

# contents

---

*author's introduction* xi  
*acknowledgments* xiii  
*about this book* xv  
*about the cover illustration* xxi

## PART I WARMING UP .....

### **JavaScript is everywhere 3**

- 1.1 Understanding the JavaScript language 4
  - How will JavaScript evolve?* 6 • *Transpilers give us access to tomorrow's JavaScript today* 6
- 1.2 Understanding the browser 7
- 1.3 Using current best practices 8
  - Debugging* 9 • *Testing* 9 • *Performance analysis* 10
- 1.4 Boosting skill transferability 10
- 1.5 Summary 11

### **Building the page at runtime 13**

- 2.1 The lifecycle overview 14

- 2.2 The page-building phase 17
  - Parsing the HTML and building the DOM* 18 ▪ *Executing JavaScript code* 20
- 2.3 Event handling 23
  - Event-handling overview* 24 ▪ *Registering event handlers* 25
  - Handling events* 27
- 2.4 Summary 29
- 2.5 Exercises 29

## PART 2 UNDERSTANDING FUNCTIONS .....31

### 3 **First-class functions for the novice: definitions and arguments** 33

- 3.1 What's with the functional difference? 34
  - Functions as first-class objects* 35 ▪ *Callback functions* 36
- 3.2 Fun with functions as objects 40
  - Storing functions* 40 ▪ *Self-memoizing functions* 42
- 3.3 Defining functions 44
  - Function declarations and function expressions* 45 ▪ *Arrow functions* 50
- 3.4 Arguments and function parameters 52
  - Rest parameters* 54 ▪ *Default parameters* 55
- 3.5 Summary 58
- 3.6 Exercises 59

### 4 **Functions for the journeyman: understanding function invocation** 61

- 4.1 Using implicit function parameters 62
  - The arguments parameter* 62 ▪ *The this parameter: introducing the function context* 67
- 4.2 Invoking functions 67
  - Invocation as a function* 68 ▪ *Invocation as a method* 69
  - Invocation as a constructor* 72 ▪ *Invocation with the apply and call methods* 77
- 4.3 Fixing the problem of function contexts 83
  - Using arrow functions to get around function contexts* 83
  - Using the bind method* 86

4.4 Summary 88

4.5 Exercises 88

## ***Functions for the master: closures and scopes 91***

5.1 Understanding closures 92

5.2 Putting closures to work 95

*Mimicking private variables 95* ▪ *Using closures with callbacks 96*

5.3 Tracking code execution with execution contexts 99

5.4 Keeping track of identifiers with lexical environments 103

*Code nesting 103* ▪ *Code nesting and lexical environments 104*

5.5 Understanding types of JavaScript variables 106

*Variable mutability 107* ▪ *Variable definition keywords and lexical environments 109* ▪ *Registering identifiers within lexical environments 113*

5.6 Exploring how closures work 117

*Revisiting mimicking private variables with closures 117* ▪ *Private variables caveat 121* ▪ *Revisiting the closures and callbacks example 122*

5.7 Summary 124

5.8 Exercises 124

## ***Functions for the future: generators and promises 126***

6.1 Making our async code elegant with generators and promises 127

6.2 Working with generator functions 129

*Controlling the generator through the iterator object 130* ▪ *Using generators 133* ▪ *Communicating with a generator 136*  
*Exploring generators under the hood 139*

6.3 Working with promises 146

*Understanding the problems with simple callbacks 147* ▪ *Diving into promises 149* ▪ *Rejecting promises 152* ▪ *Creating our first real-world promise 154* ▪ *Chaining promises 155*  
*Waiting for a number of promises 156*

6.4 Combining generators and promises 158

*Looking forward—the async function 161*

6.5 Summary 162

6.6 Exercises 163

## PART 3 DIGGING INTO OBJECTS AND FORTIFYING

YOUR CODE ..... 165

**Object orientation with prototypes 167**

- 7.1 Understanding prototypes 168
- 7.2 Object construction and prototypes 171
  - Instance properties* 173 ▪ *Side effects of the dynamic nature of JavaScript* 176 ▪ *Object typing via constructors* 179
- 7.3 Achieving inheritance 181
  - The problem of overriding the constructor property* 184 ▪ *The instanceof operator* 187
- 7.4 Using JavaScript “classes” in ES6 190
  - Using the class keyword* 190 ▪ *Implementing inheritance* 193
- 7.5 Summary 195
- 7.6 Exercises 196

**Controlling access to objects 199**

- 8.1 Controlling access to properties with getters and setters 200
  - Defining getters and setters* 202 ▪ *Using getters and setters to validate property values* 207 ▪ *Using getters and setters to define computed properties* 208
- 8.2 Using proxies to control access 210
  - Using proxies for logging* 214 ▪ *Using proxies for measuring performance* 215 ▪ *Using proxies to autopopulate properties* 217
  - Using proxies to implement negative array indexes* 218
  - Performance costs of proxies* 220
- 8.3 Summary 221
- 8.4 Exercises 222

**Dealing with collections 224**

- 9.1 Arrays 225
  - Creating arrays* 225 ▪ *Adding and removing items at either end of an array* 227 ▪ *Adding and removing items at any array location* 230 ▪ *Common operations on arrays* 232 ▪ *Reusing built-in array functions* 242
- 9.2 Maps 244
  - Don't use objects as maps* 245 ▪ *Creating our first map* 247
  - Iterating over maps* 250

- 9.3 Sets 251
  - Creating our first set* 252 ▪ *Union of sets* 253 ▪ *Intersection of sets* 255 ▪ *Difference of sets* 255
- 9.4 Summary 256
- 9.5 Exercises 256

## **Wrangling regular expressions 259**

- 10.1 Why regular expressions rock 260
- 10.2 A regular expression refresher 261
  - Regular expressions explained* 261 ▪ *Terms and operators* 263
- 10.3 Compiling regular expressions 267
- 10.4 Capturing matching segments 269
  - Performing simple captures* 269 ▪ *Matching using global expressions* 271 ▪ *Referencing captures* 272 ▪ *Noncapturing groups* 273
- 10.5 Replacing using functions 274
- 10.6 Solving common problems with regular expressions 276
  - Matching newlines* 277 ▪ *Matching Unicode* 277 ▪ *Matching escaped characters* 278
- 10.7 Summary 279
- 10.8 Exercises 280

## **Code modularization techniques 282**

- 11.1 Modularizing code in pre-ES6 JavaScript 283
  - Using objects, closures, and immediate functions to specify modules* 284
  - Modularizing JavaScript applications with AMD and CommonJS* 290
- 11.2 ES6 modules 294
  - Exporting and importing functionality* 294
- 11.3 Summary 300
- 11.4 Exercises 301

## **PART 4 BROWSER RECONNAISSANCE ..... 303**

### **Working the DOM 305**

- 12.1 Injecting HTML into the DOM 306
  - Converting HTML to DOM* 307 ▪ *Inserting elements into the document* 311

- 12.2 Using DOM attributes and properties 313
- 12.3 Styling attribute headaches 315
  - Where are my styles?* 315 ▪ *Style property naming* 318
  - Fetching computed styles* 319 ▪ *Converting pixel values* 322
  - Measuring heights and widths* 323
- 12.4 Minimizing layout thrashing 327
- 12.5 Summary 330
- 12.6 Exercises 330

## ***Surviving events* 332**

- 13.1 Diving into the event loop 333
  - An example with only macrotasks* 336 ▪ *An example with both macro- and microtasks* 339
- 13.2 Taming timers: time-outs and intervals 344
  - Timer execution within the event loop* 345 ▪ *Dealing with computationally expensive processing* 350
- 13.3 Working with events 353
  - Propagating events through the DOM* 354 ▪ *Custom events* 360
- 13.4 Summary 364
- 13.5 Exercises 364

## ***Developing cross-browser strategies* 367**

- 14.1 Cross-browser considerations 368
- 14.2 The five major development concerns 370
  - Browser bugs and differences* 371 ▪ *Browser bug fixes* 371
  - External code and markup* 373 ▪ *Regressions* 376
- 14.3 Implementation strategies 378
  - Safe cross-browser fixes* 378 ▪ *Feature detection and polyfills* 379
  - Untestable browser issues* 381
- 14.4 Reducing assumptions 383
- 14.5 Summary 384
- 14.6 Exercises 385

- appendix A Additional ES6 features* 387
- appendix B Arming with testing and debugging* 392
- appendix C Exercise answers* 411
- index* 433