索引

SYMBOLS AND NUMERICS

> (angle bracket), defined, 9
$ (dollar sign) prefix, 8
./, (dot slash) adding to require parameters, 43
\r\n delimiter, 85
[] (square brackets) for uploading multiple files, 134
404 (Not Found) status code, 117

A

absolute modules, defined, 41
accepts extension for Request, 156
accessors in v8, 25–26
acknowledgment, Socket.IO, 190–191
acknowledgments, TCP, 71
addEventListener API, 45–46
addMessage function, 187, 190
AJAX, 164–166, 180, 245
Amazon Linux, 9
angle bracket (>), defined, 9
ANSI escape codes (website), 66
Apache/PHP compared to Node, 28–29
app. users function, 212
app.configure function, 153
app.enable function, 153
app.error function, 155
apply method, 18
app.set function, 147, 148
app.use function, 159, 161
argv API, 63–64, 108
arity of a function, 19, 296
array JavaScript type, 16, 17
arrays, 23, 239–240, 242–243
assert module, 284, 290–291, 293
AssertionError, 290
asynchronous code
  advantage for Node, 32
  serial execution, 58, 59
  synchronous code, compared to, 30, 53–55
  testing, 295–297
authentication, MongoDB access, 217–218

B

base64, defined, 48
basicAuth middleware, 141–143
BDD (behavior-driven development), 297
belongsTo relationship type, 250
binary data, representing in JavaScript, 47–48
bind method, 24
block body declaration, 238
blocking code, 29–31, 32, 33, 46, 54
bodyParser middleware, 131–134, 208, 214, 251
boolean JavaScript type, 16
broadcast function, 171, 173–174, 186, 187
broadcasting issue, WebSocket, 177, 185–190
browser. *See also* code sharing
  Chrome, 1, 22
  Mocha, compiling for, 299–300
  Node's connection features, 94
  shimming browser APIs, 284
  WebSocket issues, 166, 177, 180
browserbuild project, 279, 285–288
Buffer object, 48
buffers, 47–48
bundling up client-side JavaScript, 128
byte orientation in TCP, 70–71

C

call method, 18
call stacks in v8, 32
case sensitive routes in Express, 153
casting in Mongoose, 225–226
Chrome, Google, 1, 22
chunked value for transfer encoding, 92–93
classes, 20, 46, 220, 254
CLI (command line interface), exploring APIs for, 63–66
client. *See also* browser
  bundling up JavaScript, 128
  createClient in MySQL, 234
  http.ClientRequest object, 105, 106
  IRC, 83–85
  mongo client, 207
  Socket.IO, 183–185
  telnet client, 71–73
  WebSocket, 169
client.end function, 236
client.query function, 236
close event, 77, 177, 180
closures, JavaScript, 19

cloud deployments, non-blocking IO advantages for, 33
code sharing (browser/server)
  browser APIs, shimming, 284
  browserbuild project, 285–288
  cross-browser inheritance, 284–285
  ECMA APIs, shimming, 282–283
  introduction, 279–280
  modules, exposing, 280–282
  Node APIs, shimming, 283–284
  summary, 288
Collection.insert function, 214
collections of documents in MongoDB, 205, 213–216
command line interface (CLI), exploring APIs for, 63–66
complex JavaScript types, 16
concurrency. *See* shared-state concurrency
configure function, 153
congestion control in TCP, 71
Connect API. *See also* middleware
  Connect-based website, creating, 119–120
  HTTP-based website, creating, 116–119
  introduction, 115–116
  summary, 144
connect event, 84, 180, 185–186
connect function, 84, 184, 220
connection event, 168, 181
Connection HTTP header, 92, 94
connections
  client/server communications as, 70
  HTTP, 93–94
  MongoDB access, 212–214
  node-mysql, 234
  redis, 267–268
  sequelize, 248–249
  tracking in TCP, 79–81
  WebSocket, 177
console object, 40
Content-Type HTTP header, 89, 99–100
contracts, event, defined, 47
controllers, separating HTML code from in Express, 146
convenience methods, Express, 155–157, 253
Cookie header, 134
createClient API, 234
createServer function
  Connect style, 120
  Express style, 147
  HTTP server, 112, 117
  TCP, 75
  WebSocket, 168
Crockford, Douglas (author)
  JavaScript: The Good Parts, 15
cross-browser inheritance, 284–285
crypto libraries, code sharing, 280
current working directory, 64–65

D

Dahl, Ryan (developer), 1
data deletion route, sequelize, 254–255
data event, 60–61, 77–79
data stream, client/server communication as, 70
data types, 207, 262–266
database access, non-blocking IO advantage for, 33. *See also* MongoDB; MySQL; Redis
datagram, defined, 70
:date dynamic token, 130
date manipulation toolkits, code sharing, 280
db.getLastError function, 220
db.query function, 241
default logging format, 129
delete class, 254
DELETE request, 99
deletion route, data, sequelize, 254–255
destroy command, 255
dev logging format, 129
_dirname, 64
disconnect event, 180
disconnecting and close event in WebSocket, 177
dispatchEvent API, 45–46
documents, MongoDB data as, 205, 206, 213–217, 222
dollar sign ($) prefix, 8
DOM API, code sharing (website), 282, 284
dot notation for nested keys in MongoDB, 222
dot slash (./), adding to require parameters, 43
driver APIs, defined, 208. *See also* node-mongodb-native driver; node-mysql driver; node-redis driver

E

ECMA APIs, shimming, 282–283
ECMAScript, 15
ejs template engine, 146–147, 154
elect function, 192
embedded documents, defining in Mongoose, 222
encoded string as parameter of event, 57
end event, 77
end method, 107
ensureIndex function, 216
environmental variables, 65
error event, 34, 47, 77
error handling
  AssertionError in testing, 290
  db.getLastError function, 220
  Express API, 150, 151, 155
  MySQL, 236
  Node, 34–35
  shared-state concurrency, 33–35
  stack traces, 35–37
es5-shim project (website), 282
event loop
  introduction, 1
  IO, relationship to, 32
  keeping it running with Stream, 56
  non-blocking code, relationship to, 30–31
  single thread of execution, 31
EventEmitter API, 45–46, 78, 284

events
  close, 77, 177, 180
  connect, 84, 180, 185–186
  connection, 168, 181
  data, 60–61, 77–79
  disconnect, 180
  encoded string as parameter, 57
  end, 77
  error, 34, 47, 77
  join, 186
  Node JavaScript additions, 45–47
  open, 180
  Socket.IO, 180–181, 185
exception, capturing, 21–22
executing a file, 10
exit API, 65
expect function, 292–293
expect.js project (website), 292–294
exports global object, 41, 44
exports style in Mocha, 298–299
exposing APIs, Node JavaScript additions, 44–45
Express API, 157–158. *See also* routes and route handlers
  convenience methods, 155–157
  error handling, 150, 151, 155
  HTML, 146–147
  introduction, 145
  middleware, 159–160
  module, creating, 146
  MongoDB, 208–212, 213
  node-mysql, 232–233
  organization strategies, 160–162
  search module, 150–151, 152
  sequelize, 245–248
  settings, 153–154
  setup, 147–148
  summary, 162
  template engines, 146, 147, 154, 209
  in WebSocket API, 167

F

file descriptors, 31, 56
filesystem, 92. *See also* fs module
filter method, 23
find method in sequelize, 253–254
findById query operation, 224
findOne command, 216
flag, defined, 186
flow control, 71, 121, 158
FLUSHALL command, 274
forEach method, 23
formidable module, 131
404 (Not Found) status code, 103, 117
framing, defined, 166
fs module
  interacting with, 61–63
  overview, 41
  Stream, 66–68
  sync compared to async style, 54
  watch, 67–68
fs.createReadStream function, 67
fs.stat function, 58, 117
fs.watch function, 68
fs.WriteStream function, 67
function JavaScript type, 16
function name compared to variable name, 18
functions, overview, 18–19. *See also specific functions and methods*

G

-g flag, 13
GET command, 262
Get request, 99, 108, 109–110
GitHub, 3
global object, 40–41
Google Chrome, 1, 22
Grooveshark API (website), 192–196

H

handshake, defined, 166
hash data type in Redis, 263–264
hasMany association, 250
HDEL command, 264
header extension for Request, 155
header extension for Response, 156
headers, HTTP, 89–93, 99–100
HEXLISTS command, 262, 264
HGETALL command, 264, 273, 274
hgetall function, 269
hidden option in static middleware, 127
hmset function, 269
homebrew package manager, 261
HSET command, 263
HTML. *See* browser
HTML5 WebSocket, 166–167
HTTP (Hypertext Transfer Protocol)
  Connect API, compared to, 116–119
  connections, 93–94
  headers, 89–93, 99–100
  introduction, 87
  reloading with up, 111–112
  request+response model inefficiencies, 164–166
  structure, 88–89
  summary, 112
  superagent module, 110–111
  TCP, relationship to, 69, 88, 93
  Twitter web client exercise, 104–110
  web server, creating, 93–104
  WebSocket, relationship to, 167
http.ClientRequest object, 105, 106
http.request object, 105, 106, 108, 109
http.Server object, 69, 182, 183
http.ServerRequest constructor, 88
http.ServerResponse constructor, 88
:http-version dynamic token, 130

I

if node block in browserbuild, 288
index method, 223
index setup for Mongoose, 215–216, 222–223
info object, 241
inheritance, 20–21, 25, 284–285
in-memory store, Redis, 260
input and output, basics in Node, 57–59

insert command, 214–215
instanceof operator, 17, 21
interpolation feature in jade, 240
io.connect function, 184
IP (Internet Protocol), 70
IRC (Internet Relay Chat) client
    program, 83–85
is extension for Request, 156
isarray method, 23
items array, database data,
    239–240, 242–243

J
jade template engine, 154,
    209–210, 238–240
JavaScript. *See also* Node,
    JavaScript additions
  bundling up client-side, 128
  classes, 20
  defined, 15
  functions, 18–19
  inheritance, 20–21
  JSON, compared to, 235
  Redis, compared to, 266
  summary, 26
  try/catch, 21–22
  types, 16–17
  v8, 22–26
JavaScript: The Good Parts
    (Crockford), 15
join event, 186
jQuery listener for sequelize data,
    250–251
JSON
  in Connect, 135
  creating package.json, 12–13
  data format, 104
  encoding specification
    from v8, 24
  JavaScript, compared to, 235
  MongoDB design, relationship
    to, 206
  in sequelize, 252–253
  Socket.IO's coordination of
    events, 181
  in WebSocket, 173, 177
json extension for
    Response, 156
jsonp function, 154

K
keep-alive value for
    Connection header, 94
KEYS command, 261, 262
keys method, 22–23
key-value basis for Redis, 260, 263
Kvalheim, Christian Amor
    (developer), 208

L
lastIndexof method, 24
length property, 19
Linux systems, installing
    Node.JS on, 8–9
list data type in Redis, 265
listen method, 76, 112
locals object, 150
logger middleware, 129–131
login route, MongoDB, 215
login system, creating with Con-
    nect, 134–140
long polling, defined, 180
LPUSH command, 265
LRANGE command, 265

M
Mac OS X, installing Node.JS on, 8
make test command, 9
map method, 23
math libraries, code sharing, 280
maxAge option in static
    middleware, 127
:method dynamic token, 130
methodOverride
    middleware, 141
middleware
  Connect API
    basicAuth, 141–143
    bodyParser, 131–134
    Cookie header, 134
    defined, 115
    introduction, 121–122
    logger, 129–131
    methodOverride, 141
    query, 128
    RedisStore, 140
    session, 134–140
    static, 120, 127–128
    writing reusable, 122–127
  Express API, 159–160
  MongoDB, 208, 214, 217–218
  Mongoose, 223
  next function, relationship
    to, 124, 158
  sequelize, 251
mocha command, 295
Mocha framework, 294–300
Model.count function, 225
models, database
  Mongoose, 219, 220–221,
    223–224, 227–229
  Redis, 268
  sequelize, 249–250
module global object, 41, 44,
    280–281
module system
  assert, 284, 290–291, 293
  code sharing, 280–282
  creating the module, 12–13, 53
  Express API, 146, 150–151
  formidable, 131
  fs, 41, 54, 61–63, 66–68
  HTTP, 95, 104–110
  input and output, 57–59
  Node JavaScript additions,
    41–43
  querystring, 101–102,
    106, 108
  refactoring, 59–61
  streams, 55–57
  superagent, 110–111, 146,
    151, 291–292
  synchronous compared to
    asynchronous code, 53–55
  TCP (chat program), 74
mongo client, 207
mongod server, 207
MongoDB
  compared to other database
    programs, 205–207,
    260, 263
  defined, 205
  installing, 207
  Mongoose, 220–229
  node-mongodb-native
    driver
    application setup, 208
    atomicity, 219
    connecting to, 212–214
    creating documents, 214–215

Express app, creating, 208–212, 213
finding documents, 215–217
middleware, 208, 214, 217–218
safe mode, 219–220
validation, 218–219
summary, 229
mongodb.Server, 212–213
Mongoose
atomicity, 219
automatic key populating, 225
casting, 225–226
defining the model, 220–221
embedded documents, defining, 222
example program, 226–229
index setup, 215–216, 222–223
inspecting the model state, 223–224
limiting, 225
middleware, 223
nested keys, defining, 222
querying, 224–225
selecting, 224
skipping, 225
sorting, 224
monkey-patch (overriding functions), 123
mounting with static middleware, 127
MSI installer in Windows, 8
multi function, 269
multiplexing, defined, 182
MySQL
introduction, 231
MongoDB, compared to, 205, 206
node-mysql driver
connecting to, 234
creating data, 238–242
Express app, 232–233
fetching data, 242–244
initializing the script, 234–238
setup, 232
node-sequelize ORM
connecting to, 248–249
creating data, 250–253
defining models, 249–250
Express app, 245–248
introduction, 244–245
other functionality, 256–257
removing data, 254–256
retrieving data, 253–254
setup, 245
synchronizing to the database, 250
summary, 257

N

name method, 24
namespaces in Socket.IO, 181–182
nested keys, defining in Mongoose, 222
net.connect function, 84
net.createServer, 75
net.Server API, 74–76
net.Stream API, 75, 84
next function and middleware, 124, 158
nextTick function, 40
node command, 2, 9–10
Node Package Manager (NPM), 2, 10–14, 53
node-canvas (canvas 2D context) (website), 284
NODE_ENV, 65, 153
Node.JS. *See also specific APIs and database programs*
code sharing, 279–288
data in memory disadvantage, 266–267
installing, 7–14
introduction, 1–3
JavaScript additions
buffers, 47–48
events, 45–47
exposing APIs, 44–45
global object, 40–41
introduction, 39
module system, 41–43
summary, 48
shared-state concurrency
blocking compared to non-blocking code, 29–31
error handling, 33–35
introduction, 27–29
single thread of execution, 31–33
stack traces, 35–37
state, 79
summary, 37
support resources, 3
testing
expect.js project, 292–294
introduction, 289
Mocha framework, 294–300
program, 291–292
strategy, 290–291
subject for, 290
summary, 300
node_modules directory, 41
node-mongodb-native driver
application setup, 208
atomicity, 219
connecting to, 212–214
creating documents, 214–215
Express app, creating, 208–212, 213
finding documents, 215–217
middleware, 208, 214, 217–218
safe mode, 219–220
validation, 218–219
node-mysql driver
connecting to, 234
creating data, 238–242
Express app, 232–233
fetching data, 242–244
initializing the script, 234–238
sequelize, relationship to, 245
setup, 232
node-redis driver
connecting to redis, 267–268
defining the model, 268
graph methods, 269–270
intersections, computing, 270
testing, 270–276
users, creating and modifying, 268–269
node-sequelize ORM
connecting to, 248–249
creating data, 250–253
defining models, 249–250
Express app, 245–248
introduction, 244–245
other functionality, 256–257
removing data, 254–256
retrieving data, 253–254
setup, 245
summary, 257

non-blocking code, 29–31, 32, 33, 46, 54
nonprinting characters, 66
NoSQL, 206, 231
NPM (Node Package Manager), 2, 10–14, 53
npm install command, 12
npm publish command, 12, 13
null JavaScript type, 16

O

object JavaScript type, 16
ObjectID Schema type, 221, 225, 226
Object.keys API, code sharing issue, 283
Object-Relational Mappers (ORMs), 207. *See also* node-sequelize ORM
octets, 48
ODM (Object Document Mapper), 207
once method, 142
onload handler, 174
OOP frameworks, code sharing, 280
open event, 180
option function, 63
ORMs (Object-Relational Mappers), 207. *See also* node-sequelize ORM

P

package.json file, 12–13
parse method, 24
PATCH request, 99
paused state, 56
persistence, Redis data, 260
PHP compared to Node, 28–31
piping streams, 93
PKG file in Mac OS X, 8
port method, 112
ports package manager, 261
positions object, 172–173
POST request, 46–47, 99, 131
PostgreSQL, 205
primitive JavaScript types, 16
private variable, defined, 19
process object, 40
process.argv API, 63–64, 108
process.cwd object, 64–65
process.env object, 65
process.exit object, 65
process.on handler, 297
programming requirements, setting, 52
progressive API, defined, 111
proto, 25, 284
prototype extension in code sharing, 282–283
prototypical inheritance, defined, 20
PUT request, 99

Q

quality of service (QoS) for TCP, 71
query middleware, 128
query string in URLs, 101
querying database, 215–216, 224–225, 261–262
query.limit extension, 225
query.select extension for MongoDB, 224
query.skip extension for MongoDB, 225
query.sort extension for MongoDB, 224
querystring module, 101–102, 106, 108
querystringparse module, 102

R

RAW TCP mode, 72
Read-Eval-Print Loop (REPL), 2, 9–10
ReadStream filesystem APIs, 92
reception acknowledgment in Socket.IO, 190–191
reconnection issue in WebSocket, 177
redirect extension for Response, 157
Redis
  data types, 262–266
  installing, 261
  introduction, 259–260
  middleware, 140
  node-redis driver, 266–276
  query language, 261–262
  summary, 276
redis-cli command line execution, 261
RedisStore middleware, 140
reduce method, 24
refactoring, 59–61, 226–227
:referrer dynamic token, 130
RegExp object, 157–158
relative modules, 42
:remote-addr dynamic token, 130
removeEventListener API, 45–46
render extension for Response, 156
render method, 149, 150
REPL (Read-Eval-Print Loop), 2, 9–10
:req dynamic token, 130
req.params object, 157
Request object
  Express extensions, 155–156
  HTTP, 88, 93, 104–106, 108, 109–110, 164–166
  Node/JavaScript code sharing, 280, 284
require global object
  browserbuild considerations, 286–287
  exposing APIs, 44–45
  Express, 160
  JSON file loading, 235
  keyword, 14
  module system role, 41
  relative modules, 42–43
:res dynamic token, 130
Response object
  Express extensions, 156–157
  HTTP, 88, 93, 105, 164–166
:response-time dynamic token, 130
REST principles, 245
resume state, 56
review command, 14
routes and route handlers
  case-sensitive, 153
  data deletion route in sequelize, 254–255
  defining in Express, 148–150
  error handling, 155–157

middleware, 159–160
MongoDB, 210–211
organization strategy role (route maps), 160–162
strict, 154
terms defined, 145, 146
web application capabilities, 157–158
RPUSH command, 265

S

SADD command, 266
safe option, MongoDB driver, 219–220
Schema class, 220
schema-less database, MongoDB as, 205–206
scope definition in JavaScript, 19
<script> tag, 183–184
search command, 14
search module for Express Twitter app, 150–151, 152
Select command, 242–243
self-invoked function, defined, 19
semver versioning spec, 53
send extension for Response, 156
send object, 110–111, 166
sendfile extension for Response, 157
sequelize. *See* node-sequelize ORM
Sequelize constructor, 248–249
sequelize.define, 249
sequelize.sync, 250
serial execution, 58, 59
serve function, 118
serve(1) utility, 300
server. *See also* createServer function; HTTP
communications with client, 70
creating with Node.JS, 1–2, 10
MongoDB, 207, 212–213
net global object, 74–76
Server API, 69, 95–104, 182, 183
session data, 140, 266–267
session middleware, 134–140, 208
SET command, 262
set data type in Redis, 265–266
set object, 110–111
setEncoding method, 105
setImmediate API, 40
setTimeout function, 40
shared-state concurrency
blocking compared to non-blocking code, 29–31
error handling, 33–35
introduction, 27–29
single thread of execution, 31–33
stack traces, 35–37
state, 79
summary, 37
shimming of APIs, 280, 282–284
short logging format, 129
SIGKILL signal, 65
signals, 65–66
single thread of execution, 31–33
sio.listen function, 183
slash-r, slash-n (\r\n) delimiter, 85
SMEMBERS command, 262, 266
socket, TCP, 73
Socket.IO
chat program exercise
broadcasting, 185–190
client setup, 183–185
events, 185
reception acknowledgment, 190–191
server setup, 182–183
DJ-by-turns application exercise
extending chat, 191–192
Grooveshark API integration, 192–196
playing function, 196–201
introduction, 179
summary, 201
transports, 180–182
socket.io, program setup, 182
sorted sets in Redis, 266
sorting in database, 224
square brackets ([]) for uploading multiple files, 134
SREM command, 266
Stack Overflow (website), 3
stack trace, 24–25, 35–37
Stat object, 58
state variables and TCP, 79–81
static middleware, 120, 127–128
:status dynamic token, 130
stderr stream object, 56, 57
stdin stream object, 56, 59–60
stdio process, 51
stdout stream object, 51, 56, 57, 59–60
store for login sessions, setting up with Connect, 140
Stream objects
in first program module, 55–57
fs module, 66–68
net.createServer, 75–76
net.Stream API, 75, 84
types of, 51, 56–57, 59–60
strict routing in Express, 154
string data type in Redis, 263
string JavaScript type, 16
string methods in v8, 24
stringify method, 24, 106
String.prototype, 42
suite in TDD, 298
superagent module, 110–111, 146, 151, 291–292
synchronizing to database in sequelize, 250
synchronous code, 30, 53–55

T

tablescan lookup, defined, 216
TCP (Transmission Control Protocol)
characteristics of, 69–71
chat program exercise
creating the module, 74
data event, 77–79
disconnecting, 81–83
net.Server API, 74–76
receiving connections, 76–77
state variables, 79–81
HTTP, relationship to, 69, 88, 93
introduction, 69
IRC client program, 83–85
summary, 85
Telnet, 71–73
TDD (test-driven development), 298
telnet client, 71–73
template engines
code sharing, 280
ejs, 146–147, 154
jade, 154, 209–210, 238–240

tenants in cloud environment, 33
testing
  arrays, 23
  asynchronous code, 295–297
  expect.js project, 292–294
  introduction, 289
  Mocha framework, 294–300
  node-redis driver, 270–276
  program, 291–292
  strategy, 290–291
  subject for, 290
  summary, 300
test.js (Mocha), 295–300
text editor, 2
third-party modules, 41
time complexity, Redis, 262
timeouts
  setTimeout function, 40
  Socket.IO, 180
  TCP, 71
tiny logging format, 129
TinySong API, 193–196
Transfer-Encoding HTTP header, 92
Transmission Control Protocol. *See* TCP (Transmission Control Protocol)
transports, Socket.IO, 180–182
try/catch in JavaScript, 21–22
type options in Mongoose, 221
typeof operator, 17, 281

U

Ubuntu, 9
uncaughtException handler, 34
undefined JavaScript type, 16
Unix Standard Streams, 56
up executable, 111–112
updateAttributes method, 255
uploads, file, with bodyParser, 131–134
:url dynamic token, 130
url property, 97
/url property, 98
use function for static middleware, 120
:user-agent dynamic token, 130
utf8 string, 105

V

v8 JavaScript interpreter, 1, 17, 22–26, 32, 33
validate option in sequelize, 256
validation, MongoDB, 218–219
versioning spec for NPM, 53
view options parameter in Express, 147, 209, 232
vim text editor (website), 2
virtualized operating systems in cloud environment, 33

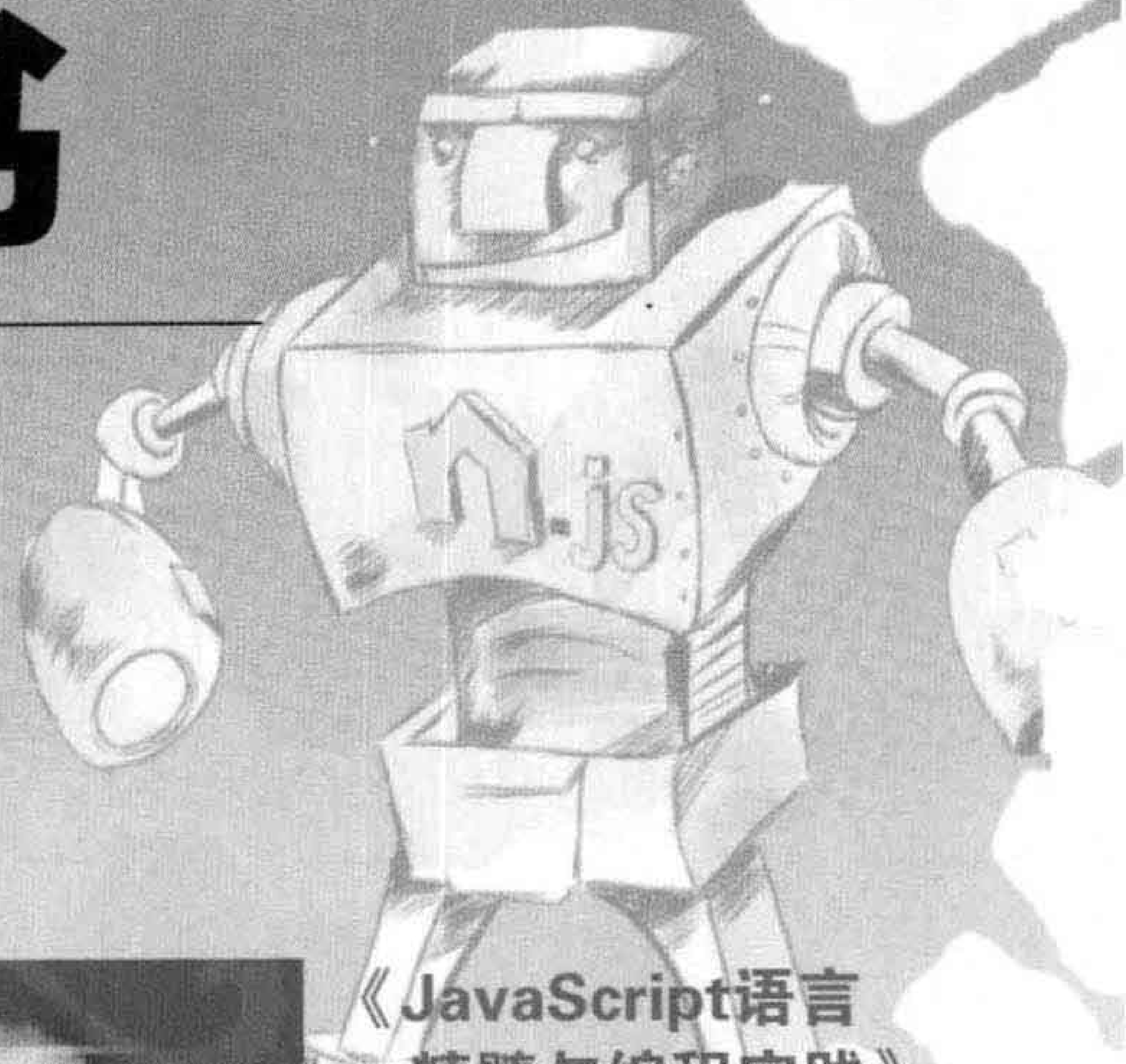
W

W3C, 166
watch function, 67–68, 112
Web 2.0, 164
web applications, rise of, 164
web browser. *See* browser
web server. *See* server
WebSocket. *See also* Socket.IO
  API compared to Protocol, 166
  background for, 163–167
  broadcasting issue, 177, 185–190
  client setups, 169
  close event and disconnecting, 177
  defined, 166
  echo example, 167–171
  JSON packet encoding/decoding, 173, 177
  mouse cursor example, 171–176
  Node/JavaScript code sharing, 280, 284
  reconnection issue, 177
  running the server, 170, 176
  server setups, 168, 172
  summary, 178
websocket.io, 167–168, 172
White, Nathan (developer), 207
window object, 40
Windows systems, installing Node.JS on, 8
working directory, CLI, 64–65
writeHead API, 90, 92

X

XCode, 8
XMLHttpRequest API, 280, 284

JS精品图书



JavaScript Web Applications
基于MVC的
JavaScript Web
富应用开发
O'REILLY

JavaScript
语言精髓与编程实践
（第2版）

JavaScript: The Good Parts
JavaScript
语言精粹
（修订版）
O'REILLY

AngularJS
用AngularJS开发
下一代Web应用
O'REILLY

Node.js是一个由JavaScript书写而成的强大的Web开发框架，它让开发强壮的、伸缩性良好的服务器端Web应用变得更加简单、容易。本书向你展示了什么是Node以及如何让你在项目中使用它。本书包含大量实际应用中的示例程序，证明了为什么Node.js会快速成为Web开发首选工具，通过本书，你能够快速熟悉和掌握达到如下目标所需的Node知识和技能：

- 了解Node基于事件轮询的架构、无阻塞I/O以及事件驱动的编程方式
- 精通Node.js的API
- 轻松实现开发实时应用相关的技术，如Socket.IO和HTML5 WebSocket
- 编写能够支持跨多台服务器的高并发应用
- 通过Node来支持多种数据库以及数据存储工具
- 编写在单台服务器情况下能够处理万级并发量的程序
- 能够在一个包含更多Node知识和注解示例（含源代码）的网站上和其他开发者进行实时的沟通交流

本书包含大量全彩插图和实用的源代码，绝对是一本革命性Web开发工具——Node的实用指南。

Guillermo Rauch（旧金山，加利福尼亚州）是一家位于旧金山，为当地教育提供相关服务的创业公司LearnBoost的CTO和联合创始人。Rauch还是几个知名Node.js项目的发明者，曾在JSConf和一些Node.js workshop做过演讲。

WILEY

策划编辑：张春雨
责任编辑：贾　莉
封面设计：侯士卿

上架建议：Web开发/JavaScript

ISBN 978-7-121-21769-2

定价：79.00元