Web
前端技术
丛书

基于v7.6.0，详细介绍Node.js各种新特性，引导读者紧跟技术潮流

- 从零开始学习开发完整可用的项目，掌握语言细节并获得实际经验
- 以实际应用为背景，阐述Node.js在实际项目中的定位和潜在的陷阱
- 适合没有Node.js经验的读者，旨在快速上手到独立开发小型应用

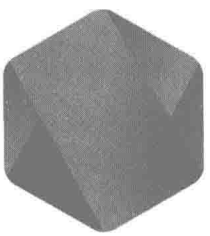
基于 Chrome V8 引擎的 JavaScript 运行环境

新时期的 Node.js入门

李锴 著

清華大學出版社

Web前端技术丛书



新时期的 Node.js入门

李锴 著

清華大學出版社
北京

内 容 简 介

Node.js 是一门开源的、为 Web 而生的语言，具有高并发、异步等特点，并且拥有一个十分活跃的开发者社区。与 Ruby、Python 等语言相比，Node.js 更年轻、更易于没有经验的人上手使用，因此很快在世界各地的开发者中收获了一大批拥趸。在国内，Node.js 在许多企业中获得了广泛应用，并取得了一系列的应用成果。然而，随着技术的不断更新、ECMAScript2015 等新标准纷纷确定，现有的中文书籍就变得有些过时。本书立足于新的技术潮流，介绍了一系列全新的语言特性和标准，以便让读者在学习基础知识的同时紧跟新技术的发展。

本书分为 8 章 6 个附录，讲解了 Node.js 的各种基础特性，使读者快速入门，同时结合语言最新的发展趋势，让读者能够紧跟技术潮流。本书围绕 Node.js 在 Web 站点开发和爬虫系统中的应用展开，对 Node.js 在大型项目中的定位与应用做了详细的说明。

本书可用于 Node.js 入门，适合未接触过 Node 的读者以及在校的学生阅读，也适合作为高等院校和培训学校相关专业的师生教学参考。

图书在版编目（CIP）数据

新时期的 Node.js 入门 / 李锴著. —北京：清华大学出版社，2018

（Web 前端技术丛书）

ISBN 978-7-302-48780-7

I. ①新… II. ①李… III. ①JAVA 语言－程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字（2017）第 272827 号

责任编辑： 夏毓彦
封面设计： 王　翔
责任校对： 闫秀华
责任印制： 刘祎淼

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者： 三河市铭诚印务有限公司
经　　销： 全国新华书店
开　　本： 190mm×260mm　　**印　张：** 18.5　　**字　数：** 474 千字
版　　次： 2018 年 1 月第 1 版　　**印　次：** 2018 年 1 月第 1 次印刷
印　　数： 1~3000
定　　价： 49.00 元

产品编号：075003-01

前 言

国内 Node 的开发者很多都读过朴灵写的《深入浅出 Node.js》（以下简称《深入浅出》）一书，笔者也不例外，笔者在 2014 年初第一次接触 Node，最初读的几本书就包含了这本《深入浅出》，该书出版于 2013 年 12 月，距今已经差不多有 4 年的时间了。

对于一门高速发展的语言来说，4 年算得上很长的时间了。4 年前 Node 的版本号还在 0.10.x，而时至今日，已经迎来 v8.0.0 的最新版本了。

Node 项目始于 2009 年，2013 年恰好处在当今（2017 年）和 2009 年的中间节点，一门语言在诞生之初的发展总是最快的，到了现在，Node 逐渐地变得稳定下来。

但即使这样，这 4 年中也发生了不少大事件：Node 从分裂又走向了统一，ES2015 标准的推出等。

那么 4 年后的今天，Node 有了哪些改变呢？

一方面，基本的概念几乎没有改变，底层的 libuv 和事件循环还是原来的样子，主要模块的 API 也没什么大的变化。

另一方面，变动最多的大概是语法了，ECMAScript 沉寂数年之后，终于推出了重量级的新版本 ES2015，并且计划每年发布一个新版本。

换个角度来说，如果现在有一份使用 Node 最新版本写的代码，拿给还在使用 0.10 的开发者看，最大可能是对里面各种奇怪的关键字和语法感到疑惑。这本身就说明了 Node 发生了如此大的变动。

在笔者看来，Node 的发展进入了平台期，这意味着在一段时间内，Node 将维持现有的模样，或许会增加或修改一些语法，底层的 V8 也可能做一些大幅度的改进，但代码的基本结构不会有大的变动。

Node 的发展大致分为几个阶段：

- 第一个阶段：从诞生到分裂，大致 5 年的时间。
- 第二个阶段：从与 io.js 合并到 ES2015 标准正式落地，只有不到半年的时间。
- 第三个阶段：从 v6.0.0 全面支持 ES2015 至今，Node 较大的更新都是围绕着新的 ECMA201x 标准展开的。

所有的新与旧都是相对的，虽然在目前来看，本书记述的内容还算是比较新的，无论是 ES201x 的使用，还是 Koa2 框架的介绍，都属于同类书中较少涉及的领域，但要是再过几年的时间，本书的内容也会变得过时。

名称约定

为了便于区分，JavaScript 在本书中特指对 ECMAScript 的实现，除非特别注明，那么它代表了 ES5 的标准，并且同时适用于浏览器 JavaScript 和 Node。当有些代码和概念特指在浏览器端运行的 JavaScript 时，我们一律使用“前端 JavaScript”来称呼。

当使用 ECMAScript 这一称谓（例如 ES2017）时，大多数是谈论标准内容，不涉及具体的实现。

纠错

笔者毕竟能力有限，在出版本书的时候可能有没有注意到的错误，例如代码运行出错、概念上的不正确等，如果读者有相关的发现，请以“××章××小结代码/内容错误”为标题发邮件至笔者的邮箱 likaiboy_2012@126.com。

致谢

首先要特别感谢的是出版社的夏毓彦编辑，是他让我有机会梳理迄今为止对 Node 的心得，然后得以出版。

另一方面还要感谢我的母亲，当我第一次将自己准备写书的想法告诉她时，不出意外地，她开始怀疑起我的水平来，“就你还想出书？可不要误人子弟啊”。

正是这句话，不断提醒我对内容进行反复修改和验证。因为我意识到这和平时的博客文章不同，是更加严肃，并且对错误的容忍更低的作品。虽然她没有编程相关的经验，但我还是准备将第一本样书送给她，希望她能够阅读。

著者

2017 年 12 月

目 录

第 1 章
◂ 基础知识 ▸

——太阳底下没有新鲜事

本章主要介绍一些基本概念和 Node 的内部机制，如果读者对这部分暂时不感兴趣（事实上没人一开始就对这些概念感兴趣），可以先跳过这部分直接阅读第 2 章的内容。等对 Node 的使用有了大致了解之后，再回来看本章也不迟。

关于本章的内容，翻开任何一本经典的操作系统的教材都可以找到比本章更加全面和权威的描述（你可能有那么一瞬间后悔在学校没有认真掌握相关的知识，但这没关系，笔者也是这样），本章只负责介绍一些基础的概念，这有助于加深对 Node 的理解。

1.1 Node 是什么

在讨论所有 Node 相关的问题之前，我们必须要明确一个问题，Node 是什么？

这看起来是一个再简单不过的问题，但如果不看答案（官网描述）直接回答起来却不是很容易，刚接触的开发者可能会认为 Node 就是 JavaScript（笔者当初也是这么想的），这种看法并不准确。

1.1.1 Node 与 JavaScript

回过头来看官网的定义：

```
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.
```

Node 是一个 JavaScript（严格来说是 ECMAScript）运行时（runtime），所谓的 runtime 直译过来就是运行时组件，读者可以将其想象成一种编程语言的运行环境。这个运行环境包括了运行代码需要的编译器（解释器）以及操作系统的底层支持等。

对一门编程语言来说，相对于语法本身，更重要的是编译器（解释器）将如何对待这些语法。Node 底层使用 C++实现，语法则是遵循 ECMAScript 规范，如果创始人愿意，完全可以将 Node 创造成一个新的 Ruby 或者 Python 运行时，只不过名字大概就要改成 Node.rb 或者 Node.py 了。

这里有个扩展问题，编程语言是什么？

编程语言是一种抽象的规范，拿 C++来说，真正的 C++其实是厚厚的一摞文档，上面规

定了每一个语法细节以及每一个有效输入对应的输出值。而开发者平时所使用的 C++，例如 Visual C++，是 C++的一种实现。就好像数学概念里的正方形一样，我们找不到一个抽象的，纯粹的“正方形”，我们平时能看到的都是正方形的物体。

为什么是 JavaScript

Ryan Dahl 选择了 JavaScript 和 V8，前者提供了灵活的语法，后者为前者的运行提供了足够高的效率和实现，例如非阻塞 IO 和事件驱动等。

关于 Node.js 的历史，可以参考朴灵的文章《Node.js 与 io.js 那些事儿》，读者可上网自行搜索。

1.1.2 runtime 和 VM

1. runtime

最出名的 runtime 应该是 VC++，微软出品的这套应用程序组件可以使开发者编写的 C/C++语言程序在其中运行。VC++本身对 C++还做了一些扩展，用来开发 Windows 程序，例如 MFC 等。

VC++可以编译和执行用户编写的 C/C++代码，而开发者不考虑这背后到底是怎样实现的。站在开发者的角度来说，一个 X 语言的 runtime 表示开发者可以在这个 runtime 上运行 X 语言编写的代码，那么将这个概念扩大一些，Chrome 也是一个 JavaScript 运行时，它靠背后的 JavaScript 引擎来运行 JavaScript 代码。

runtime 可能会对编程语言做一些扩展，例如 Node 中的 fs 模块和 Buffer 类型就是对 ECMAScript 的扩展，此外，runtime 也不一定支持语言规范定义的全部特性。

如果没有 runtime 的支持，语言规范就和废纸无异。例如截至当前的时间点（2017 年 5 月），ES2015 中的 import 语句还没有被任何浏览器或者 Node 支持（不考虑 babel 等转换工具），那么 import 语句就仅仅是一个纸面特性而已。

反过来讲，就算一个特性没有体现在标准里，而大多数的运行时都支持它，也可以变成事实上的规范，例如 JavaScript（ES6 之前）的__proto__属性。

因此当我们谈论一门语言，往往是在谈论它的实现，再具体一点，就是指其运行时实现。例如下面的代码，我们无法分辨这是一段 Node 代码或是 JavaScript 代码，虽然它们都能产生相同的输出。

```
var name = "Lear";
function greet(name){
   console.log("I am",name);
}
greet(name);
```

如果一门 X 语言实现了 ECMAScript 规范，那么上面也可能是 X 语言的代码。

2. VM

VM 的概念比较广泛，通常可以认为是在硬件和二进制文件的中间层。

C++编译好的二进制文件可以直接被操作系统调用，而对 Java 而言，编译好的字节码是交给虚拟机来运行的，这样的好处是对开发者屏蔽了操作系统之间的差异，对于不同操作系统环境的具体处理交给了虚拟机来完成，从这个角度来看，VM 是对不同计算机系统的一种抽象。

1.2 Node 的内部机制

本节的内容会涉及一些操作系统的概念，在开始之前，这里有一些前提，记住这些前提会能让你更好地理解本节的内容：

- 在任务完成之前，CPU 在任何情况下都不会暂停或者停止执行，CPU 如何执行和同步或是异步、阻塞或者非阻塞都没有必然关系。
- 操作系统始终保证 CPU 处在运行状态，这是通过系统调度来实现的，具体一点就是通过在不同进程/线程间切换实现的。

1.2.1 何为回调

1. 回调的定义

一个回调是指通过函数参数的参数传递到其他代码的，某段可执行代码的引用。

说得通俗一点，就是将一个函数作为参数传递给另一个函数，并且作为参数的函数可以被执行，其本质上是一个高阶函数。在数学和计算机科学中，高阶函数是至少满足下列一个条件的函数：

- 接受一个或多个函数作为输入。
- 输出一个函数。

JavaScript 中一个很常见的例子就是 map 方法，该方法接受一个函数作为参数，依次作用于的数组的每一个元素。

```
[1,2,3].map(function(value){
    console.log(value);
})
```

可以用如图 1-1 所示来描述回调的过程。

图 1-1