Building APIs with Node.js

Caio Ribeiro Pereira

Apress®

Building APIs with Node.js

Caio Ribeiro Pereira

Apress®

Building APIs with Node.js

Caio Ribeiro Pereira
São Vicente - SP, São Paulo, Brazil

ISBN-13 (pbk): 978-1-4842-2441-0 ISBN-13 (electronic): 978-1-4842-2442-7
DOI 10.1007/978-1-4842-2442-7

Library of Congress Control Number: 2016961314

Managing Director: Welmoed Spahr
Acquisitions Editor: Louise Corrigan
Development Editor: James Markham
Technical Reviewer: Massimo Nardone
Editorial Board: Steve Anglin, Pramila Balan, Laura Berendson, Aaron Black, Louise Corrigan, Jonathan Gennick, Todd Green, Celestin Suresh John, Nikhil Karkal, Robert Hutchinson, James Markham, Matthew Moodie, Natalie Pao, Gwenan Spearing
Coordinating Editor: Nancy Chen
Copy Editor: Teresa F. Horton
Compositor: SPi Global
Indexer: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

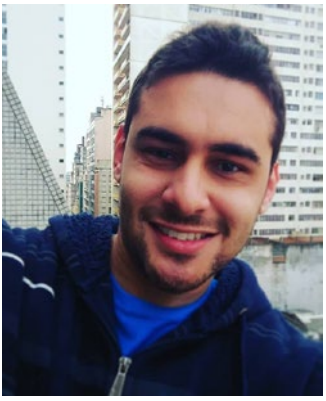
Printed on acid-free paper

I dedicate this book to my family who always supports me and has motivated me since the beginning of my life.

Contents at a Glance

Contents

About the Author



Caio Ribeiro Pereira is a software engineer from Brazil who love works with Node.js, JavaScript, Meteor, Ruby On Rails, DevOps, and front-end stuffs. He has a bacharelor's degree in Information Systems, author of the blog https://udgwebdev.com, creator of the DevFreeBooks (https://devfreebooks.github.io) and DevFreeCasts (https://devfreecasts.github.io), and is actively engaged in many local meet-ups including NodeBr, DevInSantos, Meteor Brazil and JavaScript Brazil.

About the Technical Reviewer

Massimo Nardone has more than 22 years of experience in security, Web/ and mobile development, and cloud and IT architecture. His true IT passions are security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years. He holds a master's of science degree in Computing Science from the University of Salerno, Italy.

He has worked as a project manager, software engineer, research engineer, chief security architect, information security manager, PCI/SCADA auditor, and senior lead IT security/cloud/SCADA architect for many years.

His technical skills include: security, Android, cloud, Java, MySQL, Drupal, Cobol, Perl, Web and mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, and more.

He currently works as Chief Information Security Officer for Cargotec Oyj.

He worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas).

Massimo has reviewed more than 40 IT books for different publishing companies and he is the coauthor of *Pro Android Games* (Apress, 2015).

Acknowledgments

Special thanks go to Mrs. Charlotte Bento de Carvalho, my cousin Cláudio Souza, my friends Leandro Alvares da Costa and Bruno Alvares da Costa, Paulo Silveira, and Adriano Almeida. They have had an important role in my life, influencing me to be who I am today, and consequently, to write this book.

Thanks to all the readers from Underground WebDev (https://udgwebdev.com); after all, the essence of this book is based on many posts from this blog.

Finally, thank you, dear reader, for purchasing this book.

Introduction

We live in a time in which the majority of users use different types of devices to connect to the Internet. The most popular devices are smartphones, tablets, and notebooks. Developing systems for different types of devices requires the work of building web services, also known by the name of application program interfaces (APIs).

Basically, the APIs are back-end systems that are designed to work only with data in a centralized manner, allowing client-side applications to be developed separately, to have a unique interface to the final user. These client-side applications are typically mobile apps, desktop applications, or web apps.

Since 2010 Node.js has increasingly proven to be an excellent platform to solve many problems, especially for building REST APIs. The single-thread architecture that performs nonblocking I/O running on top of JavaScript, which is a ubiquitous language in all current browsers, showed good performance in the processing of many kinds of applications. Some large companies, such as LinkedIn and PayPal, have significantly reduced expenses with servers migrating some of their projects to Node.js.

In addition, another advantage of using Node.js, which captivated many developers, is the short learning curve. After all, anyone who has worked with web development already has at least a basic understanding of the JavaScript language.

Who Is This Book For?

This book is intended for developers who have at least a basic knowledge of JavaScript and, especially, those who understand object-oriented programming (OOP), a little bit of client-server architecture, and those with an understanding of the main characteristics of REST APIs.

Mastering these concepts, or at minimum a basic knowledge of them, is essential to fully understand this book.

All the code in this book is written using the latest JavaScript implementation, the ECMAScript 2015 (also called ECMAScript 6, or ES6).

How Should I Use This Book?

Throughout the book, many concepts and codes are going to be presented for you to learn through practice all the theoretical parts of this book. It will guide you in a didactic way in the development of two projects (an API and a web client application), which in the end are both integrated to work as a single project.

It is highly recommended that you follow the book's instructions step by step, so in the end you can complete the project correctly.

CHAPTER 1

Introduction to Node.js

What Is Node.js?

Node.js is a low-level, highly scalable platform. It was created explicitly to be an experiment in asynchronous processing. Using Node.js, you will program directly with many network and Internet protocols or use libraries that have access to operating system (OS) resources. To program in Node.js you only need to master JavaScript language; that's right, only JavaScript! The JavaScript runtime used in this platform is the famous Javascript V8, which is also used in Google Chrome.

Main Features

Single-Thread Architecture

A single-thread architecture works with only the main thread instance for each initiated process. If you are used to programming in multithread architectures, like Java or .NET to use the concurrency processing, unfortunately that won't be possible with Node.js. However, you can work with parallel processing using multiple processes.

For example, you can use a native library from Node.js called `clusters` that allows you to implement a network of multiple processes. In Node.js you can create *N-1 processes* to work with parallel processing, as the main process (or master process) works on balancing the load among the other processes (or slave process). If your server has multiple cores, applying this technique will optimize the usage of the processing.

Another way is to use asynchronous programming, which is one of the main resources from JavaScript. The asynchronous functions on Node.js work as *nonblocking I/O:* If your application has to read a huge file it will not block the CPU, because the I/O operation runs out of a thread pool, allowing your application to continue to process other tasks requested by other users.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-2442-7_1) contains supplementary material, which is available to authorized users.

C. R. Pereira, *Building APIs with Node.js*, DOI 10.1007/978-1-4842-2442-7_1

Event-Loop

Node.js has an *event-oriented paradigm*. It follows the same event-oriented philosophy on the JavaScript client side; the only difference is the type of event, meaning that there are not events such as mouse's `click`, `key up` from the keyboard, or any other events from web browsers. In fact, Node.js works with I/O events; for example, `connect` from a database, `open` a file or read `data` from a stream, and many others.

The event-loop (Figure 1-1) is the agent responsible for noticing and emitting events. Simply speaking, it is an infinite loop that in each iteration verifies in the event queue if some event was triggered. When an event is triggered, the event-loop executes it and sends it to the queue of executed events. When an event is running, we can write any business logic on it inside the callback function commonly used in JavaScript.

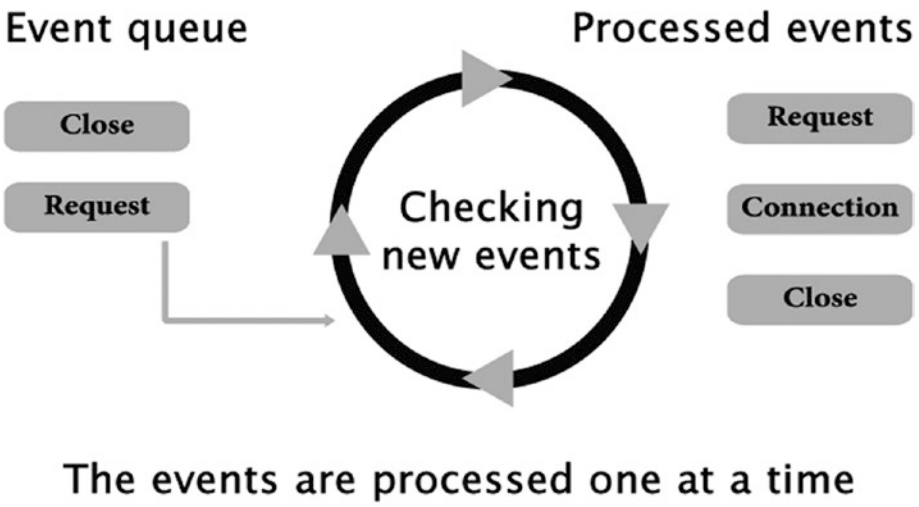


Figure 1-1. *The event-loop process*

Why Do I Need to Learn Node.js?

- *JavaScript everywhere*: Node.js uses JavaScript as a server-side programming language. This feature allows you to streamline your learning process; after all, the language is the same as the JavaScript client side. Your main challenge in this platform is to learn how to work with the asynchronous programming to make the most of this technique. Another advantage of working with JavaScript is that you can keep a project that is easy to maintain, as long as you know how to write codes using JavaScript. You are going to easily find professionals for your projects and you're going to spend less time studying a new server-side language. A technical JavaScript advantage in comparison with other back-end languages is that you will no longer use specific frameworks to parse JavaScript Object Notation (JSON) objects; after all, the JSON from the client side will be same as on the server side. There are also cases of applications running databases that persist JSON objects, such as the popular NoSQL: MongoDB, CouchDB, and Riak. An important detail is that Node.js uses many functionalities from ECMAScript 6, allowing you write elegant and robust JavaScript code.
- *Active community*: This is one of the biggest advantages in Node.js. Currently, there are many communities around the world striving to make Node.js a huge platform, either writing posts and tutorials, giving speeches at events, or publishing and maintaining new modules.
- *Active open source modules*: Today there are more than 300,000 open source modules published into NPM that can be used instantly for Node.js projects.
- *Ready to real time*: Node.js has become popular thanks to some frameworks for real-time communications between client and server. SockJS and Socket.IO are good examples. They are compatible with the WebSockets protocol and they allow you to traffic data through a single bidirectional connection, treating all data by JavaScript events.
- *Big players*: LinkedIn, WalMart, Groupon, Microsoft, Netflix, Uber, and Paypal are some of the big companies using Node.js, and there are many more.

Conclusion

Up to this point, I explained the main concepts and advantages of using Node.js. In the next chapters, we are going to see how it works in practice, with a single condition: Open your mind to new ideas and read this book with excitement

CHAPTER 2

Setting Up the Environment

In this chapter, we are going to install Node.js on the most used OSs (Windows, Linux, and MacOSX). However, the entire book uses MacOSX as the default OS. Don't worry about the differences among these OSs, because all examples in this book are compatible with all the three OS platforms.

Node.js Standard Installation

Setting up the Node.js environment is much the same despite the OS. Just a few procedures will be different in each OS, especially for Linux, because in the Linux environment you can compile an application or install it using a package manager, but this is not a major change.

The first step is to access the official Node.js web site at `http://nodejs.org`. On the home page, shown in Figure 2-1, v6.9.1 LTS Stable to download the latest compatible version for Windows or MacOSX. If you use Linux, I recommend you read the quick wiki at `https://github.com/nodejs/node/wiki/Installing-and-Building-Node.js`, which explains how to compile and install Node.js in some popular Linux distros.

C. R. Pereira, *Building APIs with Node.js*, DOI 10.1007/978-1-4842-2442-7_2

Figure 2-1. *Node.js home page*

After the download, install the software. If you use Windows or MacOSX, just click through the installation wizard screens until the installation is completed; after all, there is no specific configuration required.

To test if everything is running correctly, just open the terminal (for Linux or MacOSX), command prompt, or Power Shell (for Windows) and type this command:

```
1   node -v && npm -v
```

This checks the version of Node.js and NPM. This book uses Node v6.9.1 LTS and NPM 3.10.8**.**

About io.js and Node.js Merge

As of September 2015, Node.js was upgraded from 0.12.x to version 4.0.0 because of the merge with io.js. The io.js was a fork built and maintained by the Node.js community. They worked to include some features from ECMAScript 6 implementation, in addition to making several improvements that were slow on Node.js. When io.js reached version 3.0.0 and both groups decided to merge io.js into Node.js, the result was the Node.js v4.0.0. This merger not only gave a huge upgrade to this platform, but also resulted in a more stable and reliable version of Node.js.

■ **Note** This book uses Node v6.9.1 LTS, with a lot of ES6 codes. You can learn more about ECMAScript 6 at http://es6-features.org.

Node.js Installation Via NVM

■ **Tip** In this section, we are just going to explore an alternative way to set up Node.js using a version manager. Feel free to skip this step if you don't need to or don't want to learn how to use the NVM.

Just like Ruby language has the Ruby Version Manager (RVM) to manage multiple Ruby versions, Node.js also has a manager, the Node Version Manager (NVM).

NVM is a perfect solution to test your projects in different Node.js versions. It is also useful for people who like to test unstable or the latest versions. NVM has great advantages: It is practical, it is easy to use, it can uninstall a Node.js version with a single command, and it will save you time searching and installing a versions of Node.js. It is a good alternative, especially on Linux, because its native package managers are frequently outdated, which invalidates a new Node.js version to be installed.

Set Up NVM

In few steps you can set up NVM for a MacOSX or Linux system; you just need to run this command:

```
1   curl https://raw.githubusercontent.com/creationix/nvm/v0.32.1/install.
sh | bash
```

Unfortunately, the official version of NVM is not available for Windows, but there are alternative projects created by the community. Here are two similar alternatives for Windows:

- *NVM-Windows*: https://github.com/coreybutler/nvm-windows
- *Nodist*: https://github.com/marcelklehr/nodist

Both have a similar command-line interface (CLI) inspired by NVM.

Top NVM Commands

As as a sort of recipe, here is a list of the top commands that will be essential to manage multiple versions or at least keep your environment updated with the latest version.

- `nvm ls`: Lists all versions installed.
- `nvm ls-remote`: Lists all the Node.js versions available to download and install.
- `nvm install vX.X.X`: Downloads and installs a version.
- `nvm uninstall vX.X.X`: Uninstalls an existing version.
- `nvm use vX.X.X`: Chooses an existing version to use.
- `nvm alias default vX.X.X`: Chooses an existing version to be loaded by default in the OS boot time.

■ **Note** Replace the `vX.X.X` with the Node.js version of your choice, such as `v6.9.1`.

Installing Node.js Via NVM

To install Node.js via NVM, you need to run the following commands:

```
1   nvm install v6.9.1
2   nvm use v6.9.1
3   nvm alias default v6.9.1
```

After running these commands, Node.js will be installed and ready to use.

Conclusion

Congratulations! Now, besides having Node.js installed and working, you also learned a new way to manage multiple Node.js versions via NVM. In the next chapter, we are going to explore some important things about Node Package Manager (NPM), so keep reading, because the fun is just starting!