Rapid Prototyping and Scalable Deployment

Building

Node Applications

with MongoDB and Backbone

O'REILLY®

Mike Wilson

Building Node Applications with MongoDB and Backbone

Build an application from backend to browser with Node.js, and kick open the doors to real-time event programming. With this hands-on book, you'll learn how to create a social network application similar to LinkedIn and Facebook, but with a real-time twist. And you'll build it with just one programming language: JavaScript.

If you're an experienced web developer unfamiliar with JavaScript, the book's first section introduces you to the project's core technologies: Node.js, Backbone.js, and the MongoDB data store. You'll then launch into the project—a highly responsive, highly scalable application—guided by clear explanations and lots of code examples.

- Learn about key modules in Node.js for building real-time apps
- Use the Backbone.js framework to write clean browser code, and maintain better data integration with MongoDB
- Structure project files as a foundation for code that will arrive later
- Create user accounts and learn how to secure the data
- Use Backbone.js templates to build the application's UIs, and integrate access control with Node.js
- Develop a contact list to help users link to and track other accounts
- Use Socket.io to create real-time chat functionality
- Extend your UIs to give users up-to-the-minute information

Mike Wilson is an experienced software architect and web developer who has designed and built everything from government portals and small business sites to MMO server clusters hosting millions of players. He has worked with some of the world's most influential brands, including Disney, Microsoft, and McDonalds.

"This book will not only help you learn Node.js, but Backbone.js and MongoDB as well. Each of these is great all by itself, but this book brings them together to build an incredible, real-time social network."

—Jamie Munro
author of 20 Recipes for Programming PhoneGap *(O'Reilly)*

US $19.99 CAN $20.99
ISBN: 978-1-449-33739-1

Twitter: @oreillymedia
facebook.com/oreilly

O'REILLY®
oreilly.com

Building Node Applications with MongoDB and Backbone

Mike Wilson

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Building Node Applications with MongoDB and Backbone

by Mike Wilson

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Simon St. Laurent and Meghan Blanchette
Production Editor: Kara Ebrahim

Proofreader: Kara Ebrahim
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Rebecca Demarest

December 2012: First Edition

Revision History for the First Edition:

2012-12-07 First release

See *http://oreilly.com/catalog/errata.csp?isbn=9781449337391* for release details.

ISBN: 978-1-449-33739-1

[LSI]

Table of Contents

Preface

When Google released the first version of their V8 JavaScript engine in 2008, it felt like a hushed wave of excitement was rippling through the developer community. For the first time (the promise went), we would be able to program with JavaScript on both the client and the server: one language to rule them all. Web applications were already starting to become more desktop-like and ballooning in complexity, so the idea of reducing the number of language dependencies in favor of an open and transparent technology was seen as a way to allow for even more exciting and boundary-pushing applications.

Ryan Dahl was one of the developers who saw the new opportunity and wasted no time converting the non-blocking socket library he had written to the new V8 engine, resulting in the birth of Node.js. The technology he released has turned that original ripple of excitement into a major paradigm shift at a time when interest in responsive real-time applications is reaching a peak. Node.js is more than just a collection of socket functions; it provides a framework for asynchronous I/O that position it as the foundation of a whole new class of event-driven programming patterns.

The online landscape has changed rapidly in the past few years and doesn't show any signs of slowing down. The explosion of the "social" web has meant one big thing for us: more people are online now than ever before, and the demographic has forever shifted away from technical users. The Internet is for all of us, and the winners in this new space will be those companies that can figure out how to make the online experience warm and human by truly connecting individuals to each other.

Using JavaScript to connect your systems puts you at an advantage because you can quickly move from the front of the web stack dealing with human users to the backend

data storage, and all of the network plumbing in between. You will be able to think of your systems as truly modular; each piece can be plugged in and deployed wherever the resources are best suited to it. You will be able to create applications that grow and breathe with your userbase unlike ever before.

Audience and Assumptions

Readers of this book should have an understanding of how websites and web applications are put together. In an effort to stay focused on the core technology, this book brushes past "why" web applications are built in a certain way in favor of the "how."

Some knowledge of JavaScript would come in handy to fully understand the examples in this book. The examples will be thoroughly explained, but prior knowledge will help readers comprehend the back history for programming decisions made during the writing process.

Many developers approach NoSQL data stores as part of a transition from relational database systems. This book makes no assumptions about the reader's proficiency in database design; I will go through the details of why I chose to make various decisions throughout the database architecting phase. MongoDB is friendly to SQL concepts, which is a major motivation for choosing it as the datastore for this project.

In the final section of the book I will discuss a selection of supporting tools and technologies that step outside of the pure JavaScript environment built in the first two sections. Readers are not expected to have a deep understanding of any of those extra languages (like Scala, Java, PHP, or Bash Scripting), but because deep exploration of these concepts are outside the scope of this book, I encourage using these examples as a launching pad for further research.

Organization

This book is broadly organized into two sections, the first providing an overview of Node.js, MongoDB, and Backbone.js (the core technology discussed in this book), and the second detailing how you can go about building a website styled as a social network using these tools. If you are new to any of these I recommend starting with the Part I section to gain a bit of background before diving into the application in the second section. If you are already familiar with JavaScript you will probably be able to skip the first section and find yourself comfortable enough to get through the examples in the second section.

Here's how the book is organized:

Part I: Introduction

Chapter 1, Introduction and Overview
: This chapter introduces JavaScript and the core concepts that will be explored throughout the book.

Chapter 2, Node.js
: This chapter introduces Node.js and guides you through getting started with your first standalone applications. Here you will become acquainted with the key modules you will later use to build a complete real-time application.

Chapter 3, Backbone.js
: Next you will explore how Backbone.js is making programming in the web browser with JavaScript more like building traditional applications and less like building websites. We'll look into some of the more troubling aspects of maintaining JavaScript-based projects, and introduce templating as a way to separate your visual HTML layout from your functional JavaScript application code.

Chapter 4, MongoDB
: I love MongoDB because it is fast and easy to set up, easy to interface with, and speaks the same language as my Node.js applications. In this chapter we'll look at how to do basic querying and data manipulation as well as some more complex use cases to think about as your MongoDB usage grows.

Part II: Building a Social Network

Chapter 5, Setting Up the Project
: The lack of information about how to structure and put together files in your project is one of the biggest problems facing texts that explain how to build websites. In this chapter we'll set up the Node.js and Backbone project files that will form the website, and lay the foundation for the rest of the code that will be coming.

Chapter 6, Authentication
: Before you can do anything with your application, you need a way to create accounts and sign in. This chapter explains how to get users into your database and how to secure their data once you have it.

Chapter 7, The User Interface
: Now that the barebone structure and login functionality have been built, this chapter will take you through setting up the web page harness that will contain all of the content presented to your users. This is where we will go into detail on using templates with Backbone.js and integrating access control with Node.js.

Chapter 8, Making Friends
: The contact list is the social aspect behind this website. In this chapter you will learn how to add and remove contacts from your list, denormalizing the data into MongoDB as you go. This will be a departure for anyone coming from a relational database environment; it's recommended reading!

Chapter 9, Chat
: This chapter builds upon the contact list created in Chapter 8 by adding real-time chat functionality using Socket.io. Talk to your friends and receive messages back right away without needing to reload your page.

Chapter 10, Activities in Real Time
: Finally, the user interfaces built throughout the book will be revisited in this chapter and extended with Socket.io just like the chat list. This will add life to the site by giving your users up-to-the- minute information about the comings and goings of their contacts, and turn all of the shared message spaces into interactive rooms.

Conventions Used in This Book

The following typographical conventions are used in this book:

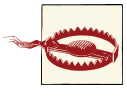
Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, if this book includes code examples, you may use the code in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Building Node Applications with MongoDB and Backbone* by Mike Wilson (O'Reilly). Copyright 2013 Mike Wilson, 978-1-449-33739-1."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online (*www.safaribooksonline.com*) is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/mongodb-backbone*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

PART I

Introducing Node.js, Backbone.js, and MongoDB

CHAPTER 1
Introduction and Overview

The Web, already one of the fastest developing areas in technology, is accelerating. This is both good news and bad news for those of us planning to draw income from writing software. Today, good developers have the rare opportunity to do what they love, grow their horizons, and continually evolve and derive even greater satisfaction from their work, as long as they're willing to put in the hard work necessary to understand a huge back catalog of rapidly-expanding knowledge.

Terrific careers come at a price. As a software developer, you must continually search for the next great tool that will help you achieve more, better, faster. What you work with 10 years from now is going to be a major departure from what you are working with today—in essence, you will be retraining yourself multiple times to keep sharp.

In his 2008 book *Outliers* (Back Bay Books), Malcolm Gladwell presents evidence that it takes 10,000 hours of effort to achieve mastery at a professional level. Even prodigies need to put in their time to achieve success; the difference between an average performer and a superb performer comes down to the amount of practice put in by the individual. Picking up a book like this puts you into the latter category; right now you are putting in the extra time to gain more exposure to the leading edge of this craft. The future is arriving, and you will be among the first positioned to take advantage of it.

Node.js has introduced an army of programmers to event-oriented programming. Regardless of what your technology background is, if you come to Node with an open mind and drop any preconceived notions you might have about JavaScript, you will come away with a greater appreciation of how powerful single-threaded programming can be in a world that has gone crazy for multi-threaded applications. What's more, you will have a greater appreciation of event handling that will help you when you do need to tackle multi-threaded problems in other programming languages.

JavaScript is a unique and sometimes misunderstood programming language that has finally taken its deserved place in the development toolbox. As the toolsets for developing JavaScript applications continue to improve and mature, you can look forward to seeing this language's importance continue to grow in organizations worldwide.

Building a Social Network

The project in this book examines how you would go about constructing a social network in a similar vein to LinkedIn, MySpace, or Facebook, with a real-time networking twist. Using Node.js, Backbone.js, and MongoDB you will learn how to create a highly responsive application that can be adapted to scale to millions of users.

By way of example many of the components described throughout the text will take some shortcuts to use a built-in method provided by Node or MongoDB in order to demonstrate certain functionality that wouldn't be practical in "real" large deployments. When one of those shortcuts is presented, I will point it out with a special note and discuss how to begin moving to a more scalable or modifiable construct. The challenge throughout will be trying to balance the need for clarity with the task of building a real and useful application.

What is a social network? "Social network" is a simple phrase that seems to communicate a lot of meaning—and in behavioral science, it does—but let's look more carefully at the individual words and apply them to the Internet. A "network" is an interconnected group of systems, which could be anything from a series of roads crisscrossing the country to a row of computers in a school lab to a Rolodex filled with professional contacts. The word "social" refers to the interaction of organisms—such as animals or people—and to their existence as an entire group. So a social network in this context means an interconnected, interactive group of people.

The human component is important above all else. When building any kind of software you are remiss to develop toward a particular goal or functionality without first (and constantly) thinking about the person who is expected to use your finished code at the end of the day, whether it is a customer, a professor, or even yourself. Unless you can visualize the end purpose for your work, resist the urge to continue down the programming road for technology's sake.

When we speak of building a social network, of course it's impossible to build a social network as defined here. What you will be creating is the forum, the raw pathways, upon which a social network can take root and grow. Every feature of the system is intended to deliver upon that goal by getting out of the users' way and by providing just enough of a feature set to promote, encourage, and facilitate communication without any extra frills. It's a difficult line to walk, but one that ultimately separates a mediocre product from a great one.

Model-View-Controller (MVC)

This book makes frequent reference to, and use of, the Model-View-Controller (MVC) design pattern for both server-side and frontend programming. While MVC was arguably popularized on the web by the growth of Ruby on Rails, it was first developed for the Smalltalk platform in the 1970s.

MVC as it is practiced today promotes decoupling your system into three components:

Model
: A structure containing the data that is being read or acted upon

View
: An interface through which the user interacts with the model

Controller
: Delegates user actions from the view to the underlying model

Models and controllers are typically paired; in this book, the controller's job will be to act as a contract for what a user is able to do to a model, and to pass information back and forth. While it is possible to have a controller perform actions on more than one model, doing so should be considered poor form: one model, one controller.

Views are a different story; just as in real life, in software there is often multiple ways to perceive the same information. For example, a textual transcription of an audio recording contains the same information as the original, but presents its contents in a way that is more accessible to some users or convenient for others. The Internet is full of great examples of this: many web services display data in both JSON and XML format, two different formats that provide the same information in different ways.

Pure JavaScript

Using Node, Backbone, and MongoDB will allow you to focus on your application logic in a single programming language, ultimately reducing the number of connections between each part of your system. As you will see, this is a compelling way to program because the boundaries between client-facing UI, backend server logic, and database persistence will blur into almost a living system. The picture becomes clearer as real-time networking is gradually added; your data will dance across the application and even across multiple users almost as if everything were happening in concert in a single process.

There are pitfalls to watch out for. Although the connectors are strong and speak the same dialect, under the covers your program code is still going across a wire between web browsers, servers, and databases. Some of the JavaScript paradigms change slightly depending on whether their primary goal is to serve UI (as in the case of Backbone in the web browser), authentication (as in the case of Node on the web server), or

persistence (as in the case MongoDB). You need to be ever vigilant about where your data is going, whether or not you are blocking any of your own processes, and how to listen for and react to incoming and outgoing events. It can be a challenge, but as with any other system with lots of moving parts, there are a ton of interesting lessons to glean from the experimentation.

CHAPTER 2
Node.js

The Internet of today is different from the Internet of 1990 and 2000. In the "old days," the interaction between a user and a website was very much oriented toward consumption. The web server would generate largely static pages and the user would navigate between them. There were of course dynamic elements but the interfaction flow was largely limited to request and reply. Years of research have gone into optimizing that client-server flow—it's safe to say that it's well understood at this point in time.

Around the time Internet Explorer 6 started to appear, a subtle but fundamental shift was beginning to take hold. Internet users were becoming more comfortable and savvy online, computers were becoming far more powerful, and broadband connections were starting to become the norm. Instead of using the Internet primarily for information and transactions, people were spending more time online for socializing and entertainment. The Internet is now a media channel, but unlike the television, radio, and newspapers before it.

Instead of consuming data, web users are now producing it in volumes never imagined. The traditional notion of web servers and browsers as consumers is still present, but understanding it provides only a glimpse into what publishers are able to accomplish. The focus now is on putting people in control of their experience, and leveraging the data they create to change, improve, and enhance that experience in real time. This is a new world where the web server and programmer are no longer the sources of experience; rather, they are the facilitators.

Node.js is one of a new breed of technologies geared toward the Internet-as-experience paradigm.