

INDEX

Symbols

- + (addition operator), 36–37
- @, 155, 197
- >!! (blocking put), 235–238
- <!! (blocking take), 235–238
- . (dot operator), 258–259
- = (equality operator), 39
- #, 128
- >! (parking put), 235–238
- <! (parking take), 235–238
- ' (quote) reader macro, 154–155
- & (rest parameter), 54

A

- abstractions, 265–266
 - implementing, 270
 - through indirection, 77
 - with macros, 183
- abstract syntax tree (AST), 149
- add-watch, 216
- alias, 132
- alter, 219–221
- alter-var-root function, 227
- and (Boolean operator), 40
 - source code, 168
- apply function, 91
- architecture, of code, 110
- arity, 52–54
 - overloading, 52–53
- artifact ecosystem, 277–280
- artifacts, 277
- assoc-in, 114
- AST (abstract syntax tree), 149
- asynchronous tasks, 191
- atomic values, 210
- atoms, 212–217
- auto-gensym, 177–178

B

- binding
 - with def, 40–41
 - dynamic vars, 223–227
 - with let, 61–63
 - local, 157–158
- blocking, 191
- blocking put (>!!), 235–238
- blocking take (<!!), 235–238
- Boolean
 - expressions, 39–40
 - forms, 37–38
 - operators, 40
 - values, 39
- Boot
 - classpath isolation, 289
 - composition, 285–288
 - deftask, 282
 - documentation, 283
 - filesets, 282, 288–289
 - middleware, 285–286
 - middleware factories, 285, 287–288
 - optarg, 284
 - REPL, 284
 - tasks, 282–284
- bound-fn, 227
- Brave and True Ale example, 180

C

- callbacks
 - hell, 244
 - with promises, 202
- Central Repository, 279
- chan, 235
- channels, 235–237
 - buffering, 236–237
 - timeout, 242

- char, 120
- cheese heist, 130, 140–144
- CIDER package, 23–28
 - handling errors, 27–28
 - installation, 23
 - key bindings, 25–27
 - starting, 23
- class instantiation, 272
- classpath, 252
- Clojars repository, 279
- Clojure
 - compiler, 4
 - hosted language, 4
 - metaphysics, 210–211
- clojure.jar*, 255–257
- closures, 58
- collection abstraction, 88–90
- command-line interaction, 121–124
- comma-separated values (CSV), 93
- commute, 221–223
- compare-and-set, 214–215
- comp function, 105–106, 120
- compilation, 248–252
- complement function, 92–93
- concat function, 84
- concurrency, 190–196
 - dining philosophers
 - problem, 195–196
 - dwarven berserker problem, 195
 - mutual exclusion problem,
 - 194–195, 210
 - preventing with delays, 199
 - nondeterministic execution,
 - 208–210
 - queues, 243–244
 - reference cell problem, 193–194,
 - 207, 209
 - preventing with
 - promises, 202
 - stateless, 228–232
 - tasks, 190
 - The Three Concurrency
 - Goblins, 193–196
- conj function, 45, 46, 47, 90
- cons function, 74–77
- contains? function, 47
- control flow, 37–40
 - Boolean expressions, 39–40
 - do operator, 38
 - if expression, 37–38
 - when operator, 38–39
- core.async
 - alts!, 235, 242
 - alts!!, 235, 241–243
 - blocking, 237–238
 - buffering, 236–237
 - events, 233
 - hot dog vending machine
 - example, 239–241
 - parking, 237–238
 - pipelines, 240, 244–245
 - put, 235–238
 - queues, 243–244
 - take, 235–238
 - thread, 238–239
 - timeout channels, 235, 242
 - waiting, 235–238
- create-ns, 129
- CSV (comma-separated values), 93
- cuddle zombie, 208

D

- data structures, 41–48
 - immutable, 42, 100–105, 210
 - keywords, 44–45
 - lists, 45–46
 - maps, 43–44
 - numbers, 42
 - sets, 46–47
 - simplicity, 48
 - strings, 42–43
 - vectors, 45
- data types, 265–266
 - extending, 270
 - instances, 266
- def
 - naming values, 40–41
 - storing objects, 127–129
- defprotocol, 270
- defrecord, 272
- deftype, 275
- delays, 196, 198–199

- deliver, 200
- dependencies, 277, 278–279
- deref, 128, 197
 - reader macro, 155
 - timeout, 202
- dereferencing
 - atoms, 212
 - delays, 198–199
 - futures, 191–198
 - promises, 200–202
- destructuring, 54–56
- dispatching function, 266, 267–268
- dispatching value, 266, 267–268
- dispatch value, 267–268
- distributed computing, 191
- domain-specific language (DSL), 283
- do operator, 38
- dorun function, 229
- doseq, 121
- dosync, 219–221
- doto macro, 260
- dot special form, 258
- dot operator (.), 258–259
- drop function, 81
- drop-while function, 81–82
- DSL (domain-specific language), 283
- Dungeons and Dragons, 265
- dynamic binding, 223–227

E

- Eastwood plug-in (lint tool), 279–280
- editors, 9
- El Chupacabra, 218
- elisp, 11, 17, 19, 72
- Emacs
 - buffers, 14–15
 - CIDER package, 23. *See also* CIDER package
 - configuration, 13
 - cursor, 20
 - customizing, 15–16
 - files, 15–17
 - frames, 24
 - help, 22
 - installation, 12–14
 - key bindings, 17
 - killing, 21–22
 - kill ring, 21–22
 - Lisp (elisp), 11, 19
 - mark, 20
 - minibuffer, 15
 - modes, 18–19
 - line, 18
 - major, 18
 - minor, 19
 - movement, 20
 - packages, 19
 - Paredit, 28–30. *See also* Paredit
 - point, 20
 - regions, 20
 - windows, 24
 - yank, 21
- empty? function, 88
- equality operator (=), 39
- eval, 151
- evaluation, 36
 - lists, 159–160
 - macros, 160–162
 - model, 148–152
 - rules, 155–162
 - to self, 156
 - symbols, 156
- evaluator, 148, 149–152, 155
- expression, 36
 - Boolean, 39–40
 - function, 98
 - if, 37–38
- extend-protocol, 271, 274
- extend-type, 270, 274

F

- false (value), 39
- falsey values, 39
- fields, 272
- file naming conventions, 134
- filter function, 83
- first function, 74
- force, 198
- forms, 36–37
- fully qualified symbols, 171

- functional programming, 79, 97
 - immutable data structures, 100–105
 - Peg Thing game, 108–124
 - pure functions, 98–100
- functions, 48–59
 - anonymous, 57–58
 - arity, 52–54
 - overloading, 52–53
 - calls, 48–51, 159
 - composition, 103–105, 285
 - defining, 51–52
 - expression, 48
 - higher-order, 49
 - pure, 98–100, 105–107
 - rest parameters, 54
- futures, 196–198, 202–205

G

- gensym, 177–178
- get function, 43–44, 45, 47
- get-in function, 44
- go blocks, 235–239
- grain size, 230

H

- Handy, Jack, 19
- hash-map function, 43
- hash-set, 46
- head (of a sequence), 65
- Hickey, Rich, 4
- hierarchical dispatching, 269
- hobbits
 - modeling, 59–61
 - targeting, 67–68
- homoiconic languages, 148, 152
- hot dog vending machine, 239–243
- humans, 265

I

- identity (Clojure metaphysics), 211
- identity function, 95
- if expressions, 37–38
- if-let, 119

- immutable data structures, 100–105, 210, 212
- implementing abstractions, 74
- importing
 - Java classes, 253–254
 - record types, 273
- in-ns function, 129
- installation
 - Emacs, 12–14
 - CIDER package, 23
 - packages, 19
 - Leiningen, 5
- instance, of a data type, 266
- interfaces, 77
- interleaving, 190, 192–193
- interning, 128
- into function, 88–89

J

- JAR files, 4, 249, 255
- Java
 - bytecode, 4, 248–249, 252
 - classes, 273
 - classpath, 252, 254–255
 - entry point, 255
 - imports, 253–255
 - interop. *See* Java interop
 - JAR files, 4, 249, 255
 - main method, 252, 255, 257
 - packages, 253–255
 - stacks, 259–260
- javac, 252
- Java interop, 250, 257–261
 - creating objects, 259–260
 - Date class, 262
 - files, 262–264
 - importing, 260–261
 - input/output, 262–264
 - method calls, 258
 - mutating objects, 259–260
 - passing arguments, 258
 - syntax, 258–259
 - System class, 261–262
- Java Virtual Machine (JVM), 4, 150, 248–249
 - threads, 191–193
- just-in-time compilation, 248

K

key functions, 84
keywords, 44–45

L

Lady Gaga, 190–191
lazy sequences, 84–88, 112
 chunking, 86
 defining, 87
 efficiency, 84–87
 infinite, 87–88
 realizing, 84
 repeatedly function, 87
 repeat function, 87
Leiningen build tool, 5–8, 277–280
 dependencies, 278–279
 identification, 278
 plug-ins, 279–280
let, 61–63
line-seq function, 264
linked list, 74–76
lint tool (Eastwood plug-in),
 279–280
Lisp, 4, 11, 36, 150
list function, 46
lists, 45–46
 evaluation rules for, 159–160
literals, 36
local binding, 157–158
loop, 63–64

M

macroexpand, 162
macros, 147–185
 argument destructuring,
 167–168
 Brave and True Ale example,
 180–184
 building lists for evaluation,
 168–173
 calling, 50–51
 characters, 154
 defining, 167
 distinguishing symbols and
 values, 168–169
 evaluation rules for, 160–162

 expansion, 162
 gotchas, 176–180
 composition, 179–180
 double evaluation, 178
 variable capture, 176–178
 infection, 166
map function, 50, 73, 79–80
maps (data structure), 43–44
 destructuring, 55
Maven, 278, 279
McCarthy, John, 148
McFishwich, 85
memoize function, 107
metaphysics, Clojure, 210–211
multimethods, 266–269
 default, 268

N

names
 in Clojure metaphysics, 211
 collision, 128–129
 for values, 40–41
namespaces, 126
 aliasing, 136
 create-ns, 129
 creating and switching to,
 129–130
 current, 126
 in-ns, 129
 ns macro, 126, 129, 138–140
 ns-interns, 128
 ns-map, 128
 refer-clojure, 138
 reference, 138
 referring, 130–132, 135
 requiring, 134–137
 user, 126
 using, 136–137
nil (value), 39, 47
nondeterministic execution, 193
not-empty, 120
ns macro, 126, 129, 138–140
ns-interns, 128
ns-map, 128
nth function, 46
numbers, 42

O

- object-oriented metaphysics, 208–210
- object-oriented programming, 104, 250–251
 - classes, 251
 - methods, 251
 - objects, 250–251
- operators, 36
 - addition (+), 36–37
 - Boolean
 - and, 40
 - or, 40
 - do, 38
 - dot (.), 258–259
 - equality (=), 39
 - when, 38–39
- or (Boolean operator), 40

P

- parallelism, 190–196. *See also* concurrency
- Pareidit, 28
 - barfing, 29–30
 - navigation, 30
 - slurping, 29
 - wrapping, 29
- partial function, 91–92, 120
- Peg Thing game, 108–124
- Perlis, Alan, 48
- philosophy (of Clojure), 48
- plug-ins, 279–280
- pmap, 228–232
- polymorphism, 77, 266
 - multimethods, 266–269
 - protocols, 77, 269–272
- predicate functions, 81–82, 119–120
- processes, 234–239
 - blocking, 237–238
 - buffering, 236–237
 - parking, 237–238
 - thread, 238–239
- programming to abstractions, 72
 - indirection, 77–78
 - linked lists, 74–77
 - sequences abstraction, 72–74

projects

- building, 7
- creating, 5–6
- organizing, 133–140
- running, 6
- promises, 196, 200–205
- protocols, 77, 269–272
- proxy, 275
- pure functions, 98–100, 117

Q

- queues
 - macro, 202–205
 - processes, 243–244
- quote, 160
- quote (') reader macro, 154–155
- quoting, 127, 169
 - simple, 169–171
 - syntax, 171–174
 - unquote splicing, 174–176
 - unquoting, 172, 175
 - with when, 170

R

- reader, 148, 150, 153–155, 264
 - form, 128, 153
 - macros, 57, 153, 154–155
- read-string function, 153–154
- realized?, 198
- records, 272–274
- recur, 102–103
- recursion, 100–103
- reduce function, 66–67, 80–81, 114
- reducers library, 231–232
- refer, 130–132
- reference types, 211
 - atoms, 212–215
- referential transparency, 98–99
- refs, 218–223
- regular expressions, 64
- reify, 275
- repeatedly function, 87, 228
- repeat function, 87
- REPL, 7–9, 23–24
 - Boot, 284
- repositories, 278

require, 111, 134–137
rest function, 74–77

S

scope, 61
Semantic Versioning system, 278
sequence (seq), 73–74
 abstraction, 72–77
 function examples, 79–84
 lazy, 84–88
sets, 46–47
s-expressions, 150
side effects, of functions, 98,
 99–100
Simmons, Richard, 165
 were-Simmons, 267
simplicity, of data structures, 48
slurp function, 29, 94, 263
sock gnomes, 218–220
software transactional memory
 (STM), 218
some function, 83
sort-by function, 84
sort function, 84
special forms, 50–51, 156, 159–160
spit function, 144, 263
Stallman, Richard, 13
state, 207, 211, 212
 mutable, 208–210
STM (software transactional
 memory), 218
strings, 42–43
 concatenating, 43
 pattern matching, 64
SVG, 140
swap!, 212–215
Swift, Taylor, 9
symbols, 126, 156–158
 fully qualified, 129–130
 resolving, 156–157
synchronous tasks, 191
syntactic abstraction, 162, 163
syntax, 36
 Java interop, 258–259

T

tail call optimization, 102
tail (of a sequence), 65
take function, 81
take-while function, 81–82
telepath, 225
thread-bound? function, 226
threads, 191–193
 delays, 198–199
 futures, 196–198
 nondeterministic programs, 193
 promises, 200–202
 spawning, 192
Thread/sleep, 196–197
Tick, The, 53
transactions, 218–221
troll, 225
true (value), 39
truthy values, 39
tuples, 119
types, 266

U

unless macro, 171
unquote splicing, 174–176
unquoting, 172, 175
use function, 136–137

V

validators, 217
values, 210–211
vampire
 data analysis, 93–96
 food journal, 79–84
 Vampire Diaries, The, 71
variable assignment, 101
vars, 126–129, 223–227
 binding conveyance, 227
 dynamic binding, 223–226
 interning, 128
 per-thread binding, 226–227
 private, 132
 reader form, 128
 roots, 227

vector function, 45
vectors, 45

W

watches, 215–216
were-Simmons, 267
when operator, 38–39, 166, 170
whitespace (to separate
 operands), 36
with-open, 264
with-redefs, 227

Y

yak butter, 200–202