

Keywords and symbols

, 25, 158
; 21
:as 40–41, 48–49, 182, 264
:exclude 41, 182, 258
:import 42, 182
:load 182
:only 182–183
:post 133–134
:pre 133–134
:private 181
:refer 41–42
:refer-clojure 182, 258
:require 41, 182
:strs 49
:test 128
:use 41, 182–183
 naked :use 183
.. 37
read-eval 270
warn-on-reflection 278, 289
& 48, 50

A

A* pathfinding 149
 astar 149–154
 candidate paths 150
 cost-estimate function 150
 path-finding 149
abstraction-oriented
 programming 16, 177,
 184, 189, 206, 285–287
abstractions 166, 188, 208, 227,
 284, 286
accessors 202

ACID 238
ad-hoc hierarchies 16, 70, 184,
 186, 188, 304
 derivation hierarchy 186–187
 make-hierarchy 187
adornment 278
Agents 235, 240, 247, 249–255,
 274
 agent-error 253
 await 249–250, 255
 await-for 250–251
 queue 247, 251
 restart-agent 253–254
 send 248, 251–253
 send-off 248, 251–253
 serialize access 249
agile 202, 278
Ahead of Time Compilation
 (AOT) 218, 227–228
alert 216
algorithm 81
alias 183
anaphora 170–171, 211
 anaphoric 173
 anaphoric macros 170, 211
ancestors 186
and (logical) 114, 175
anonymous inner classes 126
Apache Ant 181
Application Programming
 Interface (API) 204, 293
 public-facing API 181
apply 98, 127
aquarium 69
Arc (programming
 language) 170
awhen 170

arglists 164
argument 68
arrays 74, 77, 79, 218–220
 amap 220
 areduce 220
 multidimensional arrays 222
 primitive arrays 218–219
 reference arrays 218
 seq of an array 220
as-futures 263–265, 268
as-is 48
aspects 135
assertions 173, 203
 assert 134
assoc 84–86, 191
associative 66
assoc-in 85
asum-sq 220, 278–280
asynchronous 240, 247
atomicity 238, 255
Atoms 235, 240, 255–257
 compare-and-set! 255
autoboxing 288
auto-gensym 36, 165

B

backslash 73
bad-make-move 243
beget 184
benchmark 281
best 72
best practices 303
best-case 81
best-first 149
BigDecimal 273–274

- Big-O 77, 80–82
 - algorithmic complexity 77
 - asymptotic complexity 80–81, 124
 - linear time 81
 - logarithmic complexity 81
 - bind 47
 - binding 47, 300
 - bit-xor 51
 - blocks 29, 31
 - boilerplate 162, 164–165
 - Booleans 279
 - Boolean (Java class) 44, 83
 - Boolean context 45
 - bot object 139–140
 - bottom-up programming 298
 - bound 272
 - boxing 288
 - autoboxing 288
 - unboxing 288
 - breakpoint 309–310
 - build-contract 174, 181
 - build-move 205–206
 - byte 83
 - bytecode 141, 208, 210, 222
 - bytes 279
- C**
-
- C (programming language) 8, 21, 194, 198
 - calculations, precise 66
 - call stack 71
 - callbacks 138, 266
 - call-by-need 119
 - canvas 51
 - capitalize 41
 - capturing group 74
 - car 80, 90
 - case 147, 298
 - chaining 202
 - change-message 209, 211
 - char 83
 - character classes 73
 - characters, in a string 23
 - chars 279
 - chess 16, 202, 204–205, 241–242
 - choose-move 242
 - chunked sequences 283
 - clarity 7, 290
 - clear 55
 - clear-actions 253
 - Closure
 - as opinionated language 312
 - Closure 1.3 271, 307
 - expressiveness of 5
 - succinctness 72
 - clojure.contrib.json 168
 - clojure.contrib.repl-utils 52–53
 - clojure.core 41, 179, 183, 271
 - clojure.lang.IMeta 212
 - clojure.lang.PersistentStack 86
 - clojure.lang.PersistentQueue/EMPTY 92
 - clojure.main/repl 309
 - clojure.set 40, 94, 96, 183, 294
 - clojure.string 41
 - clojure.test 128, 183, 300
 - clojure.test/run-tests 128
 - clojure.xml 183, 262
 - clojure.xml/emit 305
 - clojure.xml/parse 263, 306
 - clojure.zip 183, 262
 - clojure.zip/xml-zip 263
 - closures 17, 135–141, 197, 230, 258, 305
 - closing over parameters 137
 - functions returning
 - closures 136
 - lexical closures 17
 - passing closures as
 - functions 137
 - sharing closure context 138
 - code
 - avoid repeating 196
 - is data 8, 11, 157–158, 161, 169
 - repetitive 47, 49
 - coercion 277, 287–291
 - fifth rule of coercion 291
 - first rule of coercion 288
 - fourth rule of coercion 290
 - second rule of coercion 290
 - third rule of coercion 290
 - collect-bodies 173–174
 - collection 46, 280
 - colon 23
 - columns 130
 - commenting 21, 74
 - commits 248
 - Common Lisp (programming language) 46, 72, 169, 298
 - cdr 80, 90
 - Common Lisp Object System (CLOS) 187
 - funcall 72
 - Lisp-2 72
 - comp 127, 130
 - comparator 95
 - compare-and-swap 255
 - compile 141, 213–214
 - compiler 117, 213, 218, 277, 288–291
 - Compiler.java 55
 - compile-time 141, 231
 - complement 128, 130
 - complexity 103, 238, 273
 - essential complexity 5
 - incidental complexities 164
 - composition 305
 - concat 281
 - concrecence 13
 - concrete classes 16, 193, 195, 208
 - concrete realization 286
 - concurrency 3, 14, 109, 234–237, 245, 249, 258, 261
 - concurrent modification 220, 259
 - concurrent modification errors 109
 - concurrent
 - programming 234–235, 248, 259
 - lost wakeup 236
 - model, distributed 248–249
 - priority inversion 238
 - vs. parallelism 235
 - cond 69, 162, 298
 - confounding 281
 - conj 86, 88, 90–92, 110, 196
 - cons 32–33, 90
 - consistency 7–8, 238, 242, 261
 - reporting problem 238
 - constant factor 81
 - constant time 281
 - constructors 37, 55, 198, 204, 212, 305
 - contains? 89, 91, 95
 - continuation-passing style (CPS) 148–149, 229, 304
 - accept function 148
 - return continuation 148
 - continuations 266, 307
 - continuation function 148
 - continue 252, 254
 - contracts programming 173, 175–176, 299, 302–303, 312
 - constraints 133, 173, 175, 301–302
 - defcontract 312
 - contrib 168, 271
 - control structures 157, 159, 161, 163
 - conversion specification 296

coordination 241, 269
 copies, defensive 226
 counted 90
 counter-optimizations 280
 count-tweet-text-task 263–264,
 266
 create-ns 179–180
 crypto 69

D

data structures, immutable
 108–109
 database 236
 deadlock 236, 238, 248–250,
 259, 267
 deterministic 267
 debug 309
 debug console 309
 debugging 293, 310, 312
 debug-repl 309
 decimal 22
 declarative 118, 293
 declare 173
 def 27–28, 135, 137, 181, 271,
 273
 default-handler 209–210
 defformula 303–304
 definterface 227–229
 defmacro 162, 181, 271
 defmulti 188, 271
 defn 17, 27, 136, 164, 181, 271
 defonce 271
 defprotocol 229
 defrecord 189–191, 199–200
 defstruct 182, 190
 downfall of defstructs 190
 defunits-of 296–297
 defwatched 165
 delay 113, 119, 121, 149, 287
 delegate 215, 286
 dependency injection (DI) 305
 deref 240, 255, 270, 272
 derive 186–187
 descendants 186
 design patterns 292, 303
 Abstract Factory pattern 304
 Builder pattern 305
 chain of responsibility 305
 Façade pattern 305
 Interpreter pattern 305
 Observer pattern 303–304
 Prototype pattern 305
 Strategy pattern 304

 subscriber patterns 129
 Visitor pattern 304
*Design Patterns: Elements of
 Reusable Object-Oriented
 Software* 303
 destructuring 8, 47–48, 54, 57,
 133, 242
 associative destructuring 50
 in function parameters 50
 nested 49
 versus accessor methods 50
 with a map 49
 with a vector 48
 determinacy 235
 directive 48–49, 52
 directory structure 180–181
 disappear 53
 dispatch 187
 display 212–213
 dissoc 86, 190–191, 257
 distributive 66
 do 29, 163, 262
 doall 274
 documentation, viewing 53
 domain 166
 domain-specific language
 (DSL) 10–11, 70, 164, 167,
 173, 204, 292–295, 298,
 301
 domain expertise 302
 putting parentheses around
 the specification 295
 unit conversion DSL 297
 don't panic 55
 doseq 7, 46, 55
 dosync 242, 248
 dothreads! 240, 255
 doto 38
 double 63, 83
 double-array 222
 double-backslash 73
 double-quotes 73
 doubles 65, 279
 do-until 162
 drawing 53
 duck typing 189
 dummy write 239
 durability 238
 DynaFrame.clj 213
 dynamic binding 287, 293, 306,
 308
 binding 117, 271, 273, 290,
 306, 308
 bound-fn 274

dynamic scope 273–274, 306,
 308
 dynamic type systems 278, 287

E

elegance 13, 45
 embedding 73
 empowerment 6
 empty sequence 78–79
 empty? 46
 encapsulation 16–17, 305
 block-level encapsulation 17
 local encapsulation 17
 namespace encapsulation 17
 Enlive 168
 enumeration values 68
 enumerator 78
 env 310
 ephemeral 280
 equality 68, 70–71, 109, 187
 equality partitions 79, 102,
 224
 equality semantics 79
 Erlang (programming
 language) 248–249
 actor 248
 Actor model 143
 in-process versus distributed
 concurrency models 248
 share-nothing 248
 error handling 229, 254,
 306–308
 escaped 73
 evaluation 32–33
 contextual-eval 160–161, 310
 eval 159–161, 309–310
 meta-circular 293
 exceptions 38, 55, 205, 229,
 236, 252–254, 282, 293,
 306–307
 catch 38, 308
 checked 229–230
 compile-time 55, 231
 ConcurrentModification-
 Exception 259
 finally 172, 261
 handling 229
 java.lang.ClassCastException
 231
 java.lang.Exception 230
 java.lang.NullPointer-
 Exception 114
 java.lang.RuntimeException
 230

exceptions (*continued*)
 runtime 230
 runtime vs. compile-time 230
 throw 38, 54–55, 231
 expand-clause 11
 expansion 297
 expected case 81
 experimentation 179
 expression problem 16
 extend 15, 192, 195–197, 199
 extend-protocol 192, 197
 extend-type 192–193, 197
 Extensible Markup Language
 (XML) 168, 262–264,
 305–306

F

Factor (programming
 language) 12, 301
 factory methods 305
 fail 252, 254
 false 44–45, 56
 evil-false 44
 Fantom (programming
 language) 207
 fence post errors 203
 filter 78, 119, 129, 138, 304
 find-doc 51
 find-ns 179
 finite state machines 146
 first 78, 80, 90, 121, 127
 First In, First Out (FIFO) 91,
 192
 First In, Last Out (FILO) 192
 first-class 12, 27, 128
 fixed-size pool 252
 FIXO 192–193, 195, 197
 fixo-peek 199
 fixo-push 193, 195, 200
 flexibility 9
 float 83, 291
 floating point 22, 62, 65–67
 overflow 83
 rounding error 64
 underflow 64, 83
 floats 279
 fluent builder 202–203
 FluentMove 204–205
 fn 26, 28, 82, 135–136, 146, 164
 for 7, 51
 force 113, 119, 121
 forever 68
 form 24

free variables 136
 freedom to focus 5
 frequencies 288
 Frink (programming
 language) 207, 294
 frustrating 246
 fully qualified 258, 270, 272
 fun 51
 functional programming 12,
 19, 108, 303
 currying 127
 first-class functions 72, 126,
 130
 functional composition 169,
 288, 294
 higher-order functions 115,
 129–131, 137, 142, 151,
 154, 161, 173–174, 230,
 287, 304
 partial application 127, 305
 pure functions 5, 118,
 131–132, 149, 241, 298
 referentially
 transparent 131–132, 256,
 284
 functions 25, 68
 anonymous 28, 129, 138, 210
 arity 26
 Calling Functions 25
 dangerous 75
 function signatures 192
 local 147
 multiple function bodies 50
 named arguments 132
 futures 235, 261–263, 265, 269
 as callbacks 262
 future 149, 255, 274
 future-cancel 264
 future-cancelled? 264
 future-done? 264

G

Gang of Four 303
 garbage collection 207, 280,
 283
 gcd 145
 gen-class 38, 212, 214–215,
 218, 233, 304
 generalized tail-call
 optimization 143–144, 149
 generic 46
 genotype 9
 gensym 169
 get 84, 88, 94, 98

get-in 85–86
 getter 204
 global hierarchy map 188
 goal 247
*Gödel, Escher, Bach: An Eternal
 Golden Braid* 183
 good-move 242
 Graham, Paul 298, 309
 graphic 54
 graphical user interface (GUI)
 138, 212, 214–216, 218
 graphics context 53
 greatest common
 denominator 144, 284
 green thread 254
 Greenspun's Tenth Rule 303,
 305
 Groovy (programming
 language) 207

H

Halloway, Stuart 101, 208
 has 285
 hash maps 99
 hash-map 80, 88, 97, 190
 Haskell (programming
 language) 12, 47, 113,
 118, 149, 161, 198, 285, 301
 out of order execution 149
 Template Haskell 161
 typeclasses 198, 285
 heuristic 151
 Hickey, Rich 4–5, 206
 hidden 52
 hierarchy 205
 history 246
 homoiconicity 12, 293
 hooks 304, 309
 hops 81
 host semantics 62
 Hoyte, Doug 294
 hypkens 181

I

I/O 239, 249, 252
 idempotent 239, 255–256
 identical? 70–71
 identifiers 70
 identity 3, 13–14, 71, 202, 211,
 227, 235, 247, 254
 IDEs 308
 idiom 190

- idiomatic 34, 36, 46, 62, 82, 87, 98, 114, 154, 158, 169, 179, 198, 277, 280, 298
 - if 31, 114, 162
 - if-let 120, 171
 - image 51, 56
 - immutability 7, 14, 30, 77–78, 86, 91, 93, 100, 107–109, 136, 143, 202, 224, 248, 280
 - imperative programming 14, 30, 108
 - implementation details 182
 - implicit do 30
 - inc 243
 - inconsistency 245
 - index numbers 47
 - infinite lazy range 68
 - infinite loop 46
 - infinite sequence 113, 118, 283
 - infix notation 25
 - inheritance 5, 183, 185–186
 - differential 185
 - implementation
 - inheritance 196
 - interface inheritance 16
 - prototype chain 184, 187
 - prototype maps 183, 186
 - inherited behaviors 186
 - init 55
 - init-proxy 212
 - in-process programming
 - model 249
 - instance 55, 71
 - int 83, 146
 - integers 22
 - overflow 64
 - interactive command
 - prompt 51
 - interfaces 8, 16, 192–193, 195, 200, 224, 285
 - intern 179–180, 270, 272
 - internal reduce 280
 - interop 73
 - interoperability 24, 62, 196, 200, 208, 218, 222, 224, 227, 277, 304
 - accessing Java instance members 37
 - accessing static class members 36
 - creating Java class instances 36
 - setting Java instance properties 37
 - interpreted 298
 - into 83, 89, 196
 - into-array 218–219, 222
 - ints 279
 - invariants 203, 246
 - inversion of control 164
 - invoke 214
 - Io (programming language) 185
 - Ioke (programming language) 207
 - isa? 186–187
 - ISeq 121
 - ISeqable 200
 - ISliceable 228–229
 - isolation 238
 - iteration 46, 305
 - iterator 78, 88
- J**
-
- Jahad, George 160
 - Java 37, 208, 290
 - Java (programming language) 4, 6, 8, 16, 18, 21, 32, 36, 38, 67, 91, 108–109, 129, 179, 192, 194, 198, 207, 209, 211, 215, 224, 235, 255, 260, 290
 - Java 7 126
 - Java Collections Framework 224, 226
 - Java libraries 52
 - polymorphic print facility 204–205
 - variadic constructor 222
 - Java Virtual Machine (JVM) 4, 6, 36, 62, 65, 144, 146, 207–208, 220, 229, 234, 251, 280
 - HotSpot 207
 - java.awt 52
 - java.awt.Container 212, 215
 - java.awt.Frame 52
 - java.io.FilterOutputStream 211
 - java.io.OutputStream 211
 - java.lang 42, 179
 - java.lang.Iterable 225
 - java.lang.Math/round 291
 - java.lang.Object 128, 205, 220, 279
 - java.lang.Runnable 222
 - java.lang.StackOverflowError 142, 230
 - java.lang.String 15
 - java.lang.String/format 222
 - java.lang.StringBuilder 218
 - java.lang.Thread 255
 - Java.next 208
 - Java.next Mantra 208
 - java.util.ArrayList 79
 - java.util.Collection 225
 - java.util.Collections/sort 223
 - java.util.Comparator 222–223
 - java.util.concurrent 258, 261
 - java.util.concurrent
 - FutureTask 224
 - java.util.concurrent.atomic
 - AtomicInteger 136, 256
 - java.util.concurrent.BlockingQueue 91
 - java.util.concurrent.Callable 222
 - java.util.concurrent.locks 260
 - java.util.concurrent.locks
 - ReentrantLock 260
 - java.util.concurrent.locks
 - ReentrantReadWriteLock 261
 - java.util.List 15, 79, 223, 225
 - java.util.Observable 304
 - java.util.RandomAccess 48
 - java.util.regex.Matcher 48, 75
 - java.util.regex.Pattern 73
 - javadoc 53, 73
 - JavaScript (programming language) 16, 135, 141, 184
 - javax.swing.JFrame 212
 - Jess (programming language) 207
 - Joswig, Ranier 295
 - joy.gui.DynaFrame 212
 - JRuby (programming language) 207
 - just-in-time (JIT) 207
 - juxt 188
 - Jython (programming language) 207
- K**
-
- kapow 143
 - keep-indexed 102
 - key 49, 80, 88
 - keys-apply 131–132
 - keyword arguments 278
 - keywords 23, 68–70, 241
 - plumbing, separating from domain 69
 - qualified 69
 - ubiquity of 75
 - Kingdom of Nouns 18

L

lambda calculus 12
Lambda Papers 143
 language, eager 113
 last 86
 laziness 113, 117, 119, 122, 124, 274, 282
 combinatorially exploding computations 113
 full realization 283
 full realization of interim results 117, 122
 lazy evaluation 149
 lazy sequences 68, 87, 131, 268–269, 274, 283–284
 lazy-seq 115–116, 143
 lazy-seq recipe 115, 117, 143
 short-circuiting laziness 114
 Leiningen 181
 let 29, 36, 52, 82, 117, 135, 137, 170, 273, 290
 letfn 18, 147
 lexical scope 135, 273–274, 290
 lexical bindings 309–310
 lexical context 138, 169
 lexical contour 170
 lexical environment 138, 140, 309
 line number 55
 line terminators 74
 linear search 100
 Lisp (programming language family) 4, 8–9, 22, 25, 30, 36, 68, 79–80, 87, 90, 126, 172, 202, 295, 298, 304
 beauty of 9
 cons-cell 79
 Lisp-1 62, 72
 lists 24–25, 33, 79, 90–91, 122, 159
 as stacks 91
 empty 44
 PersistentList 90
 singly linked 90
 literal 73
 literal syntax 23–25, 83, 89, 97, 189
 live-lock 239
 local-context 138, 310
 locals 29–30, 72, 117, 136, 143, 272–273, 290
 local bindings 310

locking 14, 234, 236, 238, 250, 259–261
 blocking 224, 248, 250–251, 261, 265–266
 contention 260–261
 explicit locks 261
 fairness 250
 orphaning 238, 259
 reentrant 260
 striping 261
 total ordering 238
 log-agent 249
 logarithmic 81
 logging 249, 251
 long 83, 279
 look-around clauses 73
 lookup 17–18, 68, 285–286
 loops 30–31, 146, 290
 loop invariants 134
 loop locals 31
 loop termination 46
 lowercase 74

M

M literal 21, 62–63
 macroexpand 161
 macroexpand-1 161
 macros 9, 24, 34, 55, 72, 119, 160–161, 164, 166, 169–170, 173, 231, 263, 296–298, 303, 305, 308, 310–311
 combining forms 164
 compile-time 31, 157, 161, 169, 230
 hygienic 171
 macro that builds another macro 297
 macro-definition time 171
 macro-expansion time 169, 310
 returning functions 173
 rules of thumb 161
 selective Name Capturings 171
 using to change forms 165
 using to control symbolic resolution time 169
 using to manage resources 171
 magical formula 269
 main 203
 make-array 219
 make-dumb-array 258
 make-move 246
 make-safe-array 260
 make-smart-array 261
 manip-map 131–132
 map 25, 68, 71, 78–79, 87, 97–98, 100, 102, 119, 129, 189, 204, 222, 269, 273–274, 295, 304
 array map 98, 100, 190
 PersistentHashMap 80
 thinking in maps 97
 mapped file 74
 math-context 273–274
 Maven 181
 max-history 247
 McCarthy, John 9
 memoization 132, 256–257, 277, 284–287, 291
 BasicCache 286
 cache 284–285
 caching protocol 286
 hit 285–286
 manipulable-memoize 256–257, 286
 memoization protocol 285–286
 memoize 256–257, 284–285
 miss 285–286
 PluggableMemoization 286–287
 through 286–287
 Mersenne primes 283–284
 metadata 69, 71, 128, 164, 181, 191, 256
 attaching 71
 meta 71, 191
 with-meta 71, 191
 methods 52, 55
 metric units 294
 min-by 151, 154
 min-history 247
 mini-language 8, 47, 298
 mixins 193
 ML (programming language) 12
 monitors 236, 259
 monitor context 259
 monkey-patching 16, 195
 Montoya, Inigo 77
 more 48
 Move 202, 205
 multi-line mode 74
 multimethods 15, 68, 70, 185, 187–188, 202, 304–305, 307
 multimethod dispatch 221

multimethods (*continued*)
 prefer-method 187
 remove-method 187
 multiple transactions 246
 multiversion concurrency
 control (MVCC) 236, 239
 snapshot isolation 245
 write skew 239, 246
 mutability 78, 108–109, 136,
 220, 224, 235, 239, 255,
 259, 261, 280–282
 isolated pools of mutation
 points 298
 mutable fields 202
 mutable state 14
 mutation 109, 235, 239, 257,
 298
 mutators 202

N

name resolution 72
 name shadowing 72, 169
 named arguments 133
 named structures 50
 namespaces 17, 21, 36, 39,
 42, 69, 71, 94, 163, 169,
 178, 180–182, 189, 198,
 212–213, 215, 262, 270,
 272, 305
 as class specifications 212
 compilation 214
 in-ns 179
 name-mangled local 290
 namespace compilation 214
 namespace qualification 69
 ns 39, 70, 179, 182, 212
 privacy 181
 qualification 40, 69, 170,
 179, 183
 remove-ns 180
 two-level mapping 178, 180
 user 21
 natural language 296
 neighbors 85–86, 150, 242
 nest 236
 nested syntax-quotes 297
 new 36
 next 46, 57, 79, 89–91, 110,
 115–116
 nil 24–25, 44–45, 49, 56, 78,
 80, 94, 101, 110, 117,
 163, 195, 253
 nil pun 57
 nondeterminism 235

non-termination 113
 non-thread-safe 75
 nouns 13
 ns-unmap 179
 nth 84, 91
 nthnext 116
 null 203
 numbers 21–22
 binary 22
 distribution of represented
 numbers 65
 hexadecimal 21–22
 octal 22
 promotion 63–64
 radix notation 22
 radix-32 22
 scientific notation 22
 numerical precision 75

O

object-oriented
 programming 3, 5, 8,
 13, 15, 18–19, 32, 97, 164,
 205–206, 292, 303, 305
 conflict resolution
 strategy 187
 hierarchies 5, 18, 103, 164,
 204, 298, 304
 objects 279
 obscure 47
 occur-count 288–289
On Lisp 298
 one-at-a-time realization
 282–284
 operator precedence 5, 25
 optimizations 277, 280, 289
 option flags 74
 or (logical) 49, 305
 original map 49

P

pair 49
 parallelism 149, 235, 261,
 264–266, 269
 dataflow 267
 parentheses 9
 parents 186
 partial 127–128, 130, 242
 partition 298
 Patriot missile 64–65
 pattern matching 47, 249
 patterns 54, 73

pcalls 235, 268–269
 peek 86, 91, 93, 192
 performance 190, 202, 277,
 291
 measurements 282
 Perl (programming
 language) 135
 persistent data structures 7, 77,
 81, 90, 110, 113, 280–281
 persistent hash trie 81
 pixel 51
 pmap 235, 268–269
 polling 255
 polymorphism 3, 16, 139–141,
 184–185, 188–189, 192–
 194, 202, 206
 pool 251
 pop 86, 91–93, 192
 pos 101–103
 positionally 47
 postconditions 133, 173, 175,
 302
 pow 142–143
 precision 62–63
 arbitrary precision 21
 preconditions 173, 206, 302
 predicates 103, 138
 type-based 79, 102
 prefix notation 9, 25
 primitives 21, 64, 83, 200, 229,
 279, 287, 290–291
 println 30, 114, 160, 308–309
 print-method 92
 programmer efficiency 278
Programming Clojure 101
 promises 235, 261, 266–267,
 269
 callback API to blocking
 API 266
 deliver 265
 promise 149, 265–266
 with-promises 268
 protocols 15–16, 121, 189, 192,
 195, 197, 227, 258, 285,
 305
 design of 285
 Prototype Principle 183
 prototyping 214
 proxies 305
 construct-proxy 212
 proxy 38, 171, 208, 210–212,
 261, 304
 proxy-mappings 210
 proxy-super 211
 update-proxy 210, 212

pure virtual classes 192
 purely functional 131, 282
 push 192–193
 pvalues 235, 268–269
 Python (programming language) 6, 44, 126, 132

Q

Qi (programming language) 12
 queues 91, 93, 252–253
 PersistentQueue 91–92
 priority queues 192
 queue-fish 92
 quicksort 121–123
 qsort 122–123
 quote 32–34, 70, 159, 171
 quoting forms 161
 nested syntax-quotes 160
 syntax-quote 34, 163

R

range 51, 68, 83, 90, 117, 282
 rationals 22, 66
 denominator 22, 67
 numerator 22, 67
 ratio 66
 rational? 65–67
 rationalize 66–67
 read 309–310
 reader feature 28
 Read-Eval-Print Loop (REPL) 21, 26, 32, 39, 54–55, 57, 97, 117, 123, 179, 213–214, 221, 267, 270, 273, 278, 289, 309
 read-time 73
 recompiled 73
 records 189, 191, 195, 197–198, 205–206
 recur 30–31, 142, 144–147, 200
 recursion 30, 141–142
 accumulator 143
 explicit tail-call
 optimization 145
 freeing the recursive call 142
 generalized tail-call
 optimization 145
 mundane 142–143, 145, 306
 mutual 142, 145–147
 recursive call trapped 142
 stack consumption 142
 tail 30, 145
 tail position 30–31, 143, 146

tail-call optimization (TCO) 6, 145–146
 tail-recursive 124, 142, 154
 tail-recursive self-call 144
 reduce 119, 129, 180, 280, 304
 refactoring 290
 reference types 14, 108, 110, 165, 202, 235–236, 240, 242, 245, 247, 252, 255, 257–259, 261, 270, 272, 303–304
 add-watch 164, 303
 coordinated 240–244, 247
 remove-watch 303
 set-validator 241
 synchronous 241, 247, 252, 255
 uncoordinated 255
 uniform state change
 model 242
 watchers 304
 referential transparency 12, 132
 reflection 279–280, 289
 Refs 235, 240–241, 243–246, 255
 alter 242, 244–245
 commutative 245
 commute 244–245
 ensure 239, 246
 ref-set 245
 regular expressions 52, 61, 75
 case insensitivity 74
 re-find 75
 regex 73–74
 re-groups 75
 reluctant quantifiers 73
 re-matcher 75
 re-seq 74–75
 reify 38, 141, 198, 208, 211, 304
 factory function 198
 relational functions 293–294
 SELECT 10, 293
 relative-units 296, 298
 rem 55
 remainder 55
 remote-procedure call (RPC) 262, 266–267
 rename 42
 reordering 245
 replace 85
 reset 255
 resolve 72, 272
 rest 46, 78, 80, 89, 91, 115–116, 121, 127
 reusability 234

reuse 127
 reverse 41, 87
 roll 288–289
 root binding 28, 270–271
 root cause 55
 rounding error 64
 rseq 80, 84
 RSS2 262, 299
 rss-children 266, 299
 rsubseq 99
 Ruby (programming language) 16, 135, 163
 unless 163
 runtime 141

S

SafeArray protocol 258–259
 Scala (programming language) 47, 126, 145, 170, 207, 301
 scalar 61
 scalar types 75
 duality of 48, 62
 Scheme (programming language) 87, 143–145
 actors 143
 screaming-filter 211
 self 185
 semantic coupling 302
 separation of concerns 7
 seq 46, 74, 77–80, 83–84, 98, 200
 seq protocol 305
 seq1 283–284
 sequence abstraction 77–78, 80, 82, 97, 100–101, 204
 rest vs. next 116
 sequences 78–80, 290
 chunked 277, 282–283, 291
 chunk-at-a-time model 282
 sequentials 48, 78–79
 server 209
 set 37
 sets 25, 79, 94
 difference 97
 intersection 96
 relative complement 97
 union 97
 setter 204
 shared structure 112
 shared-state concurrency 234
 short 83
 shorts 279
 shuffle 226

side effects 29, 131–132, 154,
163, 180, 248, 255, 306
simplicity 4, 103, 288
simplification 204
single quote 33
sleep 246, 252
slice 228
Sliceable 229
sliceCount 228
slope 133–134
slowly 257, 287
snapshot 235–236
 isolation 236
software transactional memory
 (STM) 3, 235–238, 243,
 245–246, 248, 255
 barging 239
 commit 236, 243, 245–246
 commit-time 245
 in-transaction 243, 245–246
 retry 236
some 94
sometimes-slowly 257
sort 47, 225
sort-by 129–130
sorted-map 98–99, 113
sorted-map-by 95, 99
sorted-set 94–95, 113
sorted-set-by 95
sort-parts 123
special form 24, 26, 33
split 74–75
spot optimizations 280
spreadsheets 130, 139, 268
SQL 10–11, 293–294
sqr 302
stack overflow 115
 errors 145
stack trace 55
stacks 86, 91
 IPersistentStack 201
state 3, 13–14, 186, 202,
 234–235, 239, 242,
 245, 247, 249, 255
 managing 238
static type system 278
static versus dynamic 278
Steele, Guy Lewis 47, 143
straight-line path 150
stress 246
strings 23, 71, 73, 79
 zero-length strings 44
structural sharing 110
stubbing 299

subseq 95, 99
subvec 88–89
superclass constructor 212
Sussman, Gerald 143
Swank-Clojure 309
swap 255, 257
Swing 126, 216
Sybil 62
symbols 23, 26, 28, 70–71, 159,
 171, 178, 264
 auto-qualification 34
 symbolic mappings 179
syntax 5, 7, 9
 syntactic sugar 222
syntax-quote 32, 34–35, 72,
 160–161, 169, 171
 nested 160

T

tail position 142
 and recur targets 145
tail recursion 143
tail recursive 151
tail-call 144
take 124
terminating condition 46
test-driven development (TDD)
 109, 292, 298–301
 specification language 302
 unit tests 128
there ain't no such thing as a
 free lunch 237
third-party libraries 171, 194,
 198
this 108, 171, 185, 198, 210–211
thread-bound 272
thread-local 28, 240, 270, 273
thread-local bindings 271–273,
 306
threads 5, 28, 220, 224, 236,
 240–241, 247, 249–252,
 254–255, 258, 260–261,
 265, 267–268, 274, 282,
 308
 starved 250
thread-safe 211, 240
time 13, 235, 247
timelines 235, 243
to-array 220
to-array-2d 220
toString 204–205
trampoline 147
 trampoline for mutual
 recursion 147

transactions 235–236, 239, 241,
 243–248, 255–257
 embedded transactions 236
 retry 236, 243, 247–248, 255,
 258
 size of 247
transients 108, 277, 280–282,
 291
 Rule of Transients 280
TreeNode 189, 193, 195, 201
trees 110–111, 166, 168, 195,
 262, 283, 306
 binary 113
 persistent binary 191
 red-black 113
 traversing 111, 306–307
 unbalanced 113
triangle 118
truncation 62, 290–291
truthiness 43–44, 56
 truthy 52, 69, 94, 119
tuning 247
tweet-items 266
tweetless-rss-children 300
tweet-occurrences 264
Twitter 262–264, 299
type conversion, automatic 63
type hints 228, 277–280,
 287, 291
types 189, 227, 304–305
 deftype 38, 200–202

U

unbox 289
unchecked 64
unchecked-inc 289
underscores 181
Unicode 73
 case insensitivity 74
unit conversions 294–295
unit testing 128, 292, 298, 302
unit-of-distance 297
Universal Design Pattern
 (UDP) 183–186, 188
Unix line terminator 74
unmodifiable 226
unquote 35, 160, 163–164
unquote-splice 35, 163
up-arrow key 51
update-in 85
uppercase 74

V

validation 205
 validator 241
 variable arguments 26
 variables 14, 29, 255
 Vars 25, 27, 32–34, 40, 72, 128,
 137, 163–164, 178, 181,
 235, 240, 270, 307
 anonymous 272
 root binding 164
 var 270, 272
 var-get 272
 with-local-vars 272–273
 vectors 24, 33, 80, 82, 98, 193
 as stacks 86
 IPersistentVector 193, 198
 of names 49
 of primitives 83
 subvectors 88
 vec 83
 vector 29, 32, 83–84, 126,
 245, 295
 vector-of 83
 walking in reverse order 84
 verbosity 140, 202

verbs 13
 very bad things 259
 visible 52
 void 212
 volatile 202

W

when 8, 31, 52, 162–163, 170
 when-let 120, 171
 when-not 163
 where-am-i 72
 while 8
 Whitehead, Alfred North 61
 whitespace, ignoring 74
 wit 73
 with-binding 273
 with-open 82, 172, 274
 with-out-str 273
 with-precision 273
 with-redefs 300
 with-resource 172
 workflow 91
 world 150–152
 worst-case 81

wrapping 195
 write-once 265

X

xconj 110, 112, 191, 193,
 195–196
 xor 51
 xs 48
 xseq 191, 200

Y

Yegge, Steve 183

Z

Zawinski, Jamie (JWZ) 73
 zencat 281
 zencat2 281
 zero 44
 zipmap 98
 zipper 263