# Title

Practical Data Structures and Algorithms Using JavaScript

# Rationale, Scope, and Audience

Over the past few years, JavaScript is being used more and more as a server-side computer programming language due to platforms such as Node.js and SpiderMonkey. Now that JavaScript programming is moving out of the browser, programmers will find a need to use many of the tools that more conventional languages such as C++ and Java provide. Among these tools are classic data structures such as linked lists, stacks, queues, and graphs, as well as classic algorithms for sorting and searching data. This book discusses how to implement these data structures and algorithms for server-side JavaScript programming. Each chapter will include at least one real-world application of the data structure or data structures discussed in the chapter, along with exercises that further explore the practical application of the various data structures and algorithms.

JavaScript programmers will find this book useful because it discusses how to implement data structures and algorithms within the constraints that JavaScript places on the programmer, such as arrays that are really objects, overly global variables, and a prototype-based object system. JavaScript has an unfair reputation as a "bad" programming language, but this book demonstrates how you can use JavaScript to develop efficient and effective data structures and algorithms using the language's "good parts."

As the title suggests, this book is practical guide to implementing data structures and algorithms. As such, there is not much mention of the theory behind the data structures and algorithms discussed in the book, but the reader is pointed to several textbooks and other references for more information on the theoretical underpinnings of the topics covered. The book also points the reader to several of the O'Reilly books on JavaScript, including books by Crockford and Flanagan.

# Other Books

The current market of JavaScript books include everything from the old "how to use JavaScript in the browser" texts, to more sophisticated books such as Crockford's short text on how to best program in JavaScript and, of course, Flanagan's JavaScript book, as well as newer texts on learning JavaScript (Eloquent JavaScript), and more advanced texts such as Resig's Ninja book. While some of these books might discuss a few of the algorithms and data structures covered in this book, no other current JavaScript book on the market provides as comprehensive a guide to implementing data structures and algorithms using JavaScript.

# Table of Contents

The content for this book will loosely follow the content of my other Data Structure and Algorithms books for Visual Basic.NET and C#, though the material will be modified for JavaScript. In essence, this book is a practical guide to the classic data structures and algorithms used in computer programming. By practical, I mean that there is more emphasis on how to implement the various data structures and algorithms, as well as how to use them to solve problems, and less emphasis on the theoretical underpinnings of the data structures and algorithms.

Below is the table of contents with a paragraph about each chapter:

Chapter 1, JavaScript Programming Overview – This chapter introduces the reader to the Node.js and SpiderMonkey JavaScript programming environments and how to work with objects in JavaScript. JavaScript uses a prototype-based object system rather than an inheritance-based object system, so this chapter demonstrates how to create and work with JavaScript objects.

Chapter 2, Arrays – This chapter discusses how to use arrays in JavaScript. Several examples are provided to highlight the difference between JavaScript arrays and the standard arrays found in other programming languages.

Chapter 3, Lists – Basic list organization and methods

Chapter 4, Basic Sorting Algorithms – This chapter explores how to implement basic sorting algorithms such as insertion sort, bubble sort, and selection sort. The chapter includes a discussion on the practical efficiency of each algorithm.

Chapter 5, Basic Search Algorithms – This chapter introduces the reader to several basic search algorithms – linear search, binary search, and search using self-organizing data. The practical efficiency of each algorithm is discussed.

Chapter 6, Stacks and Queues – The implementation of stacks and queues are explored in this chapter, as well as several examples that show how to use stacks and queues to solve programming problems.

Chapter 7, Pattern Matching and Text Processing – This chapter discusses how to use regular expressions to perform text processing.

Chapter 8, Dictionaries – This chapter discusses how to build associative data structures using JavaScript objects. A simple dictionary is built first followed by a sorted dictionary.

Chapter 9, Hashing and Hash Tables – This chapter explores how to implement hash tables using a hash function. Different hash functions are explored along with strategies for avoiding collisions.

Chapter 10, Linked Lists – The implementation of linked lists is discussed in this chapter. The chapter covers singly-linked lists, doubly-linked lists, and circularly-linked lists.

Chapter 11, Binary Trees and Binary Search Trees – This chapter discusses how to implement binary trees in general and binary search trees (BST) in particular. Different BST traversal methods are explored, as well as how to insert and delete leaves from BSTs.

Chapter 12, Sets – Creating and using sets is discussed in this chapter. The chapter explores the implementation of two different types of sets – sets that store primarily non-numeric data and sets that store numeric data.

Chapter 13, Graphs and Graph Algorithms – This chapter explore the creation and use of graphs to solve problems in areas such as networking and transportation. A Graph object is created and is used as the basis for several algorithms, such as topological sorting, depth-first search , and breadth-first search.

Chapter 14, Advanced Sorting Algorithms – This chapter covers the more advanced algorithms for sorting data, including ShellSort, MergeSort, HeapSort, and Quicksort.

Chapter 15, Advanced Data Structures and Algorithms for Searching – This chapter covers AVL trees, red-black trees, and skip lists.

Chapter 16, Advanced Algorithms - This chapter explores two advanced topics in algorithms – dynamic programming and greedy algorithms. Dynamic programming is used to implement a function to compute the Fibonacci series, searching for longest common substrings, and for finding a solution to the knapsack problem. Greedy algorithms are used to find a solution to a coin-changing problem, performing data compression using Huffman coding, and to also find a solution to the knapsack problem.

# Length of Book and Completion Date

This book will be approximately 400 pages in length and the author expects to complete the manuscript in three to four months. Several chapters have already been completed in draft form.

# Tools

This book will be written in Microsoft Word, using Notepad for a text editor and the SpiderMonkey JavaScript shell to interpret JavaScript programs.

# About the Author

I am currently employed as an instructor of Computer Information Systems at Pulaski Technical College in North Little Rock, AR. I am also an adjunct instructor of Information Science at the University of Arkansas at Little Rock. Before moving to academia, I was a programmer/analyst for Arkansas Children's Hospital, where I worked in statistical computing and data analysis.

I am currently using the Mozilla JavaScript shell and Node.js, along with MongoDB, to create a custom learning management system (LMS) to use with my programming students. I am also using JavaScript to develop online tutoring systems for the C++ and Java programming languages, starting with a tutorial-based, error messaging system for C++.