TECH TODAY

PROFESSIONAL JAVASCRIPT® FOR WEB DEVELOPERS

5TH EDITION

MATT FRISBIE

FOREWORD BY JOHN HUBBERTS
Founding Principal Engineer at Roboto

WILEY

PROFESSIONAL JAVASCRIPT® FOR WEB DEVELOPERS

FOREWORD xliii

INTRODUCTION xlv

CHAPTER 1 What Is JavaScript? 1
CHAPTER 2 JavaScript in HTML. 9
CHAPTER 3 Language Basics 19
CHAPTER 4 Variables, Scope, and Memory. 97
CHAPTER 5 Basic Reference Types 119
CHAPTER 6 Advanced Reference Types 159
CHAPTER 7 Iterators and Generators 217
CHAPTER 8 Objects, Classes, and Object-Oriented Programming 251
CHAPTER 9 Proxies and Reflect. 323
CHAPTER 10 Functions 349
CHAPTER 11 Promises and Async/Await 383
CHAPTER 12 The Browser Object Model 435
CHAPTER 13 The Document Object Model. 459
CHAPTER 14 DOM Extensions. 507
CHAPTER 15 Events 557
CHAPTER 16 Animation and Graphics with Canvas 609
CHAPTER 17 Scripting Forms 645
CHAPTER 18 JavaScript APIs 675
CHAPTER 19 Error Handling and Debugging 765
CHAPTER 20 JSON 791
CHAPTER 21 Network Requests and Remote Resources 801
CHAPTER 22 Client-Side Storage 839
CHAPTER 23 Modules 861
CHAPTER 24 Workers. 887

Continues

CHAPTER 25	Best Practices	955
APPENDIX A	ES.Next	987
APPENDIX B	Strict Mode	991
APPENDIX C	JavaScript Libraries and Frameworks	999
APPENDIX D	JavaScript Tools	1005

INDEX **1017**

PROFESSIONAL JavaScript® for Web Developers

PROFESSIONAL

JavaScript® for Web Developers

Fifth Edition

Matt Frisbie

WILEY

Copyright © 2024 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada and the United Kingdom.

ISBNs: 9781394193219 (Paperback), 9781394193233 (ePDF), 9781394193226 (ePub)

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at `www.copyright.com`. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at `www.wiley.com/go/permission`.

Trademarks: WILEY and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. JavaScript is a registered trademark of Oracle America, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

If you believe you've found a mistake in this book, please bring it to our attention by emailing our reader support team at `wileysupport@wiley.com` with the subject line "Possible Book Errata Submission."

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at `www.wiley.com`.

Library of Congress Control Number: 2023934359

Cover image: © CSA Images/Getty Images
Cover design: Wiley

To my wonderful and patient wife, Jordan, whom I assured three books ago that I wasn't going to write any more books.

ABOUT THE AUTHOR

MATT FRISBIE has worked in web development for over a decade. During that time, he has worked as an independent software consultant, a startup co-founder, an engineer at a Big Four tech company, and the first engineer at a Y Combinator startup that would eventually become a billion-dollar company. He also maintains a popular open-source project. At Google, Matt worked on both the AdSense and AMP platforms; his code still runs on most of the planet's web browsing devices. Prior to this, Matt was the first engineer at DoorDash, where he helped lay the foundation for a company that has become the leader in online food delivery. Matt has written four other books: *Building Browser Extensions (Apress, 2022); Professional JavaScript for Web Developers, fourth edition (Wiley, 2019); Angular 2 Cookbook (Packt, 2017);* and *AngularJS Web Application Development Cookbook (Packt, 2014)*. He also recorded several video series. He is a frequent guest on podcasts, speaks at frontend meetups, and is a level-1 sommelier. Matt majored in Computer Engineering at the University of Illinois Urbana-Champaign. You can reach him on Twitter as `@mattfriz` or on the Web at `mattfriz.com`.

ABOUT THE TECHNICAL EDITOR

BEN LUO is a software engineer with experience in full-stack web development, natural language processing, and DevOps. Ben, whose professional interests lie in the healthcare domain, is a proponent for user-conscious and accessible UI design in clinical and patient-facing applications. He holds a Master of Science in computer science from the University of Illinois Chicago.

Ben plays competitive fighting games and is involved in community organization efforts for online tournaments. In addition, he enjoys building and using mechanical keyboards and stenotypes. As an avid tinkerer, he is involved in the development of open-source firmware for both custom arcade video game controllers and mechanical keyboards.

ACKNOWLEDGMENTS

THANKS TO WILEY FOR allowing me to continue as the steward of this book. Writing consecutive editions of *Professional JavaScript for Web Developers* has been a complete privilege. Thanks to the Wiley staff, specifically Jim Minatel, who offered me the opportunity.

Special thanks to Patrick Walsh for helping bring this book to fruition. Having a great primary editor makes the process so much more enjoyable, and he was terrific to work with. I'd also like to thank everyone who provided feedback on the book's drafts: Ben Luo, Archana Pragash, Judy Flynn, Pete Gaughan, Ashirvad Moses, and everyone else behind the scenes. This book would be nothing without your efforts.

Finally, I'd like to thank John Hubberts for writing the Foreword. I have been close friends with John for nearly 20 years, and it did not take me long to learn he has a ferociously hungry mind and an infectious passion for technology and engineering. I am honored that he agreed to contribute to this book.

CONTENTS

FOREWORD *xliii*

INTRODUCTION *xlv*

CHAPTER 1: WHAT IS JAVASCRIPT? 1

A Short History 2
Javascript Implementations 3
ECMAScript 3
ECMAScript Editions 4
What Does ECMAScript Conformance Mean? 5
The Document Object Model 5
Why the DOM Is Necessary 6
DOM Levels 6
Other DOMs 7
The Browser Object Model 8
Summary 8

CHAPTER 2: JAVASCRIPT IN HTML 9

The <*Script*> Element 9
Tag Placement 12
Deferred Scripts 13
Asynchronous Scripts 13
Dynamic Script Loading 14
Inline Code Versus External Files 14
Document Modes 15
The <*Noscript*> Element 16
Summary 17

CHAPTER 3: LANGUAGE BASICS 19

Syntax 20
Case-Sensitivity 20
Identifiers 20
Comments 20
Strict Mode 21
Statements 21
Keywords and Reserved Words 22
Variables 22

The var Keyword 23
var Declaration Scope 23
var Declaration Hoisting 24
let Declarations 25
Temporal Dead Zone 26
Global Declarations 26
Conditional Declaration 26
let Declaration in *for* Loops 27
const Declarations 28
Declaration Styles and Best Practices 29
Don't Use *var* 29
Prefer *const* Over *let* 29
Data Types 30
The *typeof* Operator 30
The Undefined Type 31
The Null Type 32
The Boolean Type 33
The Number Type 34
Floating-Point Values 35
Numeric Separators 36
Range of Values 36
NaN 37
Number Conversions 38
The BigInt Type 40
Conversion 41
Operators 41
Static Methods 42
JSON 42
The String Type 43
Character Literals 43
The Nature of Strings 44
Converting to a String 44
Template Literals 45
Interpolation 46
Template Literal Tag Functions 47
Raw Strings 49
The Symbol Type 50
Basic Symbol Use 50
Using the Global Symbol Registry 51
Using Symbols as Properties 52
Well-Known Symbols 53
The Object Type 63

Operators 64
Unary Operators 64
Increment/Decrement 64
Unary Plus and Minus 66
Bitwise Operators 67
Bitwise NOT 69
Bitwise AND 69
Bitwise OR 70
Bitwise XOR 71
Left Shift 71
Signed Right Shift 72
Unsigned Right Shift 73
Boolean Operators 73
Logical NOT 73
Logical AND 74
Logical OR 75
Multiplicative Operators 76
Multiply 76
Divide 77
Modulus 77
Exponentiation Operator 78
Additive Operators 78
Add 78
Subtract 79
Relational Operators 80
Equality Operators 82
Equal and Not Equal 82
Identically Equal and Not Identically Equal 83
Conditional Operator 84
Nullish Coalescing Operator 84
Assignment Operators 85
Comma Operator 86
Statements 86
The *if* Statement 86
The *do-while* Statement 87
The *while* Statement 87
The *for* Statement 88
The *for-in* Statement 89
The *for-of* Statement 89
Labeled Statements 90
The *break* and *continue* Statements 90

The *with* Statement 91
The *switch* Statement 92
Functions 94
Summary 95

CHAPTER 4: VARIABLES, SCOPE, AND MEMORY 97

Primitive and Reference Values 97
Dynamic Properties 98
Copying Values 99
Argument Passing 100
Determining Type 101
Execution Context and Scope 102
Scope Chain Augmentation 105
Variable Declaration 105
Function Scope Declaration Using var 105
Block Scope Declaration Using *let* 107
Constant Declaration Using *const* 108
Identifier Lookup 109
Garbage Collection 110
Performance 111
Managing Memory 111
Performance Boosts with const and let Declarations 112
Hidden Classes and the delete Operation 112
Memory Leaks 113
Static Allocation and Object Pools 114
Summary 116

CHAPTER 5: BASIC REFERENCE TYPES 119

The Date Type 120
Inherited Methods 122
Date-Formatting Methods 122
Date/Time Component Methods 123
The Regexp Type 125
RegExp Instance Properties 128
RegExp Instance Methods 129
Primitive Wrapper Types 131
The Boolean Type 133
The Number Type 133
The *isInteger()* Method and Safe Integers 135

The String Type 136
The JavaScript Character 136
The *normalize()* Method 139
String-Manipulation Methods 140
String Location Methods 142
String Inclusion Methods 143
The *trim()* Method 144
The *repeat()* Method 144
The *padStart()* and *padEnd()* Methods 144
String Iterators and Destructuring 145
String Case Methods 145
String Pattern-Matching Methods 146
The *localeCompare()* Method 149
Singleton Built-In Objects 150
The Global Object 150
URI-Encoding Methods 150
The *eval()* Method 151
Global Object Properties 152
The Window Object 153
The Math Object 154
Math Object Properties 154
The *min()* and *max()* Methods 155
Rounding Methods 155
The *random()* Method 156
Other Methods 157
Summary 158

CHAPTER 6: ADVANCED REFERENCE TYPES 159

The Object Type 159
The Array Type 162
Creating Arrays 162
Array Holes 165
Indexing into Arrays 166
Detecting Arrays 168
Iterator Methods 168
Copy and Fill Methods 169
Spread Operator 171
Rest Operator 171
Conversion Methods 172
Stack Methods 173
Queue Methods 174

Reordering and Sorting Methods 175
Manipulation Methods 178
Search and Location Methods 180
Iterative Methods 182
Reduction Methods 183
Flattening Methods 184
Typed Arrays 186
History 186
WebGL 186
Emergence of Typed Arrays 187
Using ArrayBuffers 187
DataViews 188
ElementType 189
Big-Endian and Little-Endian 190
Corner Cases 191
Typed Arrays 192
Typed Array Behavior 193
Merging, Copying, and Changing Typed Arrays 194
Underflow and Overflow 195
The Map Type 196
Basic API 196
Order and Iteration 199
Choosing Between Objects and Maps 201
Keys 201
Memory Profile 201
Insertion Performance 201
Lookup Performance 201
Delete Performance 202
The Set Type 202
Basic API 202
Order and Iteration 204
Weak References 205
WeakRef 206
FinalizationRegistry 206
The Weakmap Type 207
Basic API 207
Weak Keys 209
Non-Iterable Keys 210
Utility 210
The Weakset Type 210
Basic API 211

Weak Keys 212
Non-Iterable Values 213
Utility 213
Iteration and Spread Operators 213
Summary 215

CHAPTER 7: ITERATORS AND GENERATORS 217

Introduction To Iteration 217
The Iterator Pattern 218
The Iterable Protocol 219
The Iterator Protocol 221
Custom Iterator Definition 223
Early Termination of Iterators 225
Generators 227
Generator Basics 227
Interrupting Execution with *yield* 229
Using a Generator Object as an Iterable 231
Using *yield* for Input and Output 231
Yielding an Iterable 233
Recursive Algorithms Using *yield** 235
Using a Generator as the Default Iterator 238
Early Termination of Generators 238
The *return()* Method 239
The *throw()* Method 240
Asynchronous Iteration 241
Creating and Using an Async Iterator 241
Understanding the Async Iterator Queue 243
Async Iterator *reject()* Handling 245
Manual Async Iteration Using *next()* 246
Top-Level Async Loops 246
Implementing Observables 247
Summary 249

CHAPTER 8: OBJECTS, CLASSES, AND OBJECT-ORIENTED PROGRAMMING 251

Understanding Objects 251
Types of Properties 252
Data Properties 252
Accessor Properties 254
Accessing Object Properties 255
Chaining Properties 256

Object Static Methods 257
Controlling Object Mutability 258
Freezing Objects 258
Sealing Objects 259
Controlling Extensibility 260
Defining Multiple Properties 260
Reading Property Attributes 261
Merging Objects 263
Object Identity and Equality 265
Enhanced Object Syntax 266
Property Value Shorthand 266
Computed Property Keys 267
Concise Method Syntax 268
Object Destructuring 269
Nested Destructuring 271
Partial Destructuring Completion 273
Parameter Context Matching 273
Rest Operator 274
Spread Operator 274
Object Creation 275
Overview 275
The Function Constructor Pattern 276
Constructors as Functions 278
Problems with Constructors 279
The Prototype Pattern 280
How Prototypes Work 281
Understanding the Prototype Hierarchy 284
Prototypes and the *in* Operator 287
Property Enumeration Order 290
Object Iteration 290
Dynamic Nature of Prototypes 292
Native Object Prototypes 293
Problems with Prototypes 294
Prototype Inheritance 295
Prototype Chaining 295
Default Prototypes 296
Prototype and Instance Relationships 297
Working with Methods 298
Problems with Prototype Chaining 299
Classes 299
Class Definition Basics 300

Class Composition 300
The Class Constructor 301
Instantiation 302
Understanding Classes as Special Functions 304
Instance, Prototype, and Class Members 305
Instance Members 305
Class Field Declarations 306
Prototype Methods and Accessors 307
Private Class Members 308
Static Class Methods and Accessors 310
Static Initialization Blocks 311
Iterator and Generator Methods 312
Class Inheritance 313
Inheritance Basics 313
Constructors, HomeObjects, and *super()* 314
Abstract Base Classes 317
Inheriting from Built-In Types 318
Class Mixins 319
Summary 321

CHAPTER 9: PROXIES AND REFLECT 323

Proxy Fundamentals 324
Creating a Passthrough Proxy 324
Defining Traps 325
Trap Parameters and the Reflect API 326
Trap Invariants 328
Revocable Proxies 329
Utility of the Reflect API 329
Reflect API vs. Object API 330
Status Flags 330
Supplanting Operators with First-Class Functions 331
Safe Function Application 331
Proxying a Proxy 331
Proxy Considerations and Shortcomings 332
this Inside a Proxy 332
Proxies and Internal Slots 333
Proxy Traps and Reflect Methods 333
get() 334
set() 334
has() 335
defineProperty() 336

getOwnPropertyDescriptor() 337
deleteProperty() 338
ownKeys() 339
getPrototypeOf() 339
setPrototypeOf() 340
isExtensible() 341
preventExtensions() 341
apply() 342
construct() 343
Proxy Patterns 343
Tracking Property Access 344
Hidden Properties 344
Property Validation 345
Function and Constructor Parameter Validation 345
Data Binding and Observables 346
Summary 347

CHAPTER 10: FUNCTIONS 349

Arrow Functions 350
Function Names 351
Understanding Arguments 353
Arguments in Arrow Functions 355
No Overloading 355
Default Parameter Values 356
Default Parameter Scope and Temporal Dead Zone 358
Spread Arguments and Rest Parameters 359
Spread Arguments 359
Rest Parameter 360
Function Declarations Versus Function Expressions 361
Functions as Values 363
Function Internals 364
arguments 364
this 365
caller 367
new.target 368
Function Properties and Methods 368
Using *apply()*, *call()*, and *bind()* 369
Serializing Functions 371
Recursion 371
Tail Call Optimization 372

Tail Call Optimization Requirements 373
Coding for Tail Call Optimization 375
Closures 376
The *this* Object 379
Memory Leaks 380
Immediately Invoked Function Expressions 381
Summary 382

CHAPTER 11: PROMISES AND ASYNC/AWAIT 383

Introduction to Asynchronous Programming 384
Synchronous vs. Asynchronous JavaScript 384
Legacy Asynchronous Programming Patterns 385
Returning Asynchronous Values 386
Handling Failure 386
Nesting Asynchronous Callbacks 387
Promises 387
The Promises/A+ Specification 387
Promise Basics 388
The Promise State Machine 388
Fulfilled Values, Rejection Reasons, and Utility of Promises 389
Controlling Promise State with the Executor 389
Promise Casting with *Promise.resolve()* 390
Promise Rejection with *Promise.reject()* 391
Synchronous/Asynchronous Execution Duality 392
Promise Instance Methods 392
Implementing the thenable Interface 393
Using *Promise.prototype.then()* 393
Using *Promise.prototype.catch()* 396
Using *Promise.prototype.finally()* 397
Non-Reentrant Promise Methods 398
Sibling Handler Order of Execution 400
Fulfilled Value and Rejected Reason Passing 400
Rejecting Promises and Rejection Error Handling 401
Promise Chaining and Composition 403
Promise Chaining 403
Using *Promise.all()* 405
Using *Promise.allSettled()* 407
Using *Promise.race()* 408
Using *Promise.any()* 409
Serial Promise Composition 410
Promises and the Microtask Queue 412

Avoiding Unhandled Rejections 412
Promise Extensions 413
Promise Canceling 414
Promise Progress Notifications 415
Async Functions 417
Async Function Basics 418
The *async* keyword 418
The *await* keyword 420
Restrictions on *await* 422
Strategies for Async Functions 426
Implementing *Sleep()* 426
Maximizing Parallelization 426
Serial Promise Execution 429
Stack Traces and Memory Management 430
Rejection-Safe Parallelization 431
Summary 432

CHAPTER 12: THE BROWSER OBJECT MODEL 435

The Window Object 436
The Global Scope 436
The *globalThis* property 437
Window Relationships 437
Window Position and Pixel Ratio 438
Pixel Ratios 438
Window Size 438
Window Viewport Position 439
Navigating and Opening Windows 440
Popping Up Windows 440
Security Restrictions 443
Pop-Up Blockers 443
Intervals and Timeouts 444
System Dialogs 446
The Location Object 448
Manipulating the Location 449
The Navigator Object 451
Registering Handlers 453
The Screen Object 454
The History Object 454
Navigation 455
History State Management 456
Summary 457

surrogate pair, 138
surroundContents() method, 531–532, 672
swapColors() function, 104
switch statement, 92–93, 973
Symbol property, 50–51, 153
Symbol type, 50–62
Symbol.asyncIterator function, 53–54
Symbol.for(), 51
Symbol.hasInstance function, 54–55
Symbol.isConcatSpreadable function, 55–56
Symbol.iterator function, 56–57
Symbol.match function, 57–58
Symbol.replace function, 58
symbols
  using as properties, 52–53
  as weakmap keys, 988
  well-known, 53–62
Symbol.search function, 59
Symbol.species function, 59–60
Symbol.split function, 60–61
Symbol.toPrimitive function, 61
Symbol.toStringTag function, 61–62
Symbol.unscopables function, 62
synchronous programming, asynchronous programming *versus,* 384–385
synchronous tasks, 403
synchronous/asynchronous execution duality, 392
syntax
  about, 19
  case-sensitivity, 20
  comments, 20–21
  identifiers, 20
  statements, 21–22
  strict mode, 21
SyntaxError object, 153, 770
system dialogs, 446–448
SystemJS, 1007

T

tabIndex property, 650
tag functions, 47–48
tagging modules, 873–874
tail call optimization, 372–376
takeRecords() method, 537
target property, 547, 566, 646
task runners, 980, 1010–1011
template literals, 45–46, 47–48
template scripts, 736
<template> tags, 735–736
templates, HTML, 733–736
temporal dead zone, 26, 108, 358
terminate() method, 894, 897, 920, 927
test() function, 23, 130
text
  decoding, 691–694
  drawing, 618–620
  encoding, 689–691
  selecting, 654–655
text boxes, scripting, 653–661
text events, 580–583
text() method, 803, 830
text nodes
  creating, 480–481
  normalizing, 481–482
  splitting, 482
text read/write, Clipboard API, 685
Text type
  about, 479–480
  creating text nodes, 480–481
  normalizing text nodes, 481–482
  splitting text nodes, 482
textAlign property, 618
textBaseline property, 618
textInput event, 582–583
textures, WebGL, 640–641
then() method, 400, 402–403, 405, 419
Thenable interface, 393
this object
  about, 365–367
  closures and, 379–380
  inside proxies, 332–333
threads, compared with workers, 888–889
three.js, 1002
throttling performance, 612–613
throw() method, 240
throwing errors, 769, 772–774, 788–789
timeout property, 703
timeouts, 444–446
timeupdate event, 712
Timing APIs
  about, 728–729
  High Resolution Time API, 729–730
  Performance Timeline API, 730–732
title property, 512
toDataURL() method, 614
toDateString() method, 122
toExponential() method, 134
toFixed() method, 134
toggle() method, 495
toGMTSring() method, 123
toISOString() method, 798–799
toJSON() method, 798–799
toLocaleDateString() method, 123
toLocaleLowerCase() method, 145–146
toLocaleString() method, 63, 122, 131, 133, 136, 172–173, 371
toLocaleTimeString() method, 123
toLocaleUpperCase() method, 145–146
toLowerCase() method, 145–146
ToObject() method, 271

tooLong property, 661
top property, 454
toPrecision() method, 134–135
toString() method, 38, 44–45, 63, 66, 79, 80, 122, 131, 133, 136, 172–173, 364, 368, 371, 672, 753
totalBytes property, 711
toTimeString() method, 122
touch devices, support for, 579–580
touch events, 594–597
toUpperCase() method, 145–146
toUTCString() method, 123
transaction() method, 849–850
transactions, IndexedDB, 849–850
transferable objects, 907–908
transform() method, 620, 723–724
transform streams, 718, 723–724
transformations, 620–622
translate() method, 620
transparency, of variables, 958–959
transpilation, 983, 1008
traps
  defining, 325–326
  invariants, 328–329
  parameters, 326–328
traversals
  about, 519–521
  NodeIterator type, 521–523
  TreeWalker, 524–525
tree shaking, 980
TreeWalker, 524–525
trim() method, 144
trimLeft() method, 144
trimRight() method, 144
trusted property, 566
try/catch statement, 27, 767–772, 774
2D, 614–629
two's complement, 68
type. *See also specific types*
  coercion errors for, 776–778
  determining, 101–102
type property, 10, 512, 513, 547, 566, 650, 694, 707, 820, 897
typed arrays
  about, 186, 192–196
  ArrayBuffers, 187–188
  behaviors, 193
  changing, 194–195
  copying, 194–195
  DataViews, 188–192
  emergence of, 187
  history, 186–187
  merging, 194–195
TypeError property, 153, 770
typeMismatch property, 661
typeof operator, 30–31, 102
TypeScript, 1008

U

Uglify, 1013
UI events, 569–572
unary minus operator, 66–67
unary operators, 64–67
unary plus operator, 39, 66–67
undefined property, 152
Undefined type, 31–32
underflow, of values, 195–196
underscore (_), 20
Underscore.js library, 1002
unhandled rejections, 412–413
unicode property, 128
unique property, 856
unit testing, 1013–1014
universal modules, 871–872
unload event, 571–572
unregister(), 206–207
unregister() method, 927
unshift() method, 175, 194–195
unsigned right shift (>>>), 73
unwrapping keys, 762–763
update() method, 852, 927
updateAnimations() method, 610
updateViaCache, managing service workers file caching with, 944–945
updating service workers, 943–944
upgrading
  custom elements, 749–750
  from GLSL 100 to GLSL 300, 638
URI-encoding methods, 150–151
URIError property, 153, 770
URL APIs, 726–728
URL object, 726
url property, 820
URLSearchParams object, 727–728
User Timing API, 731
userAgent property, 453
username property, 726

V

valid property, 661
validation
  about, 660–661, 981–982
  parameters of constructors, 345–346
  parameters of functions, 345–346
  properties, 345
value property, 650
valueMissing property, 661
valueOf() method, 38, 63, 66, 80, 81, 122, 133, 136, 172–173, 368
values
  copying, 99–100

floating-point, 35–36
functions as, 363–364
JSON, 792
passing to shaders, 636–637
range of, 36–37
values() method, 204
var declaration, 23–24, 105–107
var keyword, avoiding, 967
variable declaration
about, 105
block scope declaration using let, 107–108
constant declaration using const, 108–109
function scope declaration using var, 105–107
identifier lookup, 109–110
variables
about, 22
const declaration, 28–29
declaration styles, 29–30
let declaration, 25–28
naming, 957–958
strict mode, 992
types of transparency, 958–959
var declaration, 23–24
vendor property, 453, 704
vendorSub property, 453
verifying messages with asymmetric keys, 760–761
vibrate() method, 453
videoHeight property, 711
videoWidth property, 711
View property, 566
viewports, WebGL, 632–633
Visual Studio Code, 1010
volume property, 711
volumechange event, 713
Vue framework, 1000

W

wait queue, 683
waiting property, 713, 926
waitUntil() method, 952
warn() method, 782
weak references, 205–207
weakmap keys, symbols as, 988
WeakMap type
about, 207
basic API, 207–209
non-iterable keys, 210
utility, 210
weak keys, 209
WeakRef class, 206
WeakSet type
about, 210
basic API, 211–212
non-iterable keys, 213
utility, 213
weak keys, 209
web browsers. *See also specific browsers*
error reporting, 766
identifying, 703–708
web components
adding content, 746–747
custom elements, 744–750
shadow DOM, 736–744
Web Cryptography API
about, 750
random number generation, 750–751
SubtleCrypto object, 752–763
Web Sockets
events, 833–835
limits, 847
localStorage object, 846
restrictions, 847
sessionStorage object, 845–846
storage event, 846–847
Web Storage, 843–847
WebAssembly, 1009
webdriver property, 453
WebGL
about, 186–187, 629
basics of, 630–641
context, 630
WebGL1 *versus* WebGL2, 641–642
WebKit, 594
Webpack, 980, 1007
websites
automation systems, 1010–1011
build tools, 1010–1011
compilation/transpilation tools and static type systems, 1008
documentation generators, 1014–1015
editors, 1009–1010
formatters, 1011–1012
frameworks, 999–1001
Grunt, 980
Gulp, 980
high-performance script tools, 1009
jQuery, 490
libraries, 1001–1004
linters, 1011–1012
minifiers, 1012–1013
module bundlers, 1007–1008
module loaders, 1006–1007
Mozilla service worker cookbook, 922
npm, 980
"Offline Web Applications" Udacity course, 922
package managers, 1005–1006
Selectors API, 490
task runners, 1010–1011
unit testing, 1013–1014
Web Cryptography API, 750
Webpack, 980
WebStorm, 1010

wheel events, 573–580
while statement, 87
width property, 454
window object
  about, 153–154, 436
  global scope of, 436–437
  intervals, 444–446
  navigating windows, 440–443
  opening windows, 440–443
  pixel ratio, 438
  position, 438
  relationships, 437
  size, 438–439
  system dialogs, 446–448
  timeouts, 444–446
  viewport position, 439–440
window.open() method, 442, 443
windows. *See* window object
window.URL.createObjectURL() method, 698–699
wire weight, 982
Wise, David
  The Impact of Applicative Programming on Multiprocessing, 387
with statement, 91–92, 105, 969, 997
worker data transfer, 905–912
worker modules, 884
Worker object, 894
worker pools, 912–915
WorkerGlobalScope, 890–891
workers
  about, 887–888
  compared with threads, 888–889
  dedicated, 891–915
  service, 921–952
  shared, 915–921
  types of, 889
  WorkerGlobalScope, 890–891
wrapper libraries, 858
wrapping keys, 762–763
writable streams, 718, 721–722
write() method, 471–473, 721–722
WriteableStreamDefaultWriter, 722
writeIn() method, 471–473
writes, in Atomics API, 680
writing shaders, 634–635

X

XMLHttpRequest object, 802
XOR (^) operator, 71

Y

Yarn, 1006
yield, interrupting execution with, 229–237