Getting MEAN

with Mongo, Express, Angular, and Node

SIMON HOLMES

MANNING
SHELTER ISLAND

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact

Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

©2016 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

♾ Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editors:	Susie Pitzen, Susanna Kline, Karen Miller
Technical development editor:	Marius Butuc
Copyeditor:	Jodie Allen
Proofreader:	Alyson Brener
Technical proofreaders:	Steven Jenkins, Deepak Vohra
Typesetter:	Dennis Dalinnik
Cover designer:	Marija Tudor

ISBN: 9781617292033
Printed in the United States of America
1 2 3 4 5 6 7 8 9 10 – EBM – 20 19 18 17 16 15

brief contents

PART 1 SETTING THE BASELINE 1

1 ■ Introducing full-stack development 3
2 ■ Designing a MEAN stack architecture 24

PART 2 BUILDING A NODE WEB APPLICATION 51

3 ■ Creating and setting up a MEAN project 53
4 ■ Building a static site with Node and Express 80
5 ■ Building a data model with MongoDB and Mongoose 120
6 ■ Writing a REST API: Exposing the MongoDB database to the application 160
7 ■ Consuming a REST API: Using an API from inside Express 202

PART 3 ADDING A DYNAMIC FRONT END WITH ANGULAR......... 241

8 ■ Adding Angular components to an Express application 243

9 ■ Building a single-page application with Angular: Foundations 276

10 ■ Building an SPA with Angular: The next level 304

PART 4 MANAGING AUTHENTICATION AND USER SESSIONS 347

11 ■ Authenticating users, managing sessions, and securing APIs 349

contents

preface *xv*
acknowledgments *xvii*
about this book *xix*

PART 1 SETTING THE BASELINE 1

1 ***Introducing full-stack development*** ***3***

1.1 Why learn the full stack? 4

A very brief history of web development *4* ▪ *The trend toward full-stack developers* *6* ▪ *Benefits of full-stack development* *6*
Why the MEAN stack specifically? *7*

1.2 Introducing Node.js: The web server/platform 7

JavaScript: The single language through the stack *8*
Fast, efficient, and scalable *8* ▪ *Using prebuilt packages via npm* *11*

1.3 Introducing Express: The framework 12

Easing your server setup *12* ▪ *Routing URLs to responses* *12*
Views: HTML responses *12* ▪ *Remembering visitors with session support* *13*

1.4 Introducing MongoDB: The database 13

Relational versus document databases 13 ▪ MongoDB documents: JavaScript data store 14 ▪ More than just a document database 14 ▪ What is MongoDB not good for? 15 ▪ Mongoose for data modeling and more 15

1.5 Introducing AngularJS: The front-end framework 16

jQuery versus AngularJS 16 ▪ Two-way data binding: Working with data in a page 16 ▪ Using AngularJS to load new pages 18 Are there any downsides? 18

1.6 Supporting cast 19

Twitter Bootstrap for user interface 19 ▪ Git for source control 20 Hosting with Heroku 20

1.7 Putting it together with a practical example 21

Introducing the example application 21 ▪ How the MEAN stack components work together 22

1.8 Summary 23

2 ***Designing a MEAN stack architecture 24***

2.1 A common MEAN stack architecture 25

2.2 Looking beyond SPAs 26

Hard to crawl 26 ▪ Analytics and browser history 27 Speed of initial load 27 ▪ To SPA or not to SPA? 28

2.3 Designing a flexible MEAN architecture 28

Requirements for a blog engine 29 ▪ A blog engine architecture 30 ▪ Best practice: Build an internal API for a data layer 33

2.4 Planning a real application 34

Planning the application at a high level 35 ▪ Architecting the application 36 ▪ Wrapping everything in an Express project 38 The end product 39

2.5 Breaking the development into stages 40

Rapid prototype development stages 40 ▪ The steps to build Loc8r 41

2.6 Hardware architecture 47

Development hardware 47 ▪ Production hardware 47

2.7 Summary 49

PART 2 BUILDING A NODE WEB APPLICATION 51

3 *Creating and setting up a MEAN project 53*

3.1 A brief look at Express, Node, and npm 55

Defining packages with package.json 55 ▪ Installing Node dependencies with npm 56

3.2 Creating an Express project 58

Installing the pieces 58 ▪ Creating a project folder 58 Configuring an Express installation 59 ▪ Creating an Express project and trying it out 61 ▪ Restarting the application 62

3.3 Modifying Express for MVC 64

A bird's eye view of MVC 64 ▪ Changing the folder structure 65 Using the new views and routes folders 66 ▪ Splitting controllers from routes 67

3.4 Import Bootstrap for quick, responsive layouts 70

Download Bootstrap and add it to the application 70 Using Bootstrap in the application 70

3.5 Make it live on Heroku 74

Getting Heroku set up 74 ▪ Pushing the site live using Git 76

3.6 Summary 79

4 *Building a static site with Node and Express 80*

4.1 Defining the routes in Express 82

Different controller files for different collections 83

4.2 Building basic controllers 84

Setting up controllers 85 ▪ Testing the controllers and routes 86

4.3 Creating some views 87

A look at Bootstrap 88 ▪ Setting up the HTML framework with Jade templates and Bootstrap 89 ▪ Building a template 93

4.4 Adding the rest of the views 98

Details page 98 ▪ Adding Review page 102 The About page 104

4.5 Take the data out of the views and make them smarter 106

How to move data from the view to the controller 107 Dealing with complex, repeating data 109 ▪ Manipulating the data and view with code 113 ▪ Using includes and mixins to *create reusable layout components 113 ▪ The finished homepage 115 ▪ Updating the rest of the views and controllers 117*

4.6 Summary 119

5 *Building a data model with MongoDB and Mongoose 120*

5.1 Connecting the Express application to MongoDB using Mongoose 122

Adding Mongoose to our application 123 ▪ Adding a Mongoose connection to our application 124

5.2 Why model the data? 130

What is Mongoose and how does it work? 131

5.3 Defining simple Mongoose schemas 134

The basics of setting up a schema 135 ▪ Using geographic data in MongoDB and Mongoose 137 ▪ Creating more complex schemas with subdocuments 138 ▪ Final schema 143 Compiling Mongoose schemas into models 145

5.4 Using the MongoDB shell to create a MongoDB database and add data 147

MongoDB shell basics 147 ▪ Creating a MongoDB database 148

5.5 Getting our database live 152

Setting up MongoLab and getting the database URI 152 Pushing up the data 154 ▪ Making the application use the right database 156

5.6 Summary 159

6 *Writing a REST API: Exposing the MongoDB database to the application 160*

6.1 The rules of a REST API 161

Request URLs 162 ▪ Request methods 163 ▪ Responses and status codes 165

6.2 Setting up the API in Express 167

Creating the routes 167 ▪ Creating the controller placeholders 170 ▪ Including the model 171 Testing the API 172

6.3 GET methods: Reading data from MongoDB 172

Finding a single document in MongoDB using Mongoose 173
Finding a single subdocument based on IDs 177
Finding multiple documents with geospatial queries 180

6.4 POST methods: Adding data to MongoDB 187

Creating new documents in MongoDB 188 ▪ Creating new subdocuments in MongoDB 190

6.5 PUT methods: Updating data in MongoDB 193

Using Mongoose to update a document in MongoDB 194
Updating an existing subdocument in MongoDB 196

6.6 DELETE method: Deleting data from MongoDB 197

Deleting documents in MongoDB 198 ▪ Deleting a subdocument from MongoDB 199

6.7 Summary 200

7 Consuming a REST API: Using an API from inside Express 202

7.1 How to call an API from Express 203

Adding the request module to our project 203 ▪ Setting up default options 204 ▪ Using the request module 204

7.2 Using lists of data from an API: The Loc8r homepage 206

Separating concerns: Moving the rendering into a named function 207 ▪ Building the API request 207
Using the API response data 208 ▪ Modifying data before displaying it: Fixing the distances 209 ▪ Catching errors returned by the API 212

7.3 Getting single documents from an API: The Loc8r Details page 216

Setting URLs and routes to access specific MongoDB documents 216 ▪ Separating concerns: Moving the rendering into a named function 218 ▪ Querying the API using a unique ID from a URL parameter 219 ▪ Passing the data from the API to the view 220 ▪ Debugging and fixing the view errors 221
Creating status-specific error pages 223

7.4 Adding data to the database via the API: Add Loc8r reviews 226

Setting up the routing and views 227 ▪ POSTing the review data to the API 231

7.5 Protecting data integrity with data validation 233

Validating at the schema level with Mongoose 234 ▪ Validating at the application level with Node and Express 237 ▪ Validating in the browser with jQuery 239

7.6 Summary 240

PART 3 ADDING A DYNAMIC FRONT END WITH ANGULAR 241

8 *Adding Angular components to an Express application 243*

8.1 Getting Angular up and running 244

Uncovering two-way data binding 245 ▪ Setting up for greatness (and JavaScript code) 248

8.2 Displaying and filtering the homepage list 251

Adding Angular to an Express application 251 ▪ Moving data delivery from Express to Angular 252 ▪ Using Angular filters to format data 255 ▪ Using Angular directives to create HTML snippets 259

8.3 Getting data from an API 263

Using services for data 264 ▪ Making HTTP requests from Angular to an API 265 ▪ Adding HTML geolocation to find places near you 268

8.4 Ensuring forms work as expected 274

8.5 Summary 275

9 *Building a single-page application with Angular: Foundations 276*

9.1 Setting the groundwork for an Angular SPA 277

Getting base files in place 278

9.2 Switching from Express routing to Angular routing 279

Switching off the Express routing 279 ▪ Adding ngRoute (angular-route) to the application 282

9.3 Adding the first views, controllers, and services 284

Creating an Angular view 284 ▪ Adding a controller to a route 286 ▪ Controller best practice: Using the controllerAs syntax 288 ▪ Using services 291 Using filters and directives 294

9.4 Improving browser performance 297

Wrap each file in an IIFE 298 ▪ Manually injecting dependencies to protect against minification 299 ▪ Using UglifyJS to minify and concatenate scripts 300

9.5 Summary 303

10 ***Building an SPA with Angular: The next level 304***

10.1 A full SPA: Removing reliance on the server-side application 305

Creating an isolated HTML host page 305
Making reusable page framework directives 307
Removing the # from URLs 312

10.2 Adding additional pages and dynamically injecting HTML 314

Adding a new route and page to the SPA 315
Creating a filter to transform the line breaks 317
Sending HTML through an Angular binding 319

10.3 More complex views and routing parameters 321

Getting the page framework in place 321 ▪ Using URL parameters in controllers and services 323 ▪ Building the Details page view 326

10.4 Using AngularUI components to create a modal popup 330

Getting AngularUI in place 330 ▪ Adding and using a click handler 332 ▪ Creating a Bootstrap modal with AngularUI 333
Passing data into the modal 335 ▪ Using the form to submit a review 337

10.5 Summary 345

PART 4 MANAGING AUTHENTICATION AND USER SESSIONS 347

11 ***Authenticating users, managing sessions, and securing APIs 349***

11.1 How to approach authentication in the MEAN stack 350

Traditional server-based application approach 350 ▪ Full MEAN stack approach 352

11.2 Creating a user schema for MongoDB 354

One-way password encryption: Hashes and salts 354 ▪ Building the Mongoose schema 354 ▪ Setting encrypted paths using Mongoose methods 355 ▪ Validating a submitted password 357 Generating a JSON Web Token 357

11.3 Creating an authentication API with Passport 360

Installing and configuring Passport 360 ▪ Creating API endpoints to return JSON Web Tokens 363

11.4 Securing relevant API endpoints 368

Adding authentication middleware to Express routes 368 Using the JWT information inside a controller 369

11.5 Creating Angular authentication service 373

Managing a user session in Angular 373 ▪ Allowing users to sign up, sign in, and sign out 374 ▪ Using the JWT data in the Angular service 375

11.6 Creating register and login pages 377

Building the register page 377 ▪ Building the login page 380

11.7 Working with authentication in the Angular app 383

Updating navigation 383 ▪ Adding user data to a review 386

11.8 Summary 389

appendix A Installing the stack 391

appendix B Installing and preparing the supporting cast 395

appendix C Dealing with all of the views 399

appendix D Reintroducing JavaScript available online only

index 405