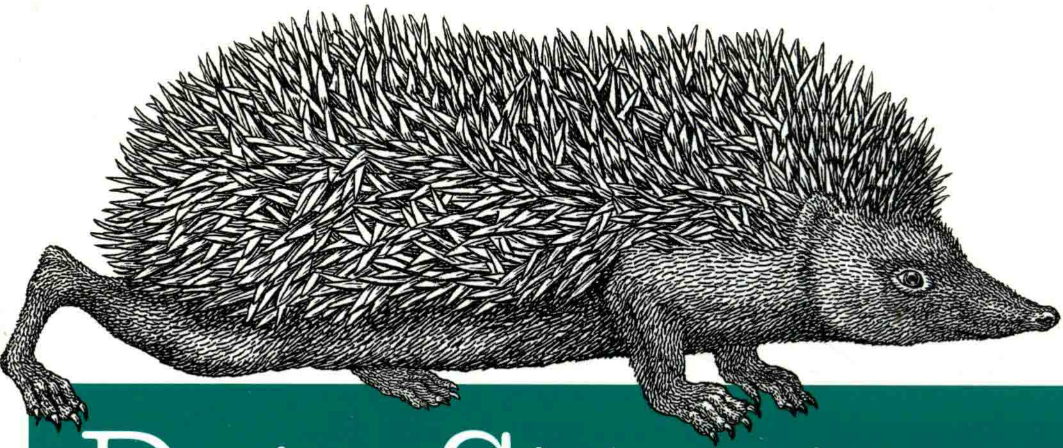
O'REILLY®



Data Structures & Algorithms with JavaScript

JavaScript中的数据结构与算法（影印版）

東南大學出版社

Michael McMillan 著

JavaScript中的数据结构与算法（影印版）

Data Structures and Algorithms with JavaScript

Michael McMillan 著

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

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

JavaScript 中的数据结构与算法:英文/(美)麦克米兰(Michael, M.)著. —影印本. —南京:东南大学出版社,2015.2

书名原文:Data Structures and Algorithms with JavaScript

ISBN 978-7-5641-5345-8

Ⅰ.①J… Ⅱ.①麦… Ⅲ.①JAVA 语言—程序设计—英文 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2014)第 273736 号

江苏省版权局著作权合同登记
图字:10-2014-153 号

JavaScripit 中的数据结构与算法(影印版)

出版发行:东南大学出版社
地　　址:南京四牌楼 2 号　　邮编:210096
出 版 人:江建中
网　　址:http://www.seupress.com
电子邮件:press@seupress.com
印　　刷:常州市武进第三印刷有限公司
开　　本:787 毫米×980 毫米　16 开本
印　　张:15.5
字　　数:304 千字
版　　次:2015 年 2 月第 1 版
印　　次:2015 年 2 月第 1 次印刷
书　　号:ISBN 978-7-5641-5345-8
定　　价:48.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

Preface

Over the past few years, JavaScript has been used more and more as a server-side computer programming language owing to platforms such as Node.js and SpiderMonkey. Now that JavaScript programming is moving out of the browser, programmers will find they need to use many of the tools provided by more conventional languages, such as C++ and Java. 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.

JavaScript programmers will find this book useful because it discusses how to implement data structures and algorithms within the constraints that JavaScript places them, 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."

Why Study Data Structures and Algorithms

I am assuming that many of you reading this book do not have a formal education in computer science. If you do, then you already know why studying data structures and algorithms is important. If you do not have a degree in computer science or haven't studied these topics formally, you should read this section.

The computer scientist Nicklaus Wirth wrote a computer programming textbook titled *Algorithms + Data Structures = Programs* (Prentice-Hall). That title is the essence of computer programming. Any computer program that goes beyond the trivial "Hello, world!" will usually require some type of structure to manage the data the program is written to manipulate, along with one or more algorithms for translating the data from its input form to its output form.

For many programmers who didn't study computer science in school, the only data structure they are familiar with is the array. Arrays are great for some problems, but for many complex problems, they are simply not sophisticated enough. Most experienced programmers will admit that for many programming problems, once they come up with the proper data structure, the algorithms needed to solve the problem are easier to design and implement.

An example of a data structure that leads to efficient algorithms is the binary search tree (BST). A binary search tree is designed so that it is easy to find the minimum and maximum values of a set of data, yielding an algorithm that is more efficient than the best search algorithms available. Programmers unfamiliar with BSTs will instead probably use a simpler data structure that ends up being less efficient.

Studying algorithms is important because there is always more than one algorithm that can be used to solve a problem, and knowing which ones are the most efficient is important for the productive programmer. For example, there are at least six or seven ways to sort a list of data, but knowing that the Quicksort algorithm is more efficient than the selection sort algorithm will lead to a much more efficient sorting process. Or that it's fairly easy to implement a sequential or linear search algorithm for a list of data, but knowing that the binary sort algorithm can sometimes be twice as efficient as the sequential search will lead to a better program.

The comprehensive study of data structures and algorithms teaches you not only which data structures and which algorithms are the most efficient, but you also learn how to decide which data structures and which algorithms are the most appropriate for the problem at hand. There will often be trade-offs involved when writing a program, especially in the JavaScript environment, and knowing the ins and outs of the various data structures and algorithms covered in this book will help you make the proper decision for any particular programming problem you are trying to solve.

What You Need for This Book

The programming environment we use in this book is the JavaScript shell based on the SpiderMonkey JavaScript engine. Chapter 1 provides instructions on downloading the shell for your environment. Other shells will work as well, such as the Node.js JavaScript shell, though you will have to make some translations for the programs in the book to work in Node. Other than the shell, the only thing you need is a text editor for writing your JavaScript programs.

Organization of the Book

- Chapter 1 presents an overview of the JavaScript language, or at least the features of the JavaScript language used in this book. This chapter also demonstrates through use the programming style used throughout the other chapters.
- Chapter 2 discusses the most common data structure in computer programming: the array, which is native to JavaScript.
- Chapter 3 introduces the first implemented data structure: the list.
- Chapter 4 covers the stack data structure. Stacks are used throughout computer science in both compiler and operating system implementations.
- Chapter 5 discusses queue data structures. Queues are an abstraction of the lines you stand in at a bank or the grocery store. Queues are used extensively in simulation software where data has to be lined up before it is processed.
- Chapter 6 covers Linked lists. A linked list is a modification of the list data structure, where each element is a separate object linked to the objects on either side of it. Linked lists are efficient when you need to perform multiple insertions and deletions in your program.
- Chapter 7 demonstrates how to build and use dictionaries, which are data structures that store data as key-value pairs.
- One way to implement a dictionary is to use a hash table, and Chapter 8 discusses how to build hash tables and the hash algorithms that are used to store data in the table.
- Chapter 9 covers the set data structure. Sets are often not covered in data structure books, but they can be useful for storing data that is not supposed to have duplicates in the data set.
- Binary trees and binary search trees are the subject of Chapter 10. As mentioned earlier, binary search trees are useful for storing data that needs to be stored originally in sorted form.
- Chapter 11 covers graphs and graph algorithms. Graphs are used to represent data such as the nodes of a computer network or the cities on a map.
- Chapter 12 moves from data structures to algorithms and discusses various algorithms for sorting data, including both simple sorting algorithms that are easy to implement but are not efficient for large data sets, and more complex algorithms that are appropriate for larger data sets.
- Chapter 13 also covers algorithms, this time searching algorithms such as sequential search and binary search.
- The last chapter of the book, Chapter 14, discusses a couple more advanced algorithms for working with data—dynamic programming and greedy algorithms.

These algorithms are useful for solving hard problems where a more traditional algorithm is either too slow or too hard to implement. We examine some classic problems for both dynamic programming and greedy algorithms in the chapter.

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.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/oreillymedia/data_structures_and_algorithms_using_javascript*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it 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: "*Data Structures and Algorithms Using JavaScript* by Michael McMillian (O'Reilly). Copyright 2014 Michael McMillan, 978-1-449-36493-9."

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 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://oreil.ly/data_structures_algorithms_JS*.

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*

Acknowledgments

There are always lots of people to thank when you've finished writing a book. I'd like to thank my acquisition editor, Simon St. Laurent, for believing in this book and getting me started writing it. Meghan Blanchette worked hard to keep me on schedule, and if I went off schedule, it definitely wasn't her fault. Brian MacDonald worked extremely hard to make this book as understandable as possible, and he helped make several parts of the text much clearer than I had written them originally. I also want to thank my technical reviewers for reading all the text as well as the code, and for pointing out places where both my prose and my code needed to be clearer. My colleague and illustrator, Cynthia Fehrenbach, did an outstanding job translating my chicken scratchings into crisp, clear illustrations, and she deserves extra praise for her willingness to redraw several illustrations at the very last minute. Finally, I'd like to thank all the people at Mozilla for designing an excellent JavaScript engine and shell and writing some excellent documentation for using both the language and the shell.

Table of Contents

CHAPTER 1

The JavaScript Programming Environment and Model

This chapter describes the JavaScript programming environment and the programming constructs we'll use in this book to define the various data structures and algorithms examined.

The JavaScript Environment

JavaScript has historically been a programming language that ran only inside a web browser. However, in the past few years, there has been the development of JavaScript programming environments that can be run from the desktop, or similarly, from a server. In this book we use one such environment: the JavaScript shell that is part of Mozilla's comprehensive JavaScript environment known as SpiderMonkey.

To download the JavaScript shell, navigate to the Nightly Build web page (*http://mzl.la/MKOuFY*). Scroll to the bottom of the page and pick the download that matches your computer system.

Once you've downloaded the program, you have two choices for using the shell. You can use it either in interactive mode or to interpret JavaScript programs stored in a file. To use the shell in interactive mode, type the command `js` at a command prompt. The shell prompt, `js>`, will appear and you are ready to start entering JavaScript expressions and statements.

The following is a typical interaction with the shell:

```
js> 1
1
js> 1+2
3
js> var num = 1;
js> num*124
124
js> for (var i = 1; i < 6; ++i) {
    print(i);
}
1
2
3
4
5
js>
```

You can enter arithmetic expressions and the shell will immediately evaluate them. You can write any legal JavaScript statement and the shell will immediately evaluate it as well. The interactive shell is great for exploring JavaScript statements to discover how they work. To leave the shell when you are finished, type the command `quit()`.

The other way to use the shell is to have it interpret complete JavaScript programs. This is how we will use the shell throughout the rest of the book.

To use the shell to intepret programs, you first have to create a file that contains a JavaScript program. You can use any text editor, making sure you save the file as plain text. The only requirement is that the file must have a *.js* extension. The shell has to see this extension to know the file is a JavaScript program.

Once you have your file saved, you interpret it by typing the `js` command followed by the full filename of your program. For example, if you saved the `for` loop code fragment that's shown earlier in a file named *loop.js*, you would enter the following:

```
c:\js>js loop.js
```

which would produce the following output:

```
1
2
3
4
5
```

After the program is executed, control is returned to the command prompt.

JavaScript Programming Practices

In this section we discuss how we use JavaScript. We realize that programmers have different styles and practices when it comes to writing programs, and we want to describe ours here at the beginning of the book so that you'll understand the more complex code we present in the rest of the book. This isn't a tutorial on using JavaScript but is just a guide to how we use the fundamental constructs of the language.

Declaring and Initializing Variables

JavaScript variables declared outside of a function are global by default and, strictly speaking, don't have to be declared before using. When a JavaScript variable is initialized without first being declared, using the `var` keyword, it becomes a global variable. In this book, however, we follow the convention used with compiled languages such as C++ and Java by declaring all variables before their first use. The added benefit to doing this is that variables declared within function are created as local variables. We will talk more about variable scope later in this chapter.

> You can use *strict mode* to ensure variables are declared before use. Insert the following line *exactly* before any other statement:
>
> ```
> 'use strict';
> ```
>
> Or
>
> ```
> "use strict";
> ```

To declare a variable in JavaScript, use the keyword `var` followed by a variable name, and optionally, an assignment expression. Here are some examples:

```
var number;
var name;
var rate = 1.2;
var greeting = "Hello, world!";
var flag = false;
```

Arithmetic and Math Library Functions in JavaScript

JavaScript utilizes the standard arithmetic operators:

- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

JavaScript also has a math library you can use for advanced functions such as square root, absolute value, and the trigonometric functions. The arithmetic operators follow the standard order of operations, and parentheses can be used to modify that order.

Example 1-1 shows some examples of performing arithmetic in JavaScript, as well as examples of using several of the mathematical functions.