

CONTENTS IN DETAIL

FOREWORD by Dan Abramov	xvii
--------------------------------	-------------

ACKNOWLEDGMENTS	xix
------------------------	------------

INTRODUCTION	xxi
---------------------	------------

The Road to ECMAScript 6	xxi
About This Book.	xxii
Browser and Node.js Compatibility.	xxiii
Who This Book Is For.	xxiii
Overview	xxiii
Conventions Used	xxiv
Help and Support	xxv

1	
BLOCK BINDINGS	1

var Declarations and Hoisting	2
Block-Level Declarations	3
let Declarations	3
No Redeclaration	4
const Declarations	4
The Temporal Dead Zone.	6
Block Bindings in Loops	7
Functions in Loops	8
let Declarations in Loops	9
const Declarations in Loops	10
Global Block Bindings	11
Emerging Best Practices for Block Bindings	12
Summary	12

2	
STRINGS AND REGULAR EXPRESSIONS	13

Better Unicode Support	13
UTF-16 Code Points.	14
The codePointAt() Method	15
The String.fromCodePoint() Method	16
The normalize() Method	16
The Regular Expression u Flag	18
Other String Changes	19
Methods for Identifying Substrings.	19
The repeat() Method	20
Other Regular Expression Changes	21
The Regular Expression y Flag	21
Duplicating Regular Expressions	23
The flags Property	24

Template Literals	25
Basic Syntax	26
Multiline Strings	26
Making Substitutions	28
Tagged Templates	29
Summary	32

3 FUNCTIONS 35

Functions with Default Parameter Values	36
Simulating Default Parameter Values in ECMAScript 5	36
Default Parameter Values in ECMAScript 6	37
How Default Parameter Values Affect the arguments Object	38
Default Parameter Expressions	40
Default Parameter TDZ	41
Working with Unnamed Parameters	43
Unnamed Parameters in ECMAScript 5	43
Rest Parameters	44
Increased Capabilities of the Function Constructor	46
The Spread Operator	47
The name Property	48
Choosing Appropriate Names	48
Special Cases of the name Property	49
Clarifying the Dual Purpose of Functions	50
Determining How a Function Was Called in ECMAScript 5	50
The new.target Metaproperty	51
Block-Level Functions	52
Deciding When to Use Block-Level Functions	53
Block-Level Functions in Non-Strict Mode	54
Arrow Functions	54
Arrow Function Syntax	55
Creating Immediately Invoked Function Expressions	57
No this Binding	58
Arrow Functions and Arrays	60
No arguments Binding	60
Identifying Arrow Functions	61
Tail Call Optimization	61
How Tail Calls Are Different in ECMAScript 6	62
How to Harness Tail Call Optimization	63
Summary	64

4 EXPANDED OBJECT FUNCTIONALITY 67

Object Categories	68
Object Literal Syntax Extensions	68
Property Initializer Shorthand	68
Concise Methods	69
Computed Property Names	70

New Methods	71
The Object.is() Method	72
The Object.assign() Method	72
Duplicate Object Literal Properties	75
Own Property Enumeration Order	75
Enhancements for Prototypes	76
Changing an Object's Prototype	76
Easy Prototype Access with Super References	77
A Formal Method Definition	80
Summary	81

5

DESTRUCTURING FOR EASIER DATA ACCESS 83

Why Is Destructuring Useful?	84
Object Destructuring	84
Destructuring Assignment	85
Default Values	86
Assigning to Different Local Variable Names	87
Nested Object Destructuring	88
Array Destructuring	90
Destructuring Assignment	90
Default Values	92
Nested Array Destructuring	92
Rest Items	92
Mixed Destructuring	93
Destructured Parameters	94
Destructured Parameters Are Required	95
Default Values for Destructured Parameters	96
Summary	97

6

SYMBOLS AND SYMBOL PROPERTIES 99

Creating Symbols	100
Using Symbols	101
Sharing Symbols	102
Symbol Coercion	103
Retrieving Symbol Properties	104
Exposing Internal Operations with Well-Known Symbols	105
The Symbol.hasInstance Method	106
The Symbol.isConcatSpreadable Property	107
The Symbol.match, Symbol.replace, Symbol.search, and Symbol.split Properties	109
The Symbol.toPrimitive Method	111
The Symbol.toStringTag Property	112
The Symbol.unscopables Property	115
Summary	117

7		
SETS AND MAPS		119
Sets and Maps in ECMAScript 5		120
Problems with Workarounds		121
Sets in ECMAScript 6		122
Creating Sets and Adding Items		122
Removing Items		123
The forEach() Method for Sets		124
Converting a Set to an Array		126
Weak Sets		127
Maps in ECMAScript 6		129
Map Methods		130
Map Initialization		131
The forEach() Method for Maps		131
Weak Maps		132
Summary		136

8		
ITERATORS AND GENERATORS		137
The Loop Problem		138
What Are Iterators?		138
What Are Generators?		139
Generator Function Expressions		141
Generator Object Methods		142
Iterables and for-of Loops		142
Accessing the Default Iterator		143
Creating Iterables		144
Built-In Iterators		145
Collection Iterators		145
String Iterators		149
NodeList Iterators		151
The Spread Operator and Nonarray Iterables		151
Advanced Iterator Functionality		152
Passing Arguments to Iterators		152
Throwing Errors in Iterators		154
Generator Return Statements		155
Delegating Generators		156
Asynchronous Task Running		159
A Simple Task Runner		159
Task Running with Data		160
An Asynchronous Task Runner		161
Summary		164

9		
INTRODUCING JAVASCRIPT CLASSES		165
Class-Like Structures in ECMAScript 5		166
Class Declarations		166
A Basic Class Declaration		166
Why Use the Class Syntax?		167

Class Expressions	169
A Basic Class Expression	169
Named Class Expressions	170
Classes as First-Class Citizens	172
Accessor Properties	173
Computed Member Names	174
Generator Methods	175
Static Members	176
Inheritance with Derived Classes	178
Shadowing Class Methods	180
Inherited Static Members	181
Derived Classes from Expressions	181
Inheriting from Built-Ins	184
The Symbol.species Property	185
Using new.target in Class Constructors	188
Summary	189

10

IMPROVED ARRAY CAPABILITIES 191

Creating Arrays	191
The Array.of() Method	192
The Array.from() Method	193
New Methods on All Arrays	196
The find() and findIndex() Methods	196
The fill() Method	197
The copyWithin() Method	197
Typed Arrays	198
Numeric Data Types	199
Array Buffers	199
Manipulating Array Buffers with Views	200
Similarities Between Typed and Regular Arrays	207
Common Methods	207
The Same Iterators	208
The of() and from() Methods	208
Differences Between Typed and Regular Arrays	209
Behavioral Differences	209
Missing Methods	210
Additional Methods	211
Summary	212

11

PROMISES AND ASYNCHRONOUS PROGRAMMING 213

Asynchronous Programming Background	214
The Event Model	214
The Callback Pattern	215
Promise Basics	217
The Promise Life Cycle	217
Creating Unsettled Promises	219
Creating Settled Promises	221
Executor Errors	224

Global Promise Rejection Handling	224
Node.js Rejection Handling	225
Browser Rejection Handling	227
Chaining Promises	228
Catching Errors	229
Returning Values in Promise Chains	230
Returning Promises in Promise Chains	231
Responding to Multiple Promises	233
The Promise.all() Method	234
The Promise.race() Method	235
Inheriting from Promises	236
Promise-Based Asynchronous Task Running	237
Summary	241

12

PROXIES AND THE REFLECTION API 243

The Array Problem	244
Introducing Proxies and Reflection	244
Creating a Simple Proxy	245
Validating Properties Using the set Trap	246
Object Shape Validation Using the get Trap	247
Hiding Property Existence Using the has Trap	249
Preventing Property Deletion with the deleteProperty Trap	250
Prototype Proxy Traps	252
How Prototype Proxy Traps Work	252
Why Two Sets of Methods?	254
Object Extensibility Traps	255
Two Basic Examples	255
Duplicate Extensibility Methods	256
Property Descriptor Traps	257
Blocking Object.defineProperty()	258
Descriptor Object Restrictions	259
Duplicate Descriptor Methods	260
The ownKeys Trap	261
Function Proxies with the apply and construct Traps	262
Validating Function Parameters	264
Calling Constructors Without new	265
Overriding Abstract Base Class Constructors	266
Callable Class Constructors	267
Revocable Proxies	268
Solving the Array Problem	269
Detecting Array Indexes	270
Increasing length When Adding New Elements	270
Deleting Elements When Reducing length	272
Implementing the MyArray Class	273
Using a Proxy as a Prototype	275
Using the get Trap on a Prototype	276
Using the set Trap on a Prototype	277
Using the has Trap on a Prototype	278
Proxies as Prototypes on Classes	279
Summary	282

13		
ENCAPSULATING CODE WITH MODULES		283
What Are Modules?		283
Basic Exporting		284
Basic Importing		285
Importing a Single Binding		286
Importing Multiple Bindings		286
Importing an Entire Module		286
A Subtle Quirk of Imported Bindings		288
Renaming Exports and Imports		288
Default Values in Modules		289
Exporting Default Values		289
Importing Default Values		290
Re-exporting a Binding		291
Importing Without Bindings		292
Loading Modules		293
Using Modules in Web Browsers		293
Browser Module Specifier Resolution		297
Summary		298

A		
MINOR CHANGES IN ECMAScript 6		299
Working with Integers		299
Identifying Integers		300
Safe Integers		300
New Math Methods		301
Unicode Identifiers		302
Formalizing the <code>__proto__</code> Property		303

B		
UNDERSTANDING ECMAScript 7 (2016)		305
The Exponentiation Operator		306
Order of Operations		306
Operand Restriction		306
The <code>Array.prototype.includes()</code> Method		307
How to Use <code>Array.prototype.includes()</code>		307
Value Comparison		308
A Change to Function-Scoped Strict Mode		308

INDEX		311
--------------	--	------------