INDEX

SYMBOLS
`
(backtick), 23  
$ (dollar sign), 23  
⇒ (fat arrow), 21  
! (exclamation mark), 102  
. (period), 176  
+ (plus operator), 34, 142  
? (question mark), 45  
... (spread operator), 27–28, 78  
[] (square brackets), 77, 102  
_ (underscore), 105  

NUMBERS
200 status code, 107, 248–249  
404 status code, 227  
405 status code, 111  
500 status code, 77, 101, 249  

A  
absolute imports, 196  
abstract syntax tree (AST), 103–104  
access token, 159  
using authorization grant to 
get, 171  
using to get protected resource, 172  
act, test cases, 133  
allowJs option, 259  
AMD format, 16  
anonymous functions, 16–17  
any type, 43  
APIs (application programming 
interfaces), 57  
containers communicating 
through, 174  
contracts, 34, 38, 94  
GraphQL APIs, 101–113  
microservices communicating 
through, 178  
REST APIs, 93–101  

routes  
creating, 90  
for GraphQL API, 110–111  
overview, 75–77  
replacing with route handlers, 
285–287  
Apollo sandbox, 111–113  
Apollo server, 108  
app directory, 72, 263–287  
exploring project structure, 269–287  
adding content and route, 
275–277  
adding server component 
that fetches remote data, 
281–284  
catching errors, 277–279  
completing application with 
navigation, 284–285  
defining layout, 273–275  
replacing API routes with 
route handlers, 285–287  
showing optional loading 
interface, 279–281  
updating CSS, 271–272  
rendering components  
dynamic rendering, 268–269  
fetching data, 266–267  
static rendering, 267–268  
server components vs. client 
components  
client components, 265  
server components, 264–265  
App function, 56  
application glue, 120  
application programming interfaces. 
See APIs  
apps  
serving from Docker container, 177  
using databases and object-relational 
mappers, 116
arranging test cases, 132–133
array.map function, 27
arrays
   dispersing, 27–28
   identifying object types as, 102
   looping through, 27
array type, 41–42
arrow functions, 20–22
   exploring practical use cases, 22
   lexical scope, 21–22
   writing, 21
assertion, test cases, 133–134
AST (abstract syntax tree), 103–104
asynchronous scripts
   avoiding traditional callbacks, 24–25
   simplifying, 26–27
   using promises, 25–26
   writing ES.Next module with, 29–30
async keyword, 26–27
audience claim, 164
auditing package.json file, 10
AuthElement component
   adding to header, 241–243
   overview, 238–240
authentication
   authorization vs., 158–159
   REST APIs, 97–98
authentication callback, 233–236
auth guard, 248–249
authorization, 157–172
   accessing protected resource, 168–172
   logging in to receive
      authorization grant, 170–171
   setting up client, 168–170
   using access token to get
      protected resource, 172
   using authorization grant to get access token, 171
authentication vs., 158–159
bearer tokens, 160–161
code flow, 160–163
creating JWT tokens, 163–168
   header, 163
   payload, 163–166
   signature, 166–168
grant types, 159–160
   role of OAuth, 159
authorization code flow, 160–163
authorization grant
   logging in to receive, 170–171
   using to get access token, 171
authorization server, 159
automated tests, 253–257
   adding Jest to project, 254
   setting up, 254–256
   writing snapshot tests for header
      element, 256–257
await keyword, 26–27

B
Babel.js, 15
backend container
   creating backend service, 187–189
   seeding the database, 186–187
backtick (’), 23
baseUrl option, 259
bearer tokens, 160–161
beforeAll hook, 132
beforeEach hook, 132
black-box test, 144
blocking time, 86
block scope, 17
Booleans, 40
Boolean scalar type, 102
built-in components
   next/head, 80–81
   next/image, 82–83
   next/Link, 81–82
built-in hooks
   handling side effects with
      useEffect, 62–63
   managing internal state with
      useState, 62
   sharing global data with
      useContext and context
      providers, 63–64
built-in matchers, 289–291
built-in types
   any, 43
   array, 41–42
   object, 42
   primitive types, 40–41
tuple, 42–43
union, 41
void, 43–44
built-in validators, 118
button, generic, 235–238, 244–247

C
@cacheControl directive, 103
cacheControl.setCacheHint resolver function, 103
cached connection, 198/callback hell, 25
callbacks
  array function running, 138
  array.map function, 27
  arrow functions simplifying, 22
  avoiding traditional, 24–25
callback URL, 162
cases, test, 130
catch all API route, 78
catch method, 25–26
claims, 163–166
  private, 166
  public, 165
  registered, 164–165
class components, 59–60
client components, 265
client credentials, 259
  flow, 160
client ID, 159
clients, 159
client secret, 159
client-side rendering, 88–89
cloning arrays and objects, 28
code coverage, 130, 138–139
code generator, 55
collections, 117
collisions, 161
compilerOptions field, 37
compilers, 36
components, 57–61
  Next.js built-in components, 80–83
  next/head, 80–81
  next/image, 82–83
  next/link, 81–82
  providing reusable behavior with hooks, 61
  styles for, 79–80
  writing class components, 59–60
concise body function, 21
constant-like data, 20
const keyword, 20
constructor function, 59
container class, 80
containerization, 173–182. See also Docker
collection providers, 63–64
COPY keyword, 176
create-next-app command, 70
create-react-app command, 55
Cross-Origin Resource Sharing (CORS), 75–76
CRUD operations, 121–123, 199
CSS styles, 78–80
  adding to list item, 216–217
  component styles, 79–217
  global styles, 79
  updating, 271–272
cumulative layout shifts, 82
curl command, 248, 251–252
cURL tool, 99
custom types, 44–45, 208
Cypress, 253
D
daemon service, 175
database-connection middleware, 120–121
data mapping, 77
data types, 20
declarative programming, 54
declaring variables, 17–20
  constant-like data, 20
  hoisted variables, 18–19
  scope-abiding variables, 19
default exports, 16–17
default keyword, 16
DefinitelyTyped repository, 46
DELETE method, 98
deleteOne function, 123
deleting document, 123
dependencies
  installing, 8–9
  overview, 6
  removing, 11
  replacing, 139–143
  creating doubles folder, 141
dependencies (continued)
creating (continued)
creating module with
dependencies, 140–141
using fakes, 142
using mocks, 143
using stubs, 142
useEffect hook managing, 63
details component, 227–228
development dependencies
installing, 9–10
overview, 6
development scripts, Next.js, 72
directives, 208
dispersing arrays and objects, 27–28
Docker, 173–182, 185–193
building local environment with
backend container, 186–189
frontend container, 189–192
containerization architecture, 174
containers, 174–178
building Docker image, 176
interacting with, 178
locating exposed Docker port, 177–178
serving application from, 177
writing Dockerfile, 175–176
Docker Compose, 178–182
interacting with, 182
rerunning tests, 181–182
running containers, 180–181
writing docker-compose.yml file, 179–180
Food Finder application, 186
installing, 174
running automated tests in,
253–257
adding Jest to project, 254
setting up, 254–256
writing snapshot tests for
telement constant, 57
dynamically typed languages, 34
dynamic feedback, 38
dynamic rendering, 268–269
dynamic URLs, 77–78
E
element constant, 57
elements, 56
encrypted tokens, 161
endpoint, 95
dynamic rendering, 268–269
dynamic URLs, 77–78

end-to-end query, 123–125
dynamic rendering, 268–269
environment problems, 144
element constant, 57
elements, 56
catching, 277–279
with const keyword, 20
Internal Server Error, 77
non-hoisted variables, 19
promises and, 25–26
TypeScript, 36
using variables before declaring, 18
escape character, 99
esModuleInterop option, 259
ES.Next modules, 15–17
importing modules, 17
using named and default exports,
16–17
writing with asynchronous code,
29–30
exclamation mark (!), 102
exclude option, 37
executing script, using npx, 12
expect function, 133–134
expiration claim, 164
export statement, 16
exported Docker port, 177–178
Express.js Fundamentals course, 14
Express.js server
building “Hello World,” 13–14
creating reactive user interface for,
64–67
extending with modern JavaScript,
29–31
extending with TypeScript, 46–51
adding type annotations to
index.ts file, 49–50
adding type annotations to
routes.ts file, 48–49

document databases, 117
document object model (DOM), 54, 57
documents, 117
dollar sign ($), 23
domain-specific language (DSL), 24
doubles folder, 141
creating `tsconfig.json` file, 47
defining custom types, 47–48
setting up, 46–47
transpiling and running code, 50–51
refactoring, 89–91
extends option, 37
external APIs, 93–94

F
fakes, 142
fat arrow (=>), 21
fetch API, 26–27, 266–267, 281–284
Fibonacci sequence, 140–143
fields, 117
filter method, 22
finally method, 25–26
findOne function, 122
float scalar type, 102
Food Finder application, 186
forceConsistentCasingInFileNames option, 260
--force flag, 10
FROM keyword, 175
frontend container
  application service
    adjusting for restarts, 191–192
    creating, 189–190
global layout components, 222–226
  header, 223–224
  layout, 224–226
  logo, 222–223
installing Next.js, 190–191
location details page, 227–230
start page, 216–222
  list item component, 216–218
  location list component, 218–219
  user interface, 215–216
fs module, 24–25
functional tests, 144
function components, 59
functions
  arrow, 20–22
    exploring practical use cases, 22
    lexical scope, 21–22
    writing, 21
  avoiding traditional callbacks, 24–25
  type annotations declaring parameters of, 39–40
  function scope, 17–18

G
gateway communications, 144–145
generic button component, 235–238, 244–247
getByTestId matcher, 292
GET method, 98–100
getServerSideProps function, 84–85
getToken function, 250
GitHub OAuth app, 232
global data, sharing with useContext hook, 63–64
global layout components, 222–226
  header, 223–224
  layout, 224–226
  logo, 222–223
global scope, 18, 44
global styles, 79, 219–220
Google Authenticator, 158
Google scoring algorithm, 86
gql tag, 210
grant types, 159–160
graph databases, 117
GraphQL APIs, 75, 101–113, 207–214
  adding API endpoint to Next.js, 212–214
  adding to Next.js
    adding data, 109
    creating API route, 110–111
    creating schema, 108–109
    implementing resolvers, 109–110
    using Apollo sandbox, 111–113
  comparing REST to
    over-fetching, 106–107
    under-fetching, 107–108
  connecting MongoDB to
    adding services to GraphQL resolvers, 126–127
    connecting to database, 125–126
  merging typedefs into final schema, 209–210
GraphQL APIs (continued)
resolvers, 103–106, 210–212
schemas, 101–103
custom types and directives, 208
mutation schema, 209
query schema, 209
securing mutations, 247–252
setting up, 208
GraphQL queries, 209
GraphQL schema, 24
guards, 248

H
hash-based message authentication code (HMAC), 161
Head elements, 80–81
header
adding AuthElement component to, 241–243
global layout components, 223–224
JWT tokens, 163
writing snapshot tests for, 256–257
hoisted variables, 18–19
hooks, 62–64
handling side effects with useEffect, 62–63
managing internal state with useState, 62
providing reusable behavior with, 61
sharing global data with useContext and context providers, 63–64
host system, 174–175
hot-code reloading, 72
HTML, 24
incremental static regeneration, 87
JSX elements and, 57
reactive user interface and, 54
static HTML exporting, 89
HTTP methods, 98–99

I
id helper program, 192
ID scalar type, 102
Image component, 82–83
images, Docker, 176
<img> element, 82
immutable data types, 20
immutable elements, 57
implicit flow, 160
importing modules, 17
import statement, 16–17
include option, 37
incremental option, 260
incremental static regeneration (ISR), 87
integration tests, 144–145
interaction-based tests, 132
interface keyword, 45
interfaces
defining, 45
Mongoose model, 118
storing, 90
inter-module communication, 144
internal APIs, 93
Internal Server Error, 77, 101
internal state, managing with useState hook, 62
Int scalar type, 102
I/O operations, 24–25
isolatedModules option, 260
ISR (incremental static regeneration), 87
issued at claim, 165
issuer claim, 164

J
JavaScript
arrow functions, 20–22
exploring practical use cases, 22
lexical scope, 21–22
writing, 21
asynchronous scripts
avoiding traditional callbacks, 24–25
simplifying, 26–27
using promises, 25–26
creating strings, 22–24
declaring variables, 17–20
dispersing arrays and objects, 27–28
ES.Next modules, 15–17, 29–30
Express.js server
building “Hello World,” 13–14
extending, 29–31
looping through arrays, 27
Node.js, 3–14
creating projects, 8–12
installing, 4
package.json file, 4–6
package-lock.json file, 6–7
working with npm, 4
TypeScript, 33–51
benefits of, 34–36
built-in types, 40–44
custom types and interfaces, 44–46
extending Express.js server with, 46–51
setting up, 36–38
type annotations, 38–40
JavaScript Syntax Extension (JSX)
example expression, 56–57
ReactDOM package, 57
JEST-DOM matchers, 292–293
Jest framework, 129–156
adding test cases to weather app creating mocks to test services, 148–151
evaluating user interface with snapshot test, 153–156
performing end-to-end test of REST API, 151–153
testing middleware with spies, 146–148
adding to project, 254
anatomy of test case act, 133
arrange, 132–133
assertion, 133–134
creating example module to test, 131–132
matchers
built-in, 289–291
JEST-DOM, 292–293
replacing dependencies, 139–143
creating doubles folder, 141
creating module with dependencies, 140–141
using fakes, 142
using mocks, 143
using stubs, 142
setting up, 130–131
test-driven development, 135–139
evaluating test coverage, 138–139
overview, 130
refactoring code, 136–138
types of tests
end-to-end tests, 145
functional tests, 144
integration tests, 144–145
snapshot tests, 145
unit testing, 130
jsonlint package, 12
JSX. See JavaScript Syntax Extension
jsx option, 260
JWT (JSON Web Token)
defined, 160–161
header, 163
payload, 163–166
private claims, 166
public claims, 165
registered claims, 164–165
signature, 166–168
JWT claim, 165
K
key-value storage, 117
kill command, 178
L
layout
app directory, 273–275
global layout components, 224–226
let keyword, 19
lexical scope, 21–22
Lib option, 260
lifecycle methods, 59
Link component, 81–82
list item component, 216–218
loading user interface, 279–281
local environment
backend container, 186–189
creating backend service, 187–189
seeding the database, 186–187
local environment (continued)

frontend container, 189–192
adjusting application service for restarts, 191–192
creating application service, 189–190
installing Next.js, 190–191
location details page, 215
adding button to, 244–247
overview, 227–230
location ID, 215
location list component, 218–219
location services
creating, 203–205
custom types for, 203
logo, 222–223
long-term support (LTS) version, 4
looping through arrays, 27

M

MAC (message authentication code), 161
major version changes, 5
matcher function, 134
matchers
built-in, 289–291
JEST-DOM, 292–293
Memcached, 117
message authentication code (MAC), 161
microservices, 178–182
interacting with Docker Compose, 182
rerunning tests, 181–182
running containers, 180–181
writing docker-compose.yml file, 179–180
middleware, 120–121, 195–206
configuring Next.js to use absolute imports, 196
connecting Mongoose, 196–199
fixing TypeScript warning, 198–199
writing database connection, 197–198
creating Mongoose model
creating location model, 201–202
creating schema, 199–200
model services, 202–206
creating location services, 203–205
testing, 206
testing with spies, 146–148
minor version changes, 5
mobile-first design pattern, 222
mocks, 143, 148–151
module option, 260
moduleResolution option, 260
module scope, 18, 44
MongoDB, 101, 115–128
connecting GraphQL API to database, 125–126
adding services to GraphQL resolvers, 126–127
creating end-to-end query, 123–125
defining Mongoose model
database-connection middleware, 120–121
interfaces, 118
model, 119–120
schema, 118–119
how apps use databases and object-relational mappers, 116
querying database
creating database, 121–122
deleting database, 123
reading database, 122
updating database, 122–123
relational and non-relational databases, 116–117
setting up Mongoose and, 117
MongoDB Query Language (MQL), 117
Mongoose
connecting middleware, 196–199
fixing the TypeScript warning, 198–199
writing database connection, 197–198
creating model
creating location model, 201–202
creating schema, 199–200
defining model
database-connection middleware, 120–121
interfaces, 118
model, 119–120
schema, 118–119
setting up, 117
MQL (MongoDB Query Language), 117
multifactor authentication, 158
mutations, 101, 211–212
defining schema, 209
input type object for, 102–103
securing GraphQL, 247–252
MySQL, 101

N
named exports, 16–17
name field, 5
--name flag, 177
navigation, 284–285
Neo4j, 117
nested page routes, 73–75
networking protocols, 8
next-auth, 231–235
adding client credentials, 232
creating authentication callback, 233–236
creating GitHub OAuth app, 232
installing, 233
sharing session across pages and components, 235
next export command, 89
next/head component, 80–81
next/image component, 82–83
Next.js, 13, 69–91
adding API endpoint to, 212–214
adding GraphQL API to, 108–113
adding data, 109
creating API route, 110–111
creating schema, 108–109
implementing resolvers, 109–110
using Apollo sandbox, 111–113
app directory, 263–287
exploring project structure, 269–287
rendering components, 266–269
server components vs. client components, 264–265
built-in components
next/head, 80–81
next/image, 82–83
next/link, 81–82
configuring to use absolute imports, 196
installing in container, 190–191
pre-rendering and publishing, 83–89
client-side rendering, 88–89
incremental static regeneration, 87
server-side rendering, 84–85
static HTML exporting, 89
static site generation, 86–87
refactoring React and Express.js applications, 89–91
routing applications, 72–78
API routes, 75–77
dynamic URLs, 77–78
nested page routes, 73–75
simple page routes, 73
setting up, 70–72
development scripts, 72
project structure, 71–72
styling applications, 78–80
component styles, 79–80
global styles, 79
wish list page, 243–244
next/link component, 81–82
Node.js, 3–14
creating projects, 8–12
auditing package.json file, 10
cleaning up node_modules folder, 11
executing script only once using npx, 12
initializing new module or project, 8
installing dependencies, 8–9, 11–12
installing development dependencies, 9–10
removing dependencies, 11
updating all packages, 11
Express.js-based Node.js server, 13–14
installing, 4
package.json file, 4–6
dependencies, 6
Node.js (continued)

package.json file (continued)
  development dependencies, 6
  required fields, 5
package-lock.json file, 6–7
  TypeScript installation in, 36–37
  working with npm, 4

node_modules folder
  cleaning up, 11
package.json file vs., 4–5
node package execute (npx) tool, 12
noEmit option, 260
non-hoisted variables, 19–20
non-nullable fields, 102
non-primitive data types, 20
non-relational databases, 116–117
NoSQL databases, 117
not before claim, 165
npm, 4
  npm audit command, 10
  npm init command, 8
  npm install command, 7, 11–12
  npm prune command, 11
  npm run build command, 72
  npm test command, 131
  npm uninstall command, 11
  npm update command, 11
npx command, 70
npx next build command, 72
npx tool, 12
null types, 40–41
numbers, as primitive types, 40

O

OAuth, 157–172
  accessing protected resource
  logging in to receive
    authorization grant, 170–171
  setting up client, 168–170
  using access token to get
    protected resource, 172
  using authorization grant to
    get access token, 171
  adding button to location detail
    component, 244–247
  adding with next-auth, 231–235
  adding client credentials, 232
  creating authentication
callback, 233–236
  creating GitHub OAuth
    app, 232
  installing next-auth, 233
  sharing session across
    pages and components, 235

AuthElement component
  adding to header, 241–243
  overview, 238–240
  authentication vs., 158–159
  authorization code flow,
    161–163
  bearer tokens, 160–161
  creating JWT tokens
    header, 163
    payload, 163–166
    signature, 166–168
  generic button component,
    235–238
  grant types, 159–160
  role of OAuth, 159
  securing GraphQL mutations,
    247–252
  wish list Next.js page, 243–244
  object data modeling, 116
  object-relational mappers, 116
  objects, 27–28
  object type, 42
  one-time password (OTP), 158
  online playground, 37, 55
  online registry, npm, 4
  OpenAPI format, 95
  over-fetching, 106–107
  over-typing, 38–39

P

package.json file, 4–6
  auditing, 10
  dependencies, 6
  development dependencies, 6
  editing, 29
  required fields, 5
package-lock.json file, 6–7
packages
  npm online registry, 4
  updating, 11
page routes
  adding, 275–277
  creating, 90–91
  nested, 73–75
  simple, 73
pages folder, 71–72
parameters of functions, 39–40
PATCH method, 98
patch version changes, 5
$PATH environment variable, 12
payload, JWT tokens, 163–166
  private claims, 166
  public claims, 165
  registered claims, 164–165
period (.), 176
persisting the data, 116
Playwright, 253
plus operator (+), 34, 142
POST method, 98
prefixes, 79–80
pre-rendering, 83–89
  client-side rendering, 88–89
  incremental static regeneration, 87
  server-side rendering, 84–85
  static HTML exporting, 89
  static site generation, 86–87
primitive types, 20, 40–41
private APIs, 93
private claims, 166
profile pages, 93
promise chain, 26
Promise object, 25
props argument, 57–58, 84, 86
protected resource
  logging in to receive authorization grant, 170–171
  setting up client, 168–170
  using access token to get protected resource, 172
  using authorization grant to get access token, 171
providers, 231
public claims, 165
public folder, 71
--publish-all flag, 177
push method, 20
PUT method, 98
Q
queries, 101
querying database, 121–123
query schema, 209
question mark (?), 45
R
ReactDOM package, 57
reactive user interface, 54
React, 53–67
  creating reactive user interface for Express.js server, 64–67
  JavaScript Syntax Extension, 56–57
  organizing code into components, 57–61
  providing reusable behavior with hooks, 61
  writing class components, 59–60
  refactoring, 89–91
  role of, 53–55
  setting up, 55–56
  working with built-in hooks
    handling side effects with useEffect, 62–63
    managing internal state with useState, 62
    sharing global data with useContext and context providers, 63–64
reading
  data, 99–100
  document, 122
  files, 25
Redis, 117
refactoring code, 136–138
refresh token, 160
registered claims, 164–165
relational databases, 116–117
remote data, fetching, 281–284
rendering components
  dynamic rendering, 268–269
  fetching data, 266–267
  static rendering, 267–268
replacing dependencies, 139–143
  creating doubles folder, 141
replacing dependencies *(continued)*
creating module with
  dependencies, 140–141
using fakes, 142
using mocks, 143
using stubs, 142
replay attack, 165
report, test-coverage, 138–139
require statement, 16
resolveJsonModule option, 260
resolvers, 210–212
  implementing, 109–110
  overview, 103–106
resource owner, 159–160
resource providers, 159
REST APIs, 75, 93–101, 212
  comparing GraphQL to
    over-fetching, 106–107
    under-fetching, 107–108
  creating end-to-end query, 123–125
  HTTP methods, 98–99
  overview, 94–95
  performing end-to-end test of,
    151–153
  specification, 95–97
  state and authentication, 97–98
URLs, 95
working with
  reading data, 99–100
  updating data, 100–101
restarts, 191–192
RESTful APIs, 159
return value, type annotations
  declaring, 39
reusable behavior, 61
root entry point, 95
root privileges, 192
route handlers, 285–287
routing applications, 72–78
  API routes, 75–77
  dynamic URLs, 77–78
  nested page routes, 73–75
  simple page routes, 73

S
--save-dev flag, 36–37, 46
scaffolding process, 55
scalar types, 102
Schema Definition Language (SDL), 101
  schemas
    GraphQL APIs, 101–103, 108–109
    custom types and directives, 208
    merging typedefs into final
      schema, 209–210
    mutation schema, 209
    query schema, 209
  Mongoose, 118–119, 199–200
scope, variable, 17
scope-abiding variables, 19
scope property, 22
screenshots, 145
script, executing once using npx, 12
SDL (Schema Definition Language), 101
secrets, 233
securing GraphQL mutations, 247–252
seeding the database, 124, 186–187
semantic versioning, 5–6
SEO metadata, 80, 86
server components, 264–265
server-side rendering (SSR), 84–85
services, 121
session information, 97–98
sessions, sharing across pages and
  components, 235
SHA-256 hash algorithm, 161
side effects, 62–63
SIGKILL command, 182
signature, JWT tokens, 166–168
signed tokens, 161
SIGTERM command, 182
single-factor authentication, 158
skipLibCheck option, 260
snapshot tests, 145
  evaluating user interface with,
    153–156
  writing for header, 256–257
specification, 95–97
spies, 146–148
spread operator (…), 27–28, 78
SQL (Structured Query Language),
  116–117
square brackets ([[]), 77, 102
SSG (static site generation), 86–87, 215
SSR (server-side rendering), 84–85
SSR (static site rendering), 216, 244
start page, 216–222
  list item component, 216–218
  location list component, 218–219
state-based tests, 132
stateless, REST APIs, 97–98
statically typed languages, 36
static exports
  API routes and, 76
  HTML, 89
static HTML file, 66
static rendering, 267–268
static site generation (SSG), 86–87, 215
static site rendering (SSR), 216, 244
steps, test, 130
sticky header, 222–223
strings
  benefits of TypeScript, 34
  creating, 22–24
  as primitive types, 40
  template literals for, 22–24
String scalar type, 102
Structured Query Language (SQL), 116–117
stubs, 142
styles folder, 71–72
styles object, 80
styling applications, 78–80
  component styles, 79–80
  global styles, 79
subject claim, 164
suites, test, 130
sum function, 131–132, 135
super function, 59–60
Swagger, 95–97
testing, 129–156
  adding test cases to weather app
    creating mocks to test the services, 148–151
    evaluating user interface with snapshot test, 153–156
    performing end-to-end test of REST API, 151–153
    testing middleware with spies, 146–148
  anatomy of test case
    act, 133
    arrange, 132–133
    assertion, 133–134
  creating example module to test, 131–132
  model services, 206
  replacing dependencies, 139–143
  creating doubles folder, 141
  creating module with dependencies, 140–141
  using fakes, 142
  using mocks, 143
  using stubs, 142
rerunning, 181–182
running automated tests in Docker, 253–257
  adding Jest to project, 254
  setting up, 254–256
  writing snapshot tests for header element, 256–257
setting up, 130–131
  test-driven development, 135–139
  evaluating test coverage, 138–139
  overview, 130
  refactoring code, 136–138
types of
  end-to-end tests, 145
  functional tests, 144
  integration tests, 144–145
  snapshot tests, 145
unit testing, 130
testing-library/dom assert package, 134
testing-library/react assert package, 134
test runner, 130
testWatch command, 254–256
then method, 25–26
third-party APIs, 93–94
this keyword, 21–22, 59
time to first paint, 86
toBeCloseTo matcher, 290
toBeGreaterThan/toBeGreaterThanOrEqual matcher, 290–291
toBeInTheDocument matcher, 292
toBeLessThan/toBeLessThanOrEqual matcher, 290–291
toBe matcher, 289–290
toBeTruthy/toBeFalsy matcher, 291
toContainElement matcher, 293
toContain matcher, 291
toEqual matcher, 290
toHaveAttribute matcher, 293
toHaveClass matcher, 293
toMatch matcher, 291
toStrictEqual matcher, 290
toThrow matcher, 291
transpilers, 36
TSC. See TypeScript Compiler
tsconfig.json file, 37–38
tuple type, 42–43
type annotations, 38–40
declaring parameters of functions, 39–40
declaring return value, 39
declaring variables, 39
type declaration files, 45–46
typedefs, 101, 209–210
type keyword, 44–45
TypeScript, 16, 33–51
benefits of, 34–36
built-in types
any, 43
array, 41–42
object, 42
primitive types, 40–41
tuple, 42–43
union, 41
void, 43–44
custom types and interfaces
defining custom types, 44–45
defining interfaces, 45
using type declaration files, 45–46
extending Express.js server with, 46–51
setting up
dynamic feedback, 38
installation in Node.js, 36–37
tsconfig.json file, 37–38
type annotations, 38–40
declaring parameters of functions, 39–40
declaring return value, 39
declaring variables, 39
using JSX with, 57
TypeScript Compiler (TSC), 36
fixing warning, 198–199
options, 259–261
@types scope, 46
U
UMD format, 16
undefined type, 40–41
under-fetching, 107–108
underscore (_), 105
union type, 41
unit testing, 130
untagged template literal, 22–23
updateOne function, 122–123
useContext hook, 63–64
useEffect hook, 61–63, 89, 244
user ID, 215, 244
user interfaces
evaluating with snapshot tests, 153–156
frontend container, 215–216
showing optional loading user interface, 279–281
user property, 192
useSession hook, 240
useState hook, 61–62, 89
V
-v (version) flag, 4
variables, 17–20
constant-like data, 20
hoisted variables, 18–19
scope-abiding variables, 19
type annotations declaring, 39
var keyword, 18–19
version field, 5
versioning APIs, 95
viewport, 81
virtual DOM, 54
Visual Studio Code, 38
void type, 43–44
--volume flag, 177
volumes, 177
vulnerabilities, 10

W
W3Schools tutorials, 14, 67
weather app
  creating mocks to test the services, 148–151
  evaluating user interface with
    snapshot test, 153–156
performing end-to-end test
  of REST API, 151–153
  testing middleware with spies, 146–148
wish list Next.js page, 243–244
WORKDIR keyword, 175

Y
YAML, 188
yarn, 4