

# CONTENTS IN DETAIL

<b>PREFACE</b>	<b>xxi</b>
----------------	------------

<b>ACKNOWLEDGMENTS</b>	<b>xxiii</b>
------------------------	--------------

<b>INTRODUCTION</b>	<b>xxv</b>
---------------------	------------

Who Should Read This Book? . . . . .	xxvi
What's the Book's Approach? . . . . .	xxvi
What's in the Book? . . . . .	xxvi

## **PART I: THE BASICS** **1**

### **1** **USING JAVASCRIPT** **3**

Modern JavaScript Features . . . . .	4
Arrow Functions . . . . .	4
Classes . . . . .	5
The Spread Operator . . . . .	6
The Destructuring Statement . . . . .	7
Modules . . . . .	8
Closures and Immediately Invoked Function Expressions . . . . .	11
JavaScript Development Tools . . . . .	13
Visual Studio Code . . . . .	13
Fira Code Font . . . . .	15
Prettier Formatting . . . . .	16
JSDoc Documentation . . . . .	16
ESLint . . . . .	18
Flow and TypeScript . . . . .	18
Online Feature Availability Resources . . . . .	19
Summary . . . . .	21

### **2** **FUNCTIONAL PROGRAMMING IN JAVASCRIPT** **23**

Why Use Functional Programming? . . . . .	24
JavaScript as a Functional Language . . . . .	25
Functions as First-Class Objects . . . . .	25
Declarative-Style Programming . . . . .	26
Higher-Order Functions . . . . .	30
Side Effects . . . . .	32
Impure Functions . . . . .	33
Summary . . . . .	35
Questions . . . . .	35

<b>3</b>	<b>ABSTRACT DATA TYPES</b>	<b>37</b>
The Theory . . . . .		38
Data Types . . . . .		38
Abstraction . . . . .		39
Operations and Mutations . . . . .		39
Implementing an ADT . . . . .		40
Implementing ADTs Using Classes . . . . .		41
Implementing ADTs Using Functions (Mutable Version) . . . . .		43
Implementing ADTs Using Functions (Immutable Version) . . . . .		45
Summary . . . . .		46
Questions . . . . .		46

<b>4</b>	<b>ANALYZING ALGORITHMS</b>	<b>49</b>
Performance . . . . .		50
Complexity . . . . .		50
Notations for Complexity . . . . .		51
Complexity Classes . . . . .		52
Performance Measurements . . . . .		54
Analysis of Algorithms in Practice . . . . .		55
Time and Space Complexity Trade-offs . . . . .		57
Summary . . . . .		58
Questions . . . . .		58

## **PART II: ALGORITHMS** **61**

<b>5</b>	<b>DESIGNING ALGORITHMS</b>	<b>63</b>
Recursion . . . . .		64
The Divide-and-Conquer Strategy . . . . .		65
The Backtracking Technique . . . . .		69
Dynamic Programming . . . . .		72
Calculating Fibonacci Series with Top-Down DP . . . . .		72
Line Breaking with Top-Down DP . . . . .		74
Calculating Fibonacci Series with Bottom-Up DP . . . . .		79
Summing Ranges Recursively with Bottom-Up DP . . . . .		80
Summing Ranges by Precomputing with Bottom-Up DP . . . . .		81
Brute-Force Search . . . . .		82
Detecting Tautologies . . . . .		82
Solving Cryptarithmic Puzzles . . . . .		83
Greedy Algorithms . . . . .		87
How to Make Change . . . . .		87
The Traveling Salesman Problem . . . . .		87
Minimum Spanning Tree . . . . .		88
Summary . . . . .		88
Questions . . . . .		88

## 6

### **SORTING**

**91**

The Sorting Problem . . . . .	92
Internal vs. External Sorting . . . . .	92
Adaptive Sorting . . . . .	92
In-Place and Out-of-Place Sorting . . . . .	93
Online and Offline Sorting . . . . .	93
Sorting Stability . . . . .	93
JavaScript's Own Sort Method . . . . .	95
Sort Performance . . . . .	96
Sorting with Comparisons . . . . .	97
Bubbling Up and Down . . . . .	97
Sorting Strategies for Playing Cards . . . . .	100
Making Bigger Jumps with Comb and Shell Sort . . . . .	103
Going for Speed with Quicksort . . . . .	105
Merging for Performance with Merge Sort . . . . .	110
Sorting Without Comparisons . . . . .	112
Bitmap Sort . . . . .	112
Counting Sort . . . . .	114
Radix Sort . . . . .	115
Inefficient Sorting Algorithms . . . . .	116
Stooge Sort . . . . .	116
Slow Sort . . . . .	116
Permutation Sort . . . . .	117
Bogosort . . . . .	117
Sleep Sort . . . . .	117
Summary . . . . .	117
Questions . . . . .	118

## 7

### **SELECTING**

**121**

Selection Without Comparisons . . . . .	122
Bitmap Selection . . . . .	122
Counting Selection . . . . .	123
Selecting with Comparisons . . . . .	124
The Quickselect Family . . . . .	125
Quickselect . . . . .	126
Median of Medians . . . . .	127
Repeated Step . . . . .	130
Finding the Median with Lazy Select . . . . .	132
Summary . . . . .	134
Questions . . . . .	134

## 8

### **SHUFFLING AND SAMPLING**

**137**

Choosing Numbers Randomly . . . . .	138
Shuffling . . . . .	139
Shuffling by Sorting . . . . .	139
Shuffling by Coin Tossing . . . . .	140
Shuffling in Linear Time . . . . .	142

Sampling . . . . .	146
Sampling with Repetition . . . . .	146
Sampling Without Repetition . . . . .	147
Summary . . . . .	154
Questions . . . . .	155

**9**  
**SEARCHING** **159**

Search Definition . . . . .	159
Searching Unsorted Arrays . . . . .	160
JavaScript's Methods . . . . .	160
Linear Search . . . . .	160
Linear Search with Sentinels . . . . .	161
Searching Ordered Arrays . . . . .	163
Jump Search . . . . .	163
Binary Search . . . . .	166
Exponential Search . . . . .	168
Interpolation Search . . . . .	169
Summary . . . . .	171
Questions . . . . .	172

**PART III: DATA STRUCTURES** **175**

**10**  
**LISTS** **177**

Basic Lists . . . . .	178
Implementing Lists with Arrays . . . . .	179
Implementing Lists with Dynamic Memory . . . . .	180
Varieties of Lists . . . . .	184
Stacks . . . . .	184
Queues . . . . .	188
Dequeues . . . . .	191
Circular Lists . . . . .	195
Summary . . . . .	200
Questions . . . . .	200

**11**  
**BAGS, SETS, AND MAPS** **203**

Introducing Bags, Sets, and Maps . . . . .	203
JavaScript's Solutions for Sets . . . . .	205
Objects as Sets . . . . .	205
Set Objects . . . . .	206
Bitmaps . . . . .	206
Using Lists . . . . .	207
Ordered Lists . . . . .	207
Skip Lists . . . . .	210
Self-Organizing Lists . . . . .	215

Hashing	218
Hashing with Chaining	219
Hashing with Open Addressing	221
Double Hashing	226
Double Hashing with Prime Lengths	229
Summary	232
Questions	232

## 12

### **BINARY TREES**

**235**

What Are Trees?	236
General Trees	237
Binary Trees	237
Binary Search Trees	239
Assured Balanced Binary Search Trees	249
AVL Trees	249
Weight-Bounded Balanced Trees	255
Probabilistic Balance Binary Search Trees	261
Randomized Binary Search Trees	262
Splay Trees	270
Summary	278
Questions	279

## 13

### **TREES AND FORESTS**

**283**

Defining Trees and Forests	283
Representing Trees with Arrays	284
Representing Trees with Binary Trees	287
Representing Forests	288
Traversing Trees	288
B-trees	291
Defining B-trees	292
Finding a Key in a B-tree	293
Traversing a B-tree	294
Adding a Key to a B-tree	295
Removing a Key from a B-tree	298
Considering Performance for B-trees	303
Red-Black Trees	304
Representing Red-Black Trees	305
Adding a Key to a Red-Black Tree	306
Restoring a Red-Black Tree Structure	307
Removing a Key from a Red-Black Tree	309
Considering Performance for Red-Black Trees	313
Summary	314
Questions	314

## 14 HEAPS

317

Binary Heaps . . . . .	318
The Structure Property . . . . .	318
The Heap Property . . . . .	319
Heap Implementation. . . . .	320
Priority Queues and Heaps . . . . .	326
Heapsort. . . . .	327
Williams' Original Heapsort. . . . .	327
Heapsort Analysis. . . . .	329
Floyd's Heap-Building Enhancement . . . . .	329
Treaps . . . . .	332
Creating and Searching a Treap. . . . .	332
Adding a Key to a Treap . . . . .	333
Removing a Key from a Treap. . . . .	336
Considering the Performance of Treaps . . . . .	339
Ternary and D-ary Heaps . . . . .	340
Summary . . . . .	341
Questions . . . . .	341

## 15 EXTENDED HEAPS

345

Meldable and Addressable Priority Queues. . . . .	346
Skew Heaps . . . . .	347
Representing a Skew Heap . . . . .	348
Merging Two Skew Heaps . . . . .	348
Adding a Key to a Skew Heap . . . . .	350
Removing the Top Key from a Skew Heap . . . . .	350
Considering Performance for Skew Heaps . . . . .	350
Binomial Heaps . . . . .	351
Binomial Trees . . . . .	351
Defining Binomial Heaps . . . . .	353
Adding a Value to a Binomial Heap . . . . .	354
Merging Two Binomial Heaps. . . . .	356
Removing a Value from a Binomial Heap . . . . .	358
Changing a Value in a Binomial Heap. . . . .	360
Considering Performance for Binomial Heaps. . . . .	362
Lazy Binomial Heaps . . . . .	363
Defining Lazy Binomial Heaps . . . . .	363
Adding a Value to a Lazy Binomial Heap. . . . .	364
Removing a Value from a Lazy Binomial Heap . . . . .	365
Changing a Value in a Lazy Binomial Heap . . . . .	366
Considering Performance for Lazy Binomial Heaps . . . . .	367
Fibonacci Heaps . . . . .	367
Representing a Fibonacci Heap . . . . .	368
Merging Two Fibonacci Trees . . . . .	370
Adding a Value to a Fibonacci Heap. . . . .	371
Removing a Value from a Fibonacci Heap . . . . .	371
Changing a Value in a Fibonacci Heap . . . . .	372
Considering Performance for Fibonacci Heaps . . . . .	375

Pairing Heaps . . . . .	376
Defining a Pairing Heap . . . . .	377
Melding Two Pairing Heaps . . . . .	377
Adding a Value to a Pairing Heap . . . . .	378
Removing the Top Value from a Pairing Heap . . . . .	378
Changing a Value in a Pairing Heap . . . . .	382
Considering Performance for Pairing Heaps . . . . .	384
Summary . . . . .	385
Questions . . . . .	385

## 16

### **DIGITAL SEARCH TREES 387**

The Classic Version of Tries . . . . .	388
Storing Extra Data in a Trie . . . . .	390
Searching a Trie . . . . .	390
Adding a Key to a Trie . . . . .	394
Removing a Key from a Trie . . . . .	397
Considering Performance for Tries . . . . .	401
An Enhanced Version of Tries . . . . .	401
Defining an Object-Based Trie . . . . .	402
Searching an Object-Based Trie . . . . .	402
Adding a Key to an Object-Based Trie . . . . .	404
Removing a Key from an Object-Based Trie . . . . .	405
Considering Performance for Object-Based Tries . . . . .	406
Radix Trees . . . . .	406
Defining a Radix Tree . . . . .	407
Searching a Radix Tree . . . . .	407
Adding a Key to a Radix Tree . . . . .	410
Removing a Key from a Radix Tree . . . . .	412
Considering Performance for Radix Trees . . . . .	414
Ternary Search Tries . . . . .	414
Defining Ternary Tries . . . . .	415
Storing Extra Data in a Ternary Trie . . . . .	416
Searching a Ternary Trie . . . . .	416
Adding a Key to a Ternary Trie . . . . .	417
Removing a Key from a Ternary Trie . . . . .	419
Considering Performance for Ternary Tries . . . . .	423
Summary . . . . .	424
Questions . . . . .	424

## 17

### **GRAPHS 425**

What Are Graphs? . . . . .	425
Representing Graphs . . . . .	428
Adjacency Matrix Representation for Graphs . . . . .	428
Adjacency List Representation for Graphs . . . . .	429
Adjacency Set Representation for Graphs . . . . .	430

Finding the Shortest Paths . . . . .	430
Floyd-Warshall's "All Paths" Algorithm . . . . .	430
Bellman-Ford Algorithm . . . . .	434
Dijkstra's Algorithm . . . . .	438
Sorting a Graph . . . . .	444
Kahn's Algorithm . . . . .	445
Tarjan's Algorithm . . . . .	448
Detecting Cycles . . . . .	452
Detecting Connectivity . . . . .	453
Detecting Connectivity with Sets . . . . .	454
Detecting Connectivity with Searches . . . . .	456
Finding a Minimum Spanning Tree . . . . .	458
Prim's Algorithm . . . . .	459
Kruskal's Algorithm . . . . .	462
Summary . . . . .	466
Questions . . . . .	466

**18**  
**IMMUTABILITY AND FUNCTIONAL DATA STRUCTURES** **469**

Functional Data Structures . . . . .	470
Arrays (and Hash Tables) . . . . .	470
Functional Lists . . . . .	470
Functional Trees . . . . .	478
Summary . . . . .	485
Questions . . . . .	485

**ANSWER KEY** **487**

**BIBLIOGRAPHY** **545**

**INDEX** **549**