399，联系邮箱为240824399@qq.com。欢迎读者通过QQ或E-mail与笔者联系，真诚期待能够听到读者的反馈意见。

致谢

感谢机械工业出版社华章公司的编辑杨福川和姜影，感谢杨福川老师的魄力和远见，感谢姜影老师的细心编辑与校对，感谢二人在这半年多的时间中始终支持我的写作，是他们的鼓励和帮助引导我顺利完成全部书稿。

谨以此书献给众多热爱Node.js开发框架的朋友们，以及中国IT界从事Web网站及Web应用程序开发的全体同行。

第1章 Node.js介绍

最近几年，Web领域出现了一个全新的JavaScript开发框架——Node.js。该框架一经问世，便以其独特的优势得到了广大开发人员的关注。本章先来对Node.js框架作一下概要介绍。

本章内容包括：

- 什么是Node.js框架，为什么要用Node.js框架，使用Node.js框架能够解决什么问题，在哪些场合下应该考虑使用Node.js框架。
- 如何下载Node.js框架。
- 什么是Node.js框架中的模块，Node.js v0.10版中内置了哪些模块以及这些模块的作用。
- 如何开发一个最简单的Node.js示例应用程序，以及如何运行这个示例应用程序。

1.1 Node.js概述

1.1.1 使用Node.js能够解决什么问题

Node.js的首要目标是提供一种简单的、用于创建高性能服务器及可在该服务器中运行的各种应用程序的开发工具。首先让我们来看一下现在的服务器端语言中存在着什么问题。在Java、PHP或ASP.NET等服务器端语言中，为每一个客户端连接创建一个新的线程，而每个线程需要耗费大约2MB的内存，也就是说，理论上，具有8GB内存的服务器可以同时连接的最大用户数为4000个左右。要让Web应用程序支持更多的用户，就需要增加服务器的数量，而Web应用程序的硬件成本也就随之增加了。不仅如此，在技术层面也会因此产生一些潜在的问题。例如，由于同一个用户的不同客户端请求可能会被不同的服务器处理，因此必须在所有的服务器之间共享所有的资源。由此可见，在一个Web应用程序中，一个主要的瓶颈是服务器所支持的最大同时连接用户量。

Node.js修改了客户端到服务器端的连接方法，解决了这个问题。因为它并不为每个客户端连接创建一个新的线程，而是为每个客户端连接触发一个在Node.js内部进行处理的事件。因此，如果使用Node.js，可以同时处理多达几万个用户的客户端连接。因此，当需要使Web应用程序能够支持大量用户的并发连接的时候，我们应该考虑使用Node.js。

欢迎访问：电子书学习和下载网站 (<https://www.shgis.cn>)

文档名称：《Node.js权威指南》陆凌牛 著.epub

请登录 <https://shgis.cn/post/284.html> 下载完整文档。

手机端请扫码查看：

