Web开发技术丛书

Node.js实战

Learning Node.js:A Hands-On Guide to Building Web Applications in JavaScript

（美）温施耐德（Wandschneider,M.） 著

姚立 彭森材 译

ISBN：978-7-111-45969-9

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。

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

译者序

从1995年诞生至今，JavaScript已经走过了18个年头，很难想象，当初Brendan Eich在10天内写出来的语言竟然会成为如今最热门的语言之一。

这些年，JavaScript一直被当做Web浏览器端脚本使用，但在这个过程中，ECMA标准也在不断地演变，不断地加入新的特性；同时，各个厂商的JavaScript实现（特别是Chrome的V8）也在不断地提升性能和稳定性，这为后来Node.js的流行埋下了伏笔。

2009年，Node.js出现在大家的视野中，它在诞生伊始就获得了众多开发者的关注。由于使用了性能优越的Chrome浏览器V8引擎和全新的事件驱动、异步编程模式，它从一开始就显得“与众不同”。Node.js是Ryan Dahl发起的开源项目，这些新特性给它带来巨大的性能提升，并且大量的第三方模块也让其开发效率得到了极大提高。如今，诸多大型互联网公司在其产品中广泛使用Node.js，如LinkedIn、淘宝等；同时大部分云计算平台也支持部署Node.js应用。由此足以看出，Node.js是今后的发展趋势之一。

本书一共分为四大部分，由易入难，循序渐进，既适合Node.js新手入门，也能让使用过Node.js的开发者在阅读过程中收获“惊喜”。如果你是一名初学者，可以跟着书中的示例一步步成长；如果你是一名有经验的开发者，也可以从后面的章节中获取灵感。本书和其他介绍Node.js的书籍的最大不同在于，书中会包含大量的后端知识，如shell脚本等，这会让读者（特别是Web前端开发者）脱离原先的思维桎梏，从后端开发的角度来学习这个全新的开发平台。

本书由姚立、彭森材合作翻译完成。其中，彭森材翻译第2章、第4～6章、第9章和第10章；姚立翻译其余部分。在翻译本书的过程中得到了华章公司责任编辑关敏老师的帮助，她多方协调并给出中肯的建议，让译者收获颇丰，在此表示衷心感谢。

由于时间仓促，加之译者水平有限，书中难免会有疏漏之处，希

望读者能够提供反馈，不胜感激。

最后，我们要感谢家人，感谢他们的理解和支持，感谢他们在本书翻译过程中所做的一切。一条毛毯、一杯热茶，都是我们翻译的动力和源泉。感激、感恩！

前言

欢迎来到Node.js的世界。Node.js是专为编写网络和Web应用而生的全新平台，在过去的几年中，迅速积累了大量的人气并在开发社区拥有一大批拥趸。在本书中，我会介绍更多关于Node.js的知识，告诉你为什么它会如此迷人。我会教你如何快速上手并编写Node.js程序。很快，你会发现Node.js名字的叫法非常灵活，它经常被叫做Node，甚至被称为"node"。本书中我也会经常这么称呼。

为什么使用Node.js

Node.js的兴起有两个主要原因，我会在下面加以解释。

万维网

在过去，编写Web应用是一个相当标准化的流程。你有一个或多个服务器部署在机器上并监听某个端口（如HTTP的80端口），每当服务器接收到一个请求时，它会创建一个进程或者线程去处理和响应请求。处理这些请求会频繁地与外部服务进行通信，比如数据库、内存缓存、外部计算服务器或者仅仅是文件系统。当所有的工作最终结束之后，线程或者进程会返回到“可用”服务器池中，这样服务器就可以处理更多的请求了。

这是一个合理的线性过程，容易理解且极易编码。但是，有两个缺点一直困扰着这种模型：

1.每一个线程或进程都会消耗一些资源。在一些机器中，对于使用PHP和Apache构建的应用，每个进程甚至会消耗高达10～15MB的内存。特别是在大型服务器不断运行并创建线程来处理请求的时候，每一个线程都会消耗一些资源来创建新的堆栈和执行环境。很快，服务器就会陷入没有内存可用的境地。

2.在最常见的使用场景下，Web服务器会与数据库、缓存服务器、外部服务器或文件系统通信，这会消耗大量的等待时间，直到这些服务处理完并返回响应。当服务器等待响应的时候，当前线程就会

被“阻塞”，无法继续执行下去。因此，消耗的服务器资源和当前服务器进程或线程会被完全冻结，直到返回响应相关资源才会得到释放。

只有当外部组件返回响应之后，进程或线程才会继续完成处理，并把结果返回给客户端，然后重置并等待处理下一个请求。

因此，尽管这种模型非常容易理解和编码，但是一旦脚本花费大量的时间来等待数据库服务器结束查询，这种模型就略显低效——而这却是现代Web应用极其常见的应用场景。

当然，这种问题已经有了很多通用的解决方案。我们可以购买更强大、拥有更多内存的Web服务器；可以使用更小的轻量级服务器如lighttpd或者nginx来替代Apache这样功能强大的HTTP服务器；可以定制或者精简你喜欢的编程语言版本，比如PHP或Python（事实上，Facebook已经先行一步，并且开发出将PHP转换成原生C++代码的系统，以达到最大执行速度和最优代码规模的目的）；甚至，可以通过购置更多服务器的方式来提高所能容纳的并发连接数。

前沿技术

多年来，Web开发者一直在为优化服务器资源和提高并发数而努力；而这期间，其他一些有趣的事情已经悄然发生。

JavaScript，这门在Web浏览器中以客户端脚本语言而闻名于世（同时也经常遭人诟病）的古老（1995年左右面世）语言，已经重新焕发生机。诸多现代Web浏览器重构了JavaScript的实现，并添加了许多新的特性，让它变得更加强大和稳定。随着诸多浏览器客户端类库的出现，如jQuery、script.aculo.us、Prototype等，JavaScript编程变得有趣且富有成效。这些类库封装了原先臃肿的API，并添加了很多有趣、动态的效果。

同时，各大浏览器群雄割据的时代业已到来，谷歌的Chrome、Mozilla的Firefox、苹果的Safari和微软的IE浏览器一起争夺浏览器之王的桂冠。作为其中的一部分，这些公司加大了在JavaScript上的

研发比重，从而让Web应用可以更可靠地依赖脚本，并且拥有更强大的动态效果。特别是谷歌公司的Chrome浏览器使用的V8JavaScript引擎，性能尤为卓越，并且是开源软件，可供任何人使用。

天时地利人和，JavaScript的机遇随着开发人员的全新网络（Web）应用开发方式而出现。这，就是Node.js的新生。

Node.js的前世今生

2009年，一位名叫Ryan Dahl的研究员（后来他加入Joyent，该公司是一家坐落于加利福尼亚州的云计算和可视化服务公司）一直在寻求开发一种专为Web应用提供类似于Gmail推送服务的功能，但是一直没有找到一个完全合适的解决方案。最后，他把目光落到了JavaScript上，因为这门语言缺乏健壮的输入/输出（I/O）模型（这也就意味着他可以自己写一个全新的I/O系统），同时还有性能优越的V8引擎可供编程使用。

受Ruby和Python社区中类似项目的启发，他最终选择了Chrome的V8引擎和一个叫做libev的事件处理类库，并且很快推出了第一版的Node.js系统。Node.js的主要理论和创新就是它完全构建在事件驱动、非阻塞的编程模型之上。简而言之，你完全不会（或者极少）编写线程阻塞的代码。

一个Web应用——为了处理请求并返回响应——一般需要执行数据库查询。Web应用处理请求，并会在返回响应之后告诉Node.js如何处理。同时，应用代码也可以开始处理其他传入的请求，实际上，它可以处理任何需要处理的任务，比如清理系统数据或分析数据。

通过这种应用处理请求和工作方式的改变，可以简单地编写能够同时处理几百甚至几千个请求的Web服务器，同时也不用消耗太多的资源。Node是在单进程上运行的，并且Node代码一般也是单线程执行，因此，相较于其他平台而言，Node对资源的需求就小得多。

但是，这种执行速度和能力是有一些瑕疵的。因此，必须充分认识到它们，然后就可以开始小心翼翼地使用Node进行工作了。

首先，这种全新的模型机制不同于以往你所接触过的模型，因此你可能会感到困惑。在充分理解这些核心概念之前，你可能会对这些Node.js代码感到奇怪。本书中的大部分篇幅会讨论编程人员在迎接Node异步、非阻塞编程挑战时所使用的核心模式，并指导你如何开发自己的Node程序。

这种编程模型的另一个限制是：它真的是在为处理大量不同任务的应用程序服务，会同很多不同的进程、服务器或服务通信。当Web应用需要连接到数据库、缓存服务器、文件系统、应用服务器或其他服务时，Node.js便会大放异彩。但是另一方面，实际上它并不是那些需要做长时间精密计算的服务器的最佳运行环境。因此，单进程、单线程的Node模型在处理一个给定的请求时，如果该请求需要花费大量的时间生成一个复杂的密码摘要（password digest）或处理图像，就会带来问题。在那些需要更多的计算密集型工作的情况下，我们必须小心谨慎地处理应用程序使用资源的情况，甚至可以考虑把这些任务移植到其他平台，并把它们作为第三方服务供Node.js程序调用。

最后，Node.js是一个值得信赖的全新平台，并且正在积极发展中。它还没有（截至2013年2月）发布1.0版本，同时Node.js的版本更新非常频繁，甚至到了让人眼花缭乱的地步。

为了减少这些频繁的更新所引起的不确定性和烦恼，开发者已经使用标签来标识系统各个部分的不同稳定级别，从不稳定（Unstable）到稳定（Stable）再到锁定（Locked）。系统中稳定或锁定的部分几乎不会改动；如需改动，社区会通过大量的讨论以确定是否会产生太多问题。在你阅读本书的过程中，我们会指出哪些地方不太稳定，并提供相应的解决方法，从而减少因API改变而带来的危险。

好消息是，Node.js已经有一个庞大而活跃的用户社区和一堆邮

件列表、论坛及用户组，它们致力于促进平台的发展并提供力所能及的帮助。通过简单的谷歌搜索，就可以在几秒钟之内回答你99%的问题，所以，请不要害怕Node.js！

本书读者

本书的读者要喜欢编程，并且要熟悉至少一门主流编程语言的功能和语法，比如Java、C/C++、PHP或者C#等。读完本书之后，你不必成为一名专家，但你一定要高于“Y天精通X语言”的水平。

如果像我一样，你可能已经写过一些HTML/CSS/JavaScript代码，从而和JavaScript“打过交道”。但是你可能并不是非常熟悉JavaScript，只是单纯地从博客或者邮件列表中把代码复制出来。事实上，由于笨重的用户界面和令人沮丧的浏览器不匹配问题，你可能在一开始就对JavaScript没有好感。但是不用害怕——在本书第一部分结束之后，你对这门语言的糟糕印象从此会变成过去时。我希望你能放松心情，面露微笑，轻松愉快地编写你的第一个Node.js程序。

你还需要了解Web应用的基本工作原理：浏览器向远程服务器发送HTTP请求；服务器处理请求并返回表明成功或者失败的标识，返回对象中有可能会附带一些数据（比如渲染页面用的HTML代码或者请求返回的JSON数据）。在过去，你可能会在连接数据库服务器并发送查询请求后，等待返回数据集等。而在使用示例和程序来描述一些全新的概念的时候，我会详细讲解并帮助大家理解这些全新或者生僻的概念。

如何使用本书

本书实质上就是一本新手教程。我会尽量使用代码去解释原理，避免使用冗长的篇幅和晦涩的语言。因为我觉得好的解释说明应该是生动有趣的，如果你感兴趣的话，我还会告诉你一些资源或者一些其他的文档（当然这些并不是必需的）。

本书一共分为四大部分。

第一部分　基础篇——首先，你会学习如何安装并运行Node；

然后，会进一步了解JavaScript这门语言以及在V8和Node.js中使用的扩展；最后，会编写你的第一个Node.js应用。

第二部分　提高篇——在本篇中，你会开始学习开发更强大且有趣的应用服务器。同时，我会教你编写Node.js程序时会用到的一些核心概念和最佳实践。

第三部分　实战篇——在本篇中，你会看到一些强大的工具和模块，利用它们，你可以编写自己的Web应用，比如Web服务器的中间件及与数据库服务器的通信。

第四部分　进阶篇——最后，我会以一些高级特性的介绍结束本书，比如如何在生产环境中运行应用，如何测试代码以及如何使用Node.js编写命令行实用程序。

在阅读本书的过程中，最好花些时间打开编辑器，输入这些代码，然后看看这些代码在你当前的Node.js版本下是如何运行的。或者是在看书的时候，编写并开发你自己的小程序。在阅读本书的过程中，你会开发一个小小的相册应用，希望能给你编写其他应用提供一些灵感和想法。

下载源代码

本书示例的源代码可以在github.com/marcwan/LearningNodeJS上找到。希望你能下载代码并运行一遍。如果条件允许，不要放弃亲手把代码敲出来的机会，可以多尝试尝试。

GitHub上托管了功能齐全的示例代码，并且已经使用最新版的Node.js，在Mac、Linux和Windows下测试过。如果最新版的Node需要更新源代码，我会在GitHub上及时更新并做出注释，所以请确保每隔几个月就获取一次最新的代码。

如果你对本书中的示例代码有任何疑问，请在github.com/marcwan/LearningNodeJS上提交issue。我们会看到这些issue，并且会及时做出合理的解答。

致谢

我要感谢PHPTR各位朋友给予的支持和建议，是你们的帮助让本书和其他项目付诸实践。感谢本书文字编辑的杰出贡献。

非常感谢Bill Glover和Idriss Juhoor，是你们出色的技术和样式审校让本书愈加精彩。

最后，感谢至爱Tina，感谢一路有你。

第一部分　基础篇

第1章　入门

在本章中，我们会开始投入到相关的学习中，并在电脑上安装Node.js。在继续深入学习语言和编写网络应用之前，要确保Node.js能正常运行。本章的最后，应该已经成功地在电脑上安装Node.js并正常运行。我们还会使用一些小的测试程序来熟悉它，以及学会如何使用内置的Node调试器。

1.1 安装Node.js

首先，让我们来看看如何在Windows下安装Node。除非同时拥有Windows操作系统，否则Mac和Linux用户可以跳过本节，去阅读相应的章节。

1.1.1 在Windows上安装

要想在Windows电脑上安装Node.js，可以使用nodejs.org网站上提供的简易安装程序。可以访问下载页面，然后选择32位或者64位的Node.js安装程序（.msi）。当然，这完全取决于运行Node的操作系统。我们将会展示Windows 7/64位操作系统下Node.js的安装过程。

下载完MSI文件之后，双击该文件，将会看到如图1.1所示的安装程序界面。

阅读并同意授权协议之后，点击安装（Install）。安装过程非常快捷和方便，几十秒之后，点击完成（Finish）结束安装。

验证安装

为了测试Node.js是否正确安装，你可以使用Windows命令提示符cmd.exe（如果使用PowerShell，也是可行的）。如果你对此不熟悉的话，可以先找到开始（Start）/运行（Run），然后输入cmd，如图1.2所示。

图1.1　Windows下的Node安装程序

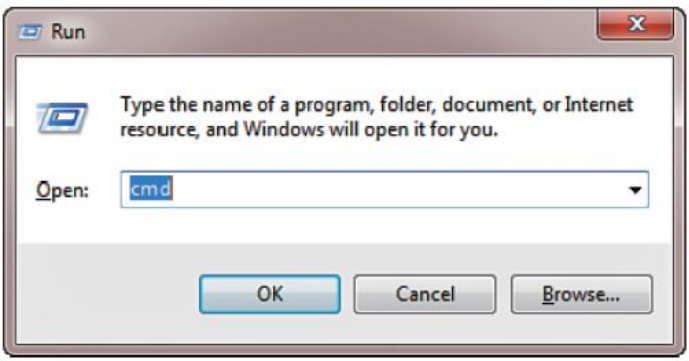

图1.2　启动Windows命令提示符

如图1.3所示，你会看到一个命令解释器。如果你想学习更多关于命令提示符的知识，可以在互联网上通过搜索“学习使用Windows cmd.exe”或者“PowerShell入门”（如果你使用的是Windows 7操作系统）来找到更多信息。

为了确认Node已经正确安装，在命令窗口中输入node--version，可以看到如图1.4所示的输出。

图1.3　Windows命令提示符

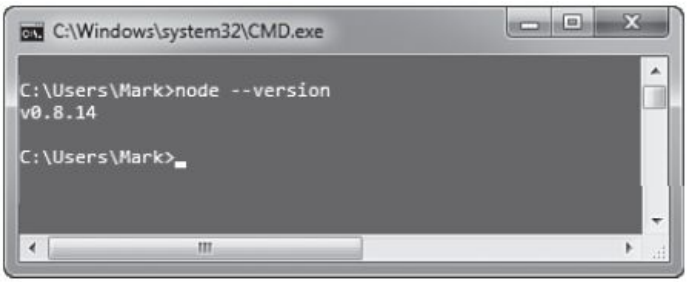

图1.4　验证Node是否正确安装，检测版本号

命令提示符窗口会打印出刚刚安装完的Node的版本号。（如果图中版本号和你看到的不一致，不用担心——事实上，如果一样的话，才会让我大吃一惊呢！）如果你没有看到版本号或者你看到的输出是“'node'不是外部或内部命令”（见图1.5），那肯定是出问题了，你需要进行如下操作。

- 进入控制面板/程序中，查看程序有没有安装成功。如果没有，则再次安装，并且注意安装过程，有可能会出现安装出错。

- 进入Program Files\nodejs目录，确保node.exe文件存在。如果不存在，尝试再次安装（如有需要，请先卸载旧版本）。

- 确保node.exe在PATH环境变量中。在命令提示符中可以看到，Windows有一个目录列表，当输入程序名称时，会在该列表中进行搜索。只要在命令提示符窗口中输入path就可以看到该列表，类似图1.6所示。尤其要仔细看后面两个高亮显示的目录名，如果Node.js安装正确的话，就会在PATH中看到相似的目录名。

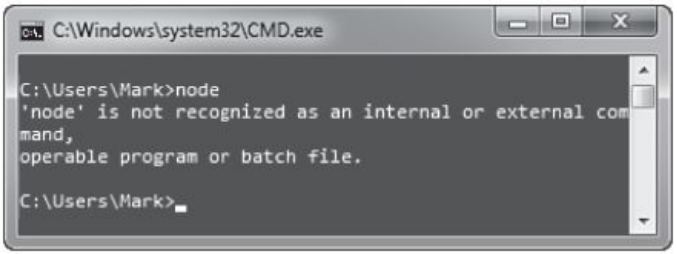

图1.5　Node不是内部或外部命令

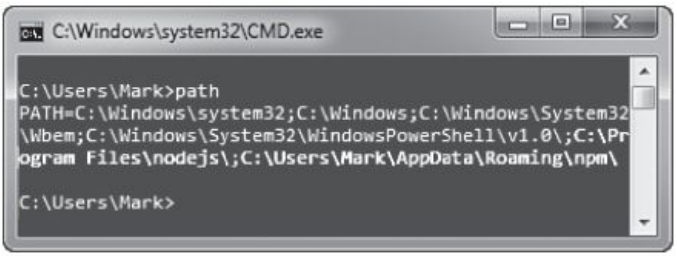

图1.6　检测PATH环境变量

■ 如果Program Files\nodejs或者Users\..\AppData\..\npm不在环境变量PATH中，而这些文件夹却真实存在，你可以手动将它们添加到PATH环境变量中，可以在系统控制面板窗口中设置它。点击高级系统设置，然后是环境变量。将npm文件夹的路径（类似于C:\Users\UserName\Local\npm）添加到“用户名”的用户变量下的PATH变量中。将Node.js文件夹（类似于C:\Program Files\nodejs）添加到系统变量下的PATH变量中。注意：npm文件夹可能会在Username\Remote\npm路径下，而不是Username\Local\npm下，这完全取决于你的电脑是如何设置的。

在确认node.exe正确安装和运行之后，就可以开始你的JavaScript之旅了。

1.1.2　在Mac上安装

在Mac上安装Node.js有两种不同的方式——使用PKG安装程序或者编译源代码——这里，我只想介绍前一种安装方式，这是目前最快最便捷的安装方式。如果你倾向于以编译源代码的方式安装Node.js，我已经在本书的git资源中准备了安装指南。

使用PKG安装程序

目前为止，在运行OS X的苹果Mac电脑中安装Node.js最快的方式就是下载并运行Node PKG安装程序，该安装程序文件在nodejs.org网站中提供下载。

下载完安装程序以后，双击程序，你会看到如图1.7所示的安装界面。我倾向于使用默认安装，因为我想使用所有的组件，并且这个默认安装路径（/usr/local/bin）也正是我想要的，建议你也这样做。

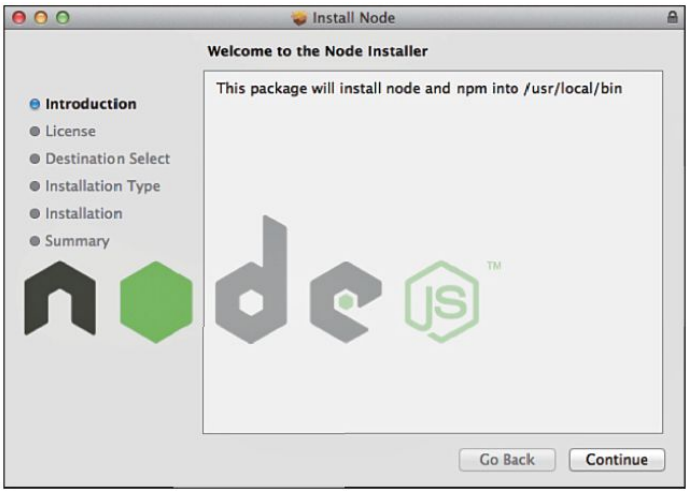

图1.7　在Mac上运行Node.js的PKG安装程序

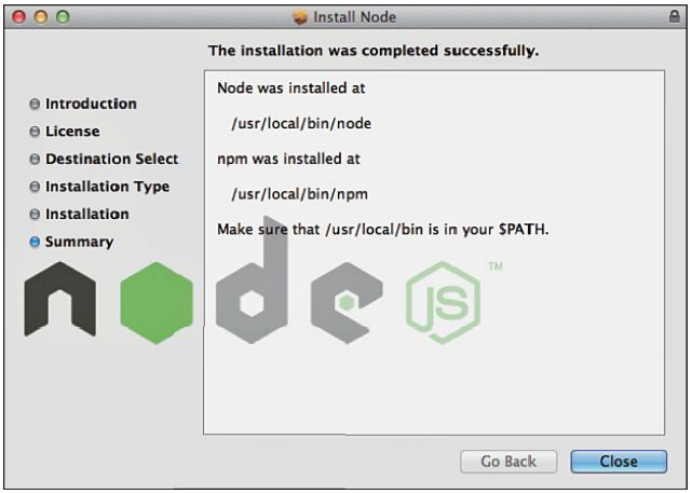

图1.8　确认path环境变量正确设置

当安装结束之后，可以看到如图1.8所示的界面，正如图中包安装程序解释的那样，一定要确认/usr/local/bin在PATH环境变量中。你可以打开Terminal.app程序（在/Applications/Utilities下启动终端），在终端窗口中，输入：

```
echo $PATH
```

在我的电脑中，输出如下：

```
/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/Users/marcw/bin:/usr/local/git/bin
```

你可以看到/usr/local/bin就在PATH环境变量中；如果没有，就需要编辑~/.bash_profile文件（如果不存在，可以创建该文件），添加下面这段脚本：

```
PATH=${PATH}:/usr/local/bin
```

关闭该终端窗口，并重新启动一个新的终端窗口，验证/usr/local/bin已经在PATH环境变量中，现在可以输入：

```
node --version
```

可看到如下所示内容：

```
client:LearningNode marcw$ node --version
v0.10.3
client:LearningNode marcw$
```

如果版本号和你电脑上的版本号不一致，请不用担心，这是正常情况。

在确认node程序已经正确安装并运行之后，就可以编写JavaScript代码了。

1.1.3　在Linux上安装

安装Node.js

尽管许多的Linux发行版已经预置了Node.js安装包，但我还是更喜欢手动在Linux上安装Node。安装过程非常简单，所以无需过多额外的工作。

在Linux上安装命令行编译器

要想在Linux上编译源码包，首先要确保已经安装了命令行编译器工具。要安装编译器工具，可以输入：

```
g++
```

如果在终端窗口中看到如下所示：

```
marcw@hostname:~$ g++
g++: no input files
```

则编译器已经成功安装，可以编译源码了。否则，你会看到：

```
marcw@hostname:~$ g++
-bash: g++: command not found
```

你需要清楚如何在你当前的Linux发行版上安装编译工具。在绝大部分Ubuntu Linux版本中，可以如下所示使用apt-get工具。如果要安装编译工具，则需要知道在某个特定版本下应该安装哪些安装包。对于Ubuntu 8，必须安装：

```
# apt-get install build-essential libssl-dev libxml2-dev autoconf2.13
```

而在Ubuntu 10上，则需要安装：

```
# apt-get install pentium-builder g++ libssl-dev libxml2-dev autoconf2.13
```

当所有安装结束之后，可以再次输入：

```
g++
```

一切运行正常。

下面的安装指南已经在Ubuntu Linux版本中使用多年，前提是你使用（ba）sh作为默认的shell解释器。首先，创建一个暂存空间用来下载和编译文件：

```
cd
mkdir -p src/scratch
cd src/scratch
```

下一步就是下载并解压node源代码，可以使用curl或者wget命令：

```
curl http://nodejs.org/dist/v0.10.1/node-v0.10.1.tar.gz -o node-v0.10.1.tar.gz
tar xfz node-v0.10.1.tar.gz
cd node-v0.10.1
```

接下来运行配置脚本来为编译做准备：

```
./configure
```

可以使用默认的/usr/local作为安装目录，因为这是运行这个软件的最佳位置。如果想把软件安装到其他目录下，可以指定--prefix来修改配置脚本，如下所示：

```
./configure --prefix =/opt/nodejs
```

这个配置脚本会执行得非常快，并会在结束的时候打印出一些JSON信息。现在，你可以编译Node了。输入下面的命令，然后喝一杯咖啡，静静等待（也许是两杯哦，速度快慢完全取决于电脑的性能）。

```
make
```

在成功编译完成之后（如果编译失败，可以谷歌搜索下这个问题，因为一般情况下，不只你一个人遇到相同的问题），可以将软件安装到指定的目录下（如果没有指定，则默认目录是/usr/local）：

```
sudo make install
```

当完成这一切，输入：

```
node --version
npm --version
```

可以得到如下输出结果：

```
marcw@hostname:~$ node --version
v0.10.1
marcw@hostname:~$ npm --version
1.1.65
```

1.2 "Hello World!"

在电脑上使用Node.js有两种主要方式：直接使用Node Shell或者保存JavaScript文件后运行。

1.2.1 Node Shell

运行Node.js的第一种方式就是使用Node Shell，这种方式一般会被称为Node REPL——REPL是Read-Eval-Print-Loop的缩写。这是一种快速测试Node的途径。如果你不能准确记住某个函数的用法，可以使用REPL，把东西输入进去，看看会发生什么。

如果想启动Node Shell，可以在任何shell中输入node：

```
client:node marcw$ node
>
```

Shell会返回>符号，然后你就可以输入一些代码了：

```
> console.log("Hello World!");
Hello World!
undefined
>
```

第一行输出就是刚才输入的代码执行之后的结果。在上述示例中，我们使用了Node的全局变量console，并使用log函数打印出"Hello World!"（下一章中会介绍更多关于console和其他全局变量的内容）。该语句的期望结果是，打印出"Hello World!"。

最后一行输出结果往往是最后一行语句的返回值。每一个语句、函数调用或者表达式都有一个相关联的值，这个值会在Node shell中打印出来。如果表达式或者被调用的函数没有任何返回值，则会返回一个特殊的值undefined。

如果想要退出REPL，只需要按下Ctrl+D（在Windows平台下也一样）。