**Table of Content**

Title Page

Copyright and Credits

Learning Javascript Data Structure and Algorithms Third Edition

Dedication

Packt Upsell

Why subscribe?

PacktPub.com

Contributors

About the author

About the reviewers

Packt is searching for authors like you

Preface

Who this book is for

What this book covers

To get the most out of this book

Download the example code files

Conventions used

Get in touch

Reviews

**1. JavaScript &#x2013; A Quick Overview**

JavaScript data structure and algorithms

Setting up the environment

The minimum setup to work with JavaScript

Using web servers

Node.js http-server

JavaScript basics

Variables

Scope variable

Operators

Truthy and falsy

Functions of the equals operators (== and ===)

Control structures

Conditional statements

Loops

Functions

Object-oriented programming in JavaScript

Debugging and tools

Debugging with VSCode

Summary

**2. ECMAScript and TypeScript Overview**

ECMAScript or JavaScript?

ES6, ES2015, ES7, ES2016, ES8, ES2017, and ES.Next

The compatibility table

Using Babel.js

ECMAScript 2015+ functionalities

let and const instead of var

Variables scope with let and const

Template literals

Arrow functions

Default parameter values for functions

Declaring the spread and rest operators

Enhanced object properties

Object-oriented programming with classes

Inheritance

Working with getters and setters

Exponentiation operator

Modules

Running ES2015 modules in the browser and with Node.js

Using native ES2015 imports in Node.js

Running ES2015 modules in the browser

ES2015+ backward compatibility

Introducing TypeScript

Type inference

Interfaces

Generics

Other TypeScript functionalities

TypeScript compile-time checking in JavaScript files

Summary

**3. Arrays**

Why should we use arrays?

Creating and initializing arrays

Accessing elements and iterating an array

Adding elements

Inserting an element at the end of the array

Using the push method

Inserting an element in the first position

Using the unshift method

Removing elements

Removing an element from the end of the array

Removing an element from the first position

Using the shift method

Adding and removing elements from a specific position

Two-dimensional and multi-dimensional arrays

Iterating the elements of two-dimensional arrays

Multi-dimensional arrays

References for JavaScript array methods

Joining multiple arrays

Iterator functions

Iterating using the every method

Iterating using the some method

Iterating using forEach

Using map and filter

Using the reduce method

ECMAScript 6 and new array functionalities

Iterating using the for...of loop

Using the @@iterator object

Array entries, keys, and values

Using the from method

Using the Array.of method

Using the fill method

Using the copyWithin method

Sorting elements

Custom sorting

Sorting strings

Searching

ECMAScript 2015 - the find and findIndex methods

ECMAScript 2016 - using the includes method

Outputting the array into a string

The TypedArray class

Arrays in TypeScript

Summary

**4. Stacks**

Creating a JavaScript data structure and algorithm library

The stack data structure

Creating an array-based Stack class

Pushing elements to the stack

Popping elements from the stack

Peeking the element from the top of the stack

Verifying whether the stack is empty

Clearing the elements of the stack

Using the Stack class

Creating a JavaScript object-based Stack class

Pushing elements to the stack

Verifying whether the stack is empty and its size

Popping elements from the stackPeeking the top of the stack and clearing it

Creating the toString method

Protecting the internal elements of the data structure

The underscore naming convention

ES2015 classes with scoped symbols

ES2015 classes with WeakMap

ECMAScript class field proposal

Solving problems using stacks

Converting decimal numbers to binary

The base converter algorithm

Summary

**5. Queues and Deques**

The queue data structure

Creating the Queue class

Enqueuing elements to the queue

Dequeuing elements from the queue

Peeking the element from the front of the queue

Verifying whether the queue is empty and its size

Clearing the queue

Creating the toString method

Using the Queue class

The deque data structure

Creating the Deque class

Adding elements to the front of the deque

Using the Deque class

Solving problems using queues and deques

The circular queue &#x2013; Hot Potato

Palindrome checker

JavaScript task queues

Summary

**6. Linked Lists**

The linked list data structure

Creating the LinkedList class

Pushing elements to the end of the linked list

Removing elements from the linked list from a specific position

Looping through the list until we get to the desired position

Refactoring the remove method

Inserting an element at any position

The indexOf method: returning the position of an element

Removing an element from the linked list

The isEmpty, size, and getHead methods

The toString method

Doubly linked lists

Inserting a new element at any position

Removing elements from any position

Circular linked lists

Inserting a new element at any position

Removing elements from any position

Sorted linked lists

Inserting elements in order

Creating the StackLinkedList class

Summary

**7. Sets**

Structuring a&#xA0;dataset

Creating a Set class

The has(element) method

The add method

The delete and clear methods

The size method

The values method

Using the Set class

Set operations

Set union

Set intersection

Improving the intersection method

Set difference

Subset

ECMAScript 2015&#xA0;&#x2013; the Set class

ES2015 Set class operations

Simulating the union operation

Simulating the intersection operation

Simulating the difference operation

Using the spread operator

Multisets or bags

Summary

**8. Dictionaries and Hashes**

The dictionary data structure

Creating the Dictionary class

Verifying whether a key exists in the dictionary

Setting a key and value in the dictionary and&#xA0;the ValuePair cla

ss

Removing a value from the dictionary

Retrieving a value from the dictionary

The keys, values, and valuePairs methods

Iterating each ValuePair of the dictionary with forEach

The clear, size, isEmpty, and toString methods

Using the Dictionary class

The hash table

Creating a HashTable class

Creating a hash function

Putting a key and a value in the hash table

Retrieving a value from the hash table

Removing a value from the hash table

Using the HashTable class

Hash table versus hash set

Handling collisions between hash tables

Separate chaining

The put method

The get method

The remove method

Linear probing

The put method

The get method

The remove method

Creating better hash functions

The ES2015 Map class

The ES2015 WeakMap and We

akSet classes

Summary

**9. Recursion**

Understanding recursion

Calculating the factorial of a number

Iterative factorial

Recursive factorial

The call stack

JavaScript limitation on the call stack size

The Fibonacci sequence

Iterative&#xA0;Fibonacci

Recursive Fibonacci

Fibonacci with memoization

Why use recursion? Is it faster?

Summary

**10. Trees**

The tree data structure

Tree terminology

The binary and binary search trees

Creating the Node and BinarySearchTree classes

Inserting a key into the BST

Tree traversal

In-order traversal

Pre-order traversal

Post-order traversal

Searching for values in a tree

Searching for minimum and maximum values

Searching for a specific value

Removing a node

Removing a leaf node

Removing a node with a left or right child

Removing a node with two children

Self-balancing trees

Adelson-Velskii and Landi&#x2019;s tree (AVL tree)

Height of a node and the balancing factor

Balancing operations&#x2013; AVL rotations

Left-left case: single rotation to the right

Right-right case: single rotation to the left

Left-right case: double rotation to the right

Right-left case &#x2013; double rotation to the left

Inserting a node in the AVL tree

Removing a node from the AVL tree

Red-Black tree

Inserting a node in the Red-Black tree

Verifying the Red-Black tree properties after insertion

Red-Black tree rotations

Summary

**11. Binary Heap and Heap Sort**

The binary heap data structure

Creating the MinHeap class

Binary tree array representation

Inserting a value into the heap

The sift up operation

Finding the minimum or maximum value from the heap

Extracting the minimum or maximum value from the heap

The sift down operation (heapify)

Creating the MaxHeap class

The heap sort algorithm

Summary

**12. Graphs**

Graph terminology

Directed and undirected graphs

Representing a graph

The adjacency matrix

The adjacency list

The incidence matrix

Creating the Graph class

Graph traversals

Breadth-first search (BFS)

Finding the shortest paths using BFS

Further study on the shortest paths algorithms

Depth-first search (DFS)

Exploring the DFS algorithm

Topological sorting using DFS

Shortest path algorithms

Dijkstra's algorithm

The Floyd-Warshall algorithm

Minimum spanning tree (MST)

Prim's algorithm

Kruskal's algorithm

Summary

**13. Sorting and Searching Algorithms**

Sorting algorithms

The bubble sort

The improved bubble sort

The selection sort

The insertion sort

The merge sort

The quick sort

The partition process

The quick sort in action

The counting sort

The bucket sort

The radix sort

Searching algorithms

The sequential search

The binary search

The interpolation search

Shuffle algorithms

The Fisher-Yates shuffle

Summary

**14. Algorithm Designs and Techniques**

Divide and conquer

Binary search

Dynamic programming

The minimum coin change problem

The knapsack problem

The longest common subsequence

Matrix chain multiplication

Greedy algorithms

The min-coin change problem

The fractional knapsack problem

Backtracking algorithms

Rat in a Maze

Sudoku Solver

Introduction to functional programming

Functional versus imperative programming

ES2015+ and functional programming

The JavaScript functional toolbox &#x2013;&#xA0;map, filter, and reduce

JavaScript functional libraries and data structures

Summary

**15. Algorithm Complexity**

Big O notation

Understanding big O notation

O(1)

O(n)

O(n2)

Comparing complexities

Data structures

Graphs

Sorting Algorithms

Searching Algorithms

Introduction to the NP-completeness theory

Impossible problems and heuristic algorithms

Having fun with algorithms

Summary

Other Books You May Enjoy

Leave a review - let other readers know what you think

**SET**

Sequential data structures such as arrays (lists), stacks, queues, and linked lists should sound familiar to you by now. In this chapter, we will cover a data structure called sets, which is also a sequential data structure that does not allow duplicated values.