Extracted from:

# Node.js 8 the Right Way

Practical, Server-Side JavaScript That Scales

The Pragmatic Bookshelf

Raleigh, North Carolina

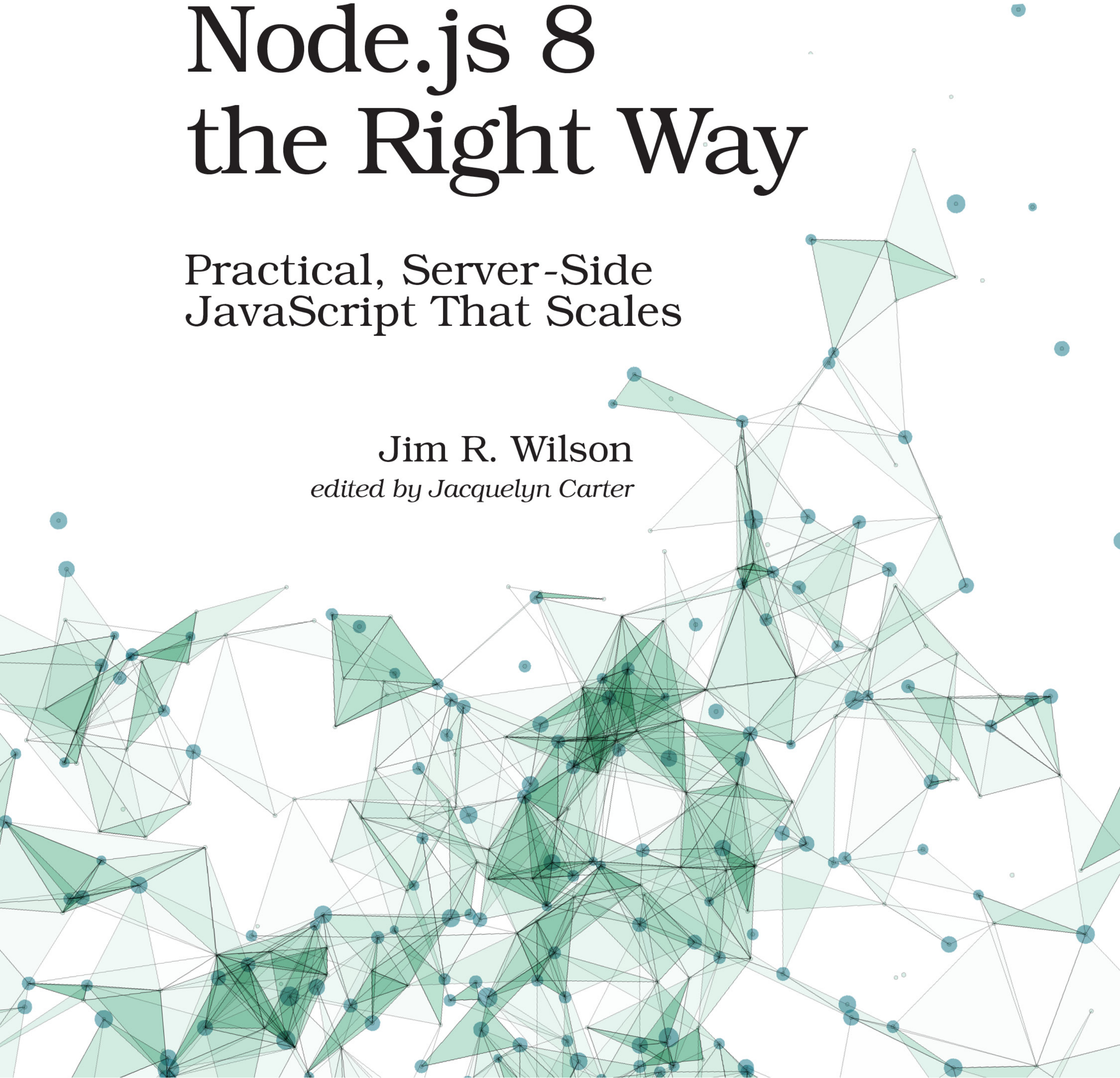
The Pragmatic Programmers

# Node.js 8 the Right Way

Practical, Server-Side JavaScript That Scales

Jim R. Wilson

*edited by Jacquelyn Carter*



# Node.js 8 the Right Way

Practical, Server-Side JavaScript That Scales

Jim R. Wilson

The Pragmatic Bookshelf

Raleigh, North Carolina

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Brian MacDonald
Supervising Editor: Jacquelyn Carter
Indexing: Potomac Indexing, LLC
Copy Editor: Candace Cunningham
Layout: Gilson Graphics

Printed in the United States of America.
ISBN-13: 978-1-68050-195-7
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—January 2018

# Preface

In recent years, two big shifts have happened in the practice of writing software—and Node.js has been at the forefront of both.

First, software is becoming increasingly *asynchronous*. Whether you're waiting on a Big Data job, interacting with end users, steering a quadcopter, or simply responding to an API call, chances are you'll need asynchronous programming techniques.

Second, JavaScript has quietly become the world's standard code-execution environment. It's everywhere: in web browsers, modern NoSQL databases, DIY robots, and now on the server as well.

Node.js is an integral part of these trends, and it has taken off in a big way.

## Why Node.js the Right Way?

Way back in March of 2010, I gave a lightning talk titled "Full-Stack JavaScript" at the NoSQL Boston conference. Back then, and even more so now, I knew that using JavaScript for every layer of the application stack was not only possible, but was a great way to reduce software complexity.

When each layer of your stack speaks JavaScript, you sidestep impedance mismatches and facilitate code reuse. Node.js is an important piece of the puzzle, filling the middle space between your front-end user-facing code and your data-storage layer.

The *Right Way* in this book's title refers to both the process of learning Node.js and the practice of writing Node.js code.

### Learning Node.js

As with any growing technology, there are plenty of resources available for learning Node.js. Unfortunately, many of those resources are narrowly focused on serving up web resources.

The web is great, but it's not enough, and it's not the whole story of Node.js. Node.js is about more than just serving web apps, and this book treats it that way.

*Node.js 8 the Right Way* teaches you the concepts you'll need to be an effective Node.js programmer, no matter what kinds of programs you need to write.

### Writing Node.js

One thing I love about JavaScript is that there are seven ways to do anything. There's breathing room, where developers can explore and experiment and find better approaches to everything.

The community of Node.js developers, the conventions in Node.js development, and even the semantics of the JavaScript language itself are all rapidly evolving. With eyes to the near future, the code examples and recommendations in this book reflect current best practices and standards.

## What's in This Book

This book is for intermediate to advanced developers who want to learn how to write asynchronous JavaScript for the server using Node.js. Some prior JavaScript experience will definitely help, but you don't have to be an expert.

The book proceeds in three parts, outlined here briefly.

### Part I: Getting Up To Speed on Node.js 8

Part I is about getting you up to speed on Node.js 8. You'll write Node.js programs that use core modules—and a few external modules as well—to do things like interact with the filesystem, spin up a cluster of worker processes, and manage network connections.

#### Getting Started

Chapter 1, *Getting Started*, on page ?, introduces the Node.js event loop, explaining how it empowers Node.js to be highly parallel and single-threaded at the same time. This chapter also outlines the five aspects of Node.js development that frame each subsequent chapter and has some brief instructions on getting Node.js installed on your machine.

#### Wrangling the File System

In Chapter 2, *Wrangling the File System*, on page ?, you'll start writing Node.js programs. If you've done any server-side programming in the past, chances are you've had to access a filesystem along the way. We'll start in this familiar domain, using Node.js's filesystem tools to create asynchronous, nonblocking file utilities. You'll use Node.js's ubiquitous EventEmitter and Stream classes to pipe data, and you'll spawn and interact with child processes.

### Networking with Sockets

We'll expand on those concepts while exploring Node.js's network I/O capabilities in Chapter 3, *Networking with Sockets*, on page ?. You'll create TCP servers and client programs to access them. You'll also develop a simple JSON-based protocol and a custom class for working with these messages. To develop unit tests for the code, you'll use Mocha, a popular Node.js test harness.

### Connecting Robust Microservices

Then, in Chapter 4, *Connecting Robust Microservices*, on page ?, we'll branch away from the Node.js core and into the realm of third-party libraries. You'll use npm to import ØMQ (pronounced "Zero-M-Q")—a high-efficiency, low-latency library for developing networked applications. With ØMQ, you'll develop programs that communicate using several important patterns, such as publish/subscribe and request/reply. You'll create suites of programs that work together in concert, and you'll learn the clustering tools to manage them.

## Part II: Working with Data

In Part II, you'll work with real data and lay the groundwork for an end-to-end application. This starts with processing data files in a testable way. You'll also learn to compose rich command-line utilities using Node.js and interact with HTTP services.

### Transforming Data and Testing Continuously

Chapter 5, *Transforming Data*, on page ?, kicks off an ongoing project that spans Part II and Part III. You'll download the catalog from Project Gutenberg, an online resource for ebooks in the public domain. Using a module called Cheerio, you'll write Node.js code to parse the data files and extract the important fields. You'll use npm, Mocha, and an assertion library called Chai to set up continuous testing, and you'll learn to use Chrome DevTools for interactive debugging.

### Commanding Databases

In Chapter 6, *Commanding Databases*, on page ?, you'll insert the extracted Project Gutenberg catalog into an Elasticsearch index. To get this done, you'll write a command-line utility program called esclu using a Node.js module called Commander. Since Elasticsearch is a RESTful, JSON-based datastore, you'll use the Request module to interact with it. You'll also learn to use a handy and powerful command-line tool called jq for manipulating JSON.

## Part III: Implementing an Application

Part III is where everything comes together. You'll develop web services that mediate between your API users and your back-end data services. End users don't interact directly with APIs, though, so for that you'll implement a beautiful UI. At the end, you'll tie it all together with session management and authentication.

### Developing RESTful Web Services

Node.js has fantastic support for writing HTTP servers, and in Chapter 7, *Developing RESTful Web Services,* on page ?, you'll do exactly that. You'll use Express, a popular Node.js web framework for routing requests. We'll dive deeper into REST semantics, and you'll use Promises and async functions for managing code flows. In addition, you'll learn to configure your services using the nconf module, and keep them running with nodemon.

### Creating a Beautiful User Experience

With the web services in place, in Chapter 8, *Creating a Beautiful User Experience,* on page ?, you'll craft a front end for them. You'll learn how to assemble a front-end project using a Node.js-based build tool called webpack, along with a host of peer-dependency plugins for it. You'll transpile your code for consumption by the browser using TypeScript, a language and transpiler from Microsoft that features inferred type checking. To make your UI look modern and fabulous, you'll bring in Twitter's Bootstrap styling framework, and implement templating with Handlebars.

### Fortifying Your Application

Chapter 9, *Fortifying Your Application,* on page ?, is where everything comes together. You'll combine the user experience with the web services from the previous two chapters for an end-to-end solution. Using Express middleware, you'll create authenticated APIs and implement stateful sessions. You'll also learn how to use npm's shrinkwrap option to insulate yourself from upstream module changes.

## Developing Flows with Node-RED

After Part III concludes, there's a special bonus chapter on Node-RED. Chapter 10, *BONUS: Developing Flows with Node-RED,* on page ?, walks you through this clever visual editor for designing event-based code flows. It ships directly with Raspbian, the default operating system of Raspberry Pi.

Using Node-RED, you can quickly stub out exploratory HTTP APIs. I'll show you how!

### Appendices on Angular and React

In case you're interested in using the front-end frameworks Angular and React, Appendix 1, *Setting Up Angular*, on page ?, and Appendix 2, *Setting Up React*, on page ?, show you how to integrate them with webpack and Express. The appendixes will help you put the pieces in place to start experimenting, but they don't take the place of a good tutorial on how to fully develop with them.

## What This Book Is Not

Before you commit to reading this book, you should know what it *doesn't* cover.

### Everything About Everything

At the time of this writing, npm houses more than 528,000 modules, with a growth rate of more than 500 new modules per day.[1] Since the ecosystem and community around Node.js is so large and still growing so rapidly, this book does not attempt to cover everything. Instead, this short book teaches you the essentials you need to get out there and start coding.

In addition to the wealth of Node.js modules available, there's the added complexity of working with non-Node.js services and platforms. Your Node.js code will invariably act as an intermediary between various systems and users both up and down the stack. To tell a cohesive story, we'll naturally only be able to dive deep on a few of these, but always with an eye to the bigger picture.

### MEAN

If you're looking for an opinionated book that focuses only on a particular stack like MEAN (Mongo, Express, Angular, and Node.js), this is not it! Rather than prescribe a particular stack, I'll teach you the skills to put together the Node.js code, no matter which back end you connect to or front end you choose to put on top.

Instead of MongoDB, I've selected Elasticsearch to back the projects in this book because it's increasingly popular among experienced Node.js developers,

1. http://www.modulecounts.com/

as evidenced by a 2016 survey by RisingStack.[2] Moreover, with its REST/JSON API, Elasticsearch offers a way to ease into HTTP services as a consumer before jumping into writing your own.

This book also shies away from front-end JavaScript frameworks. The two most popular front-end frameworks at the time of this writing are React, by Facebook,[3] and Angular, by Google.[4] This book covers neither of them in detail, by design. They both deserve more coverage than fits in these pages.

I want you to be the best Node.js coder you can be, whether you use any particular database or front-end framework.

## JavaScript Beginner's Guide

The JavaScript language is probably the most misunderstood language today. Although this book does discuss language syntax from time to time (especially where it's brand-new), this is not a beginner's guide to JavaScript. As a quick quiz, you should be able to easily read and understand this code:

```
const list = [];
for (let i = 1; i <= 100; i++) {
  if (!(i % 15)) {
    list.push('FizzBuzz');
  } else if (!(i % 5)) {
    list.push('Buzz');
  } else if (!(i % 3)) {
    list.push('Fizz');
  } else {
    list.push(i);
  }
}
```

You may recognize this as a solution to the classic programming puzzle called FizzBuzz, made famous by Jeff Atwood in 2007.[5] Here's another solution—one that makes gratuitous (and unnecessary) use of some of the newer JavaScript features.

```
'use strict';
const list = [...Array(100).keys()]
  .map(n => n + 1)
  .map(n => n % 15 ? n : 'FizzBuzz')
  .map(n => isNaN(n) || n % 5 ? n : 'Buzz')
  .map(n => isNaN(n) || n % 3 ? n : 'Fizz');
```

2. https://blog.risingstack.com/node-js-developer-survey-results-2016/
3. https://facebook.github.io/react/
4. https://angularjs.org/
5. https://blog.codinghorror.com/why-cant-programmers-program/

If you don't recognize the techniques used in this code, that's expected! You'll learn to use several of them, and many others, in this book.

### A Note to Windows Users

The examples in this book assume you're using a Unix-like operating system. We'll make use of standard input and output streams, and pipe data between processes. The shell session examples have been tested with Bash, but other shells may work as well.

If you run Windows, I recommend setting up Cygwin.[6] This will give you the best shot at running the example code successfully, or you could run a Linux virtual machine.

## Code Examples and Conventions

The code examples in this book contain JavaScript, shell sessions, and a few HTML/XML excerpts. For the most part, code listings are provided in full—ready to be run at your leisure.

Samples and snippets are syntax-highlighted according to the rules of the language. Shell commands are prefixed by `$`.

When you write Node.js code, you should always handle errors and exceptions, even if you just rethrow them. You'll learn how to do this throughout the book. However, some of the code examples lack error handling. This is to aid readability and save space, and sometimes to provide opportunities for reader tasks at the end of the chapter. In your code, you should always handle your errors.

## Online Resources

The Pragmatic Bookshelf's page for this book is a great resource.[7] You'll find downloads for all the source code presented in this book, and feedback tools, including a community forum and an errata-submission form.

Thanks for choosing this book to show you Node.js *the right way*.

**Jim R. Wilson**

December 2017

6. http://cygwin.com/
7. http://pragprog.com/book/jwnode2/node-js-8-the-right-way