

Index

■ Symbols

- + (addition) function, 53
- ` (backquote) character, 171
- \ (backslash character), 57
- { (curly braces), 66, 71
- [] (square brackets), 64
- # character, 33, 71, 172
- #() function syntac, 145
- #^ reader macro, 129–130
- . (dot) special form, 143, 145
- @ symbol, 98, 104
- ^ reader macro, 129
- ~@ (splicing unquote), 171
- = function, 194
- == function, 56, 194

■ A

- abstraction
 - datatypes and, 181
 - macros and, 168
 - protocols and, 181
- AbstractMethodError, 183
- accessor function, 68
- aclose function, 148
- action functions, 106–107
- addition function (+), 53
- add-watch function, 112
- agent function, 105
- agent thread pools, 159–160
- agent-based processing, 1
- agent-error function, 107
- agents, 97, 159–161
 - about, 105
 - asynchronous, 105–109
 - creating and updating, 105–106
 - errors and, 107
 - failure modes, 107
 - in failed state, 107–108

- shutting down, 108
- update semantics, 106
- waiting for, 108
- when to use, 109

- aget function, 148, 195
- ahead-of-time (AOT) compilation, 152–154
- alength function, 148
- alias function, 117
- aliases, 117
- all-ns function, 124
- alter function, 100
- alter-meta! function, 131
- amap macro, 148
- ancestors function, 139
- and macro, 60
- anonymous inner classes, 185
- apply function, 93
- areduce macro, 148
- arguments
 - functions with multiple, 31
 - functions with variable, 32
- arithmetic functions, 193–195
- arities, 31–32, 182–183
- array maps, 67
- arrays
 - Java, 146–148
 - creating, 147
 - iterating over, 148
 - manipulating, 148
 - primitive, 195
- artificial intelligence, 45
- aset function, 148, 195
- aset-boolean function, 148
- aset-byte function, 148
- aset-char function, 148
- aset-double function, 148, 195
- aset-float function, 148
- aset-int function, 148, 195
- aset-long function, 148
- aset-short function, 148
- :as keyword, 26

association function (`assoc`), 66, 69, 181
 asynchronous agents, 105–109
 asynchronous updates, 97
 atom function, 104
 atoms, 97, 104–105
 attack multimethod, 134
 :author metadata, 122
 auto gensym, 172
 await function, 108
 await-for function, 108

■ B

backquote (```) character, 171
 backquote ``` reader macro, 123
 backslash character (`\`), 57
 base case, 36
 base condition, 36
 binary class name, 120
 binding form, 109–110
 binding vector, 35
 Boolean functions, 60
 Boolean values, 60–61
 boxed numbers, 193
 built-in functions, 116, 142
 built-in types, 51, 145
 bytecode, 152

■ C

cached values, 105
 char argument, 192
 char array, 151
 character coercion function (`char`), 61
 characters, 61
 characters method, 151
 CharSequence argument, 192
 Church, Alonzo, 4
 classes, 10

- anonymous inner, 185
- creating Java, 150–157
- datatypes as, 183
- extending Java, 184
- hierarchies and, 136–138
- in object-oriented languages, 133
- See also* Java classes; *specific classes*

 Classloaders, 149
 classpaths, 26

- configuring, in AOT compilation, 153
- loading namespaces from, 118–119

- loading resources from, 118

Clojure
 code, loading, 149, 168
 as dynamically typed language, 51
 environment, 17–27
 calling from Java, 148–150
 calling Java from, 143–148
 features, 1
 flexibility of, 11
 as functional language, 2–9
 immutable data structures and, 7–9
 JVM and, 2
 Lisp and, 1
 loading, 17
 as next-generation language, 1, 9
 object-oriented programming and, 9–10
 popularity of, 1
 program structure, 10–14
 starting, 17

Clojure 1.2, 179
 Clojure source code, 168
 Clojure types. *See* datatypes
 clojure.core namespace, 116, 121
 clojure.lang.Compile class, 153
 clojure.lang.Reversible interface, 182
 clojure.lang.RT class, 149
 clojure.lang.Seqable interface, 182
 clojure.set/difference function, 72
 clojure.set/intersection function, 72
 clojure.set/union function, 72
 clojure.walk library, 176
 clojure.xml namespace, 26
 CLOS (Common Lisp Object System), 11
 closures, 46
 code

- dynamically compiled, 152
- functional, 5–6
- loading and evaluating Clojure, 149, 168
- vs. data, 167

 code abstraction, 9
 code compilation, 152

- ahead-of-time, 152–154
- just-in-time, 152

 code encapsulation, pure functions and, 5
 code templating, 171
 coercions, 192
 collection functions, metadata preservation and, 128
 collections datatypes, 62–72

- lists, 63–64
- maps, 66–71

- properties of, 62
- sets, 71–72
- vectors, 64,–66
- comma character, 66
- command-line programs, 156–157
- Common Lisp Object System (CLOS), 11
- commutative functions, 101
- commute function, 14, 100–101
- comp function, 47–48
- compare function, 182
- comparison functions, 193
- Compile class, 153
- compile function, 153
- *compile-path* Var, 153
- Compiler.eval method, 149
- composing functions, 47–48
- composite forms, 19
- concat function, 87
- concurrency, 12, 159–166
 - agents and, 105, 109, 159, 161
 - futures and, 163–164
 - Java-based threading, 165–166
 - promises and, 164–165
- concurrency functions, 161–163
- cond form, 34–35
- conditional expressions, 34–35
- conditional logic, 48
- conditionals, 134
- conjoin function (conj), 65, 70, 76–77
- cons function, 76–77, 87
- constructors, adding to generated classes, 155
- :constructors option, gen-class marco, 155
- contains function (contains?), 71
- :continue mode, 107
- control structures, 173–174
- coordinated state, 97
- create-ns function, 125
- curly braces {}, 66, 71
- currying, 46–47
- cycle function, 88

■ D

- data, 51–72
 - vs. code, 167
- data structures, 168
 - immutable, 7–9
 - Java interfaces and, 145–146
 - persistence of, 7
- transients, 195–196
- datatypes, 51, 180–183
 - advanced, 186
 - built-in, 51
 - as classes, 183
 - collections, 62–72
 - lists, 63–64
 - maps, 66–71
 - properties of, 62
 - sets, 71–72
 - vectors, 64–66
 - extending protocols to preexisting, 183–184
 - in-line methods, 181
 - Java interfaces and, 145–146, 182–183
 - Java types, 51
 - reifying anonymous, 184–185
 - primitive types, 52–62
 - Boolean values, 60–61
 - characters, 61
 - keywords, 61
 - numbers, 52–57
 - strings, 57–60
 - state and, 96
 - working with, 185–186
- deadlocks, 14
- debugging, macros, 170
- decimal numbers, 53
- declare macro, 122
- decrement function (dec), 55
- :default keyword, 135
- def form, 21, 125, 129, 173
- definline form, 197
- defmacro form, 129–130, 169
- defmethod function, 133
- defmulti function, 133, 142
- defn form, 31, 129–130, 173
- defn- macro, 124, 131
- defprotocol function, 179–180
- defrecord function, 180,–185
- defstruct function, 68, 185
- deftype macro, 186
- deleting namespaces, 126
- deliver function, 165
- dependencies, order of, 22
- deref function, 98, 104
- dereferencing, 98–99
- derive function, 136, 142
- descendants function, 139
- difference function, 72
- disassociation function (dissoc), 69
- dispatch function, 133–134

- dispatch values, 133
 - default, 135
 - specifying order of, 140
- dissoc function, 181
- distinct function, 85
- division function (/), 54
- do special form, 42
- doall function, 95, 162, 190
- doc function, 31
- :doc metadata, 122
- doc-strings, 31
- Domain Specific Languages (DSLs), 2, 177–178
- dorun function, 94–95
- dosync macro, 99
- dotimes macro, 196
- dot (.) special form, 143, 145
- double-array function, 147
- double-quotes, 57
- drop function, 89
- drop-last function, 90
- drop-while function, 89
- dynamically typed languages, 51, 191
- dynamically-compiled code, 152

■ E

- Eclipse, 26
- else keyword, 34
- empty? function, 93
- encapsulation, 10
- Enclojure, 177
- encounter multimethod, 135, 138
- end function, 159
- ensure function, 101
- equality semantics, 62
- equals function (==), 56, 194
- error-handler function, 107
- error-mode function, 107
- errors, agents and, 107
- eval function, 144, 149
- every? function, 94
- :exclude option, 116
- execution tree, in functional programming, 6
- expansions, 175
- explicitly managed state, 95
- exponents, calculating, 38
- :exposes method, 156
- :exposes-methods options, gen-class macro, 156
- extend function, 183–184

- extend-protocol function, 184
- extends argument, gen-class macro, 154
- extend-type function, 184
- extract-text function, 151

■ F

- factories, adding to generated classes, 155
- :factory option, gen-class macro, 155
- :fail mode, 107
- failure state, agents in, 107–108
- file names, vs. namespace names, 118
- filter function, 85
- filters, 116
- find-ns function, 124
- first function, 73, 92
- first-class functions, 29, 43–45
 - consuming, 44–45
 - producing, 45
- flags, 119
- float-array function, 147
- floating-point decimals, 52–53
- fn special form, 29–30
- forms, 18–20
 - composite, 19
 - evaluating individual, 26
 - literals, 18
 - quoted, 168
 - special, 19
 - symbols, 19

See also specific forms
- forward declarations, 122
- fully-qualified names, 25
- function calls, lists evaluated as, 19
- function composition, 3–4, 47–48
- function definitions, 43
- functional code, 5–6
- functional programming, 2–9
 - closures, 46
 - currying, 46–47
 - first-class functions, 43–45
 - function composition, 47–48
 - immutability and, 7–9
 - imperative programming and, 4
 - program structure, 4
 - pure functions, 4–6
 - techniques, 43–48
- functions, 29–34
 - binding to symbols, 31
 - Boolean, 60

functions (*cont.*)

- built-in, 116, 142
 - collection, 128
 - communtative, 101
 - comparison, 193
 - composing, 47–48
 - concurrency, 161–163
 - currying, 46– 47
 - defined, 3
 - defining
 - with `defn`, 31
 - with `fn`, 29–30
 - dynamically generating, 45
 - first-class, 29, 43–45
 - higher-order, 44
 - list, 63
 - map, 69–71
 - of multiple arities, 31–32
 - nested, 3
 - non-pure, 5
 - numeric, 53–57
 - pure, 4–6
 - regular expression, 58–60
 - sequence, 73–95
 - sequence generator, 81
 - set, 72
 - shorthand form of declaring, 33–34
 - string, 57–58
 - symbols and, 30
 - with variable arguments, 32
 - vector, 64–66
 - See also specific functions*
- future macro, 163–164
- future? function, 164
- future-call function, 164
- future-cancel function, 164
- future-cancelled function, 164
- future-done function, 164
- futures, 163–164

■ **G**

- garbage collection, 15
- gen-class macro, 151–156, 185
- generated classes
 - adding constructors and factories, 55
 - adding methods to, 155
 - adding state to, 154
 - defining methods for, 154

- exposing superclass members, 156
- loading implementation, 156
- generic types, 146
- get function (`get`), 65, 70
- get-val function, 110
- global environment, 21–22
- global hierarchy, 136–137, 142
- global symbols, 23–24
- global variables, 24
- greater-than function (`<`), 56
- greater-than-or-equals function (`<=`), 56

■ **H**

- hash maps, 8, 67
- hashed sets, 8
- Haskell, 6
- Hello World program, 17–21
- hexadecimal notation, 52
- Hickey, Rich, 11
- hierarchies
 - about, 136
 - conflict resolution and, 139–140
 - global hierarchy, 142
 - independent, 141–142
 - inheritance and, 136
 - with Java classes, 138
 - with multimethods, 137–141
 - querying, 137–139
 - user-defined, 141–142
- `:hierarchy` argument, 142
- higher-order functions, 44
- homoiconicity, 167–168
- Hotspot, 189
- hot-swapping, 126
- hyphens (`-`), 118

■ **I**

- identities
 - independent, 97
 - keeping track of, 111–113
 - synchronous vs. asynchronous updates, 97
 - state and, 96– 97
 - updates to, 97
- if form, 34
- if-not form, 34
- immutability, 7–9
- immutable data structures, 7–9
- imperative languages, 2–3

- imperative programming, 3–4
- :implements argument, gen-class macro, 154
- :impl-ns argument, gen-class macro, 154
- import function, 120
- Incanter, 177
- increment function (inc), 55, 162
- independent hierarchies, 141–142
- independent state, 97
- indexes, 64
- infinite sequences, 81
- infix macro, 171
- inheritance, 9
 - hierarchies and, 136
 - multiple, 133, 136
 - protocols/datatypes and, 180, 185
- :init function, 155
- in-line methods, 181
- inlining, 197
- inner classes, 120
- in-ns function, 115–116, 121, 124–125
- int-array function, 147
- integers, 52–53
- interface injection, 184
- interfaces
 - extending, 184
 - protocols as, 180
- interleave function, 88
- intern function, 125
- interning Vars, 125
- interpose function, 88
- intersection function, 72
- into-array function, 147
- invert multimethod, 139
- IS-A relationships, 133
- isa? function, 137–138
- iterate function, 84
- iteration, over arrays, 148

■ J

- Java
 - calling Clojure from, 148–150
 - calling from Clojure, 143–148
 - libraries, 38, 143
 - objects, 7
 - profiling tools, 190
 - types, 51
- Java API, 148
- Java arrays, 146–148
 - creating, 147

- iterating over, 148
- manipulating, 148
- as sequences, 75
- Java bytecode, 152
- Java classes
 - creating, 150–157
 - extending, 184
 - generating command-line programs, 156
 - hierarchies with, 138
 - importing, 120
 - loading implementation, 156
 - proxying, 150–151
- Java Classloaders, 149
- Java collections, as sequences, 75
- Java generics, 146
- Java interfaces
 - Clojure types and, 145–146
 - extending, 182–184
- Java interoperability, 143–157
 - calling Clojure from Java, 148–150
 - calling Java from Clojure, 143–148
 - convenience forms, 144–145
 - special forms, 143–144
- Java Reflection API, 144
- Java Virtual Machine (JVM), 2, 152
 - performance tips, 189
 - profiling on, 189–190
- java.lang.IndexOutOfBoundsException exception, 64
- java.lang.Object, 182
- java.lang.Runnable interface, 165
- java.lang.String class, 57
- java.lang.Thread object, 165
- java.math.BigDecimal class, 53
- java.util.Collections framework, 62
- java.util.concurrent.atomic package, 104
- Java-based threading, 165–166
- just-in-time compilation, 152

■ K

- keys function (keys), 71, 83
- key-value pairs, 66
- keyword function (keyword), 61, 123
- keyword test function (keyword?), 62
- keywords, 61
 - constructing, 123
 - as map keys, 67
 - namespaced, 61, 122–134

■ **L**

last function, 92
 lazy sequences, 77–83, 190
 constructing, 80–81
 example, 78–80
 memory management and, 82–83
 lazy-cat macro, 87
 lazy-seq macro, 80–81, 84
 less-than function (>), 56
 less-than-or-equals function (>=), 56
 let form, 35–36
 libraries, 143
 libspec argument, 119
 libspecs, 119
 linked lists, 8, 63
 Lisp programming language, 1, 19
 list function (list), 63, 169
 list literals, declaring, 63
 list test function (list?), 64
 lists, 19, 63–64
 as data structures, 63
 constructing recursively, 77
 linked, 63
 literal forms, 129
 literals, 18, 63
 load function, 118, 149
 load-file function, 117, 121
 :load-impl-ns false option, gen-class macro,
 156
 local bindings, 35–36
 local symbols, defining in macro, 172
 local-name argument, 117
 locking policies, 11
 locks, 95, 99
 long-array function, 147
 lookups, Var, 196
 loop primitives, 193–194
 loop special form, 41–42
 looping, 36–42

■ **M**

macroexpand function, 170
 macroexpand-1 function, 170
 macros, 168–178
 about, 168–169
 code templating, 171
 creating, 169
 creating DSLs using, 177–178

 debugging, 170
 generating symbols, 172
 implementing control structures, 173–174
 implementing, using recursion, 176
 implementing, with variadic arguments,
 174–175
 inlining and, 197
 reader, 33
 splicing unquotes, 171
 using, 173–176
 when to use, 173
 working with, 169–170
 -main function, 157
 :main true option, gen-class macro, 156
 make-array function, 147
 make-hierarchy function, 141
 map association function (assoc), 69
 map disassociation function (dissoc), 69
 map functions, 69–71, 78–80, 92
 map keys, 67–68
 map keys function (keys), 71
 map merge function (merge), 70
 map test function (map?), 71
 mapcat function, 87
 mappings, removing from namespace, 125
 maps, 66–71, 180–181
 array, 67
 hash, 67
 key-value pairs, 66
 method, 183
 as objects, 69–71
 sorted, 67
 struct, 68–69
 mathematic operations, 53–57
 maximum function (max), 55
 memfn macro, 145
 memoization, 191
 memoize function, 105
 memory
 access to, 11
 management, 82–83
 software transactional memory (STM), 12–
 15
 merge function (merge), 70
 merge-with function (merge-with), 70
 meta function, 127
 metadata, 127–131
 defined, 127
 metadata-perserving operations, 128
 namespace, 122
 reading and writing, 127–128

- metadata (*cont.*)
 - read-time, 129
 - on reference types, 131
 - on Vars, 129–131
- metalanguage, 167
- metaprogramming
 - about, 167–168
 - code vs. data, 167
 - homoiconicity and, 167–168
- method implementations, 136
- method-one function, 179
- methods
 - adding to generated classes, 155
 - overloaded, 182
- methods maps, 183
- :methods option, gen-class macro, 155
- method-two function, 179
- microbenchmarks, 189
- minimum function (min), 55
- modularity, 9
- modularization, 9
- modulus function, 55
- move multimethod, 134, 141
- multimethods, 9, 35, 133–142, 185
 - about, 133–136
 - conflict resolution and, 139–140
 - default dispatch values, 135–136
 - global hierarchy and, 142
 - hierarchies with, 137–141
 - multiple dispatch, 135
- multiple dispatch, 133–135
- multiple inheritance, 133, 136
- multiple-arity methods, 150
- multiplication function (*), 46–47, 54
- multithreaded programs, 11
- my-ref Var, 98

■ N

- name argument, 144, 149
- :name argument, gen-class macro, 154
- name function, 123
- names
 - fully-qualified, 25
 - namespace names vs. file names, 118
- namespace function, 122–123
- namespace-name argument, 117
- namespace-qualified keywords, 61, 122–123, 134
- namespace-qualified symbols, 122–123

- namespaces, 24–26
 - about, 115
 - advanced operations, 124–126
 - basics of, 115–117
 - common prefixes, 119
 - declaring, 25–26, 115, 121, 156
 - deleting, 126
 - importing Java classes, 120
 - loading, 117–120
 - from file or stream, 117
 - from the classpath, 118–119
 - in one step, 120
 - manipulating, 125–126
 - metadata, 122
 - names, vs. file names, 118
 - naming conventions, 26
 - querying, 124–125
 - as references, 126
 - referring
 - in one step, 120
 - to other, 116–117
 - removing mapping from, 125
 - switching, with in-ns, 115–116
 - symbols and, 121–124
- namespacing mechanism, 9
- negative test function (neg?), 57
- nested Java classes, 120
- NET Common Language Runtime, 146
- Netbeans, 26
- new special form, 143–145
- next function, 88
- nil, 52
- non-pure functions, 5
- not function (not), 60
- ns macro, 121, 125–156
- ns-aliases function, 124
- ns-imports function, 125
- ns-map function, 124
- ns-name function, 124
- ns-publics function, 124
- ns-refers function, 125
- ns-resolve function, 125
- ns-unmap function, 125
- *ns* Var, 124
- nth function, 79, 92
- number test function (number?), 57
- numbers, 52–57
- numeric functions, 53–57
- numeric literals, 52, 194

■ O

object-oriented languages, 133, 180
 object-oriented programming (OOP), 9–10
 object-oriented programs, 10
 objects, 5
 first-class, 29
 maps as, 69–71
 octal notation, 52
On Computable Numbers (Turing), 4
 :only option, 116, 120
 or macro, 61
 overloaded methods, 182

■ P

parallel processing, 1, 5
 parallel programming, 159–166
 parallelism, 159, 162
 parents function, 139
 parse method, 151
 partial function, 46–47
 partition function, 91
 patterns, in code, 173
 pcalls function, 161–162
 peek function (peek), 63–65
 performance
 concurrency functions and, 162–163
 inlining and, 197
 macros and, 173
 memoization and, 191
 primitives and, 193
 reflection and, 191–193
 STM and, 14
 tips for Java, 189
 transients and, 195–196
 type hints and, 191–193
 Var lookups and, 196
 performance, 189–198
 periods (.), 118, 144
 Perlis, Alan, 10
 persistence, 7
 persistent collections, as sequences, 75
 persistent! function, 196
 pmap function, 161–163
 polymorphic functions, 179
 polymorphism, 9, 133
 pop function (pop), 64–66
 positive test function (pos?), 57, 89
 :post-init function, 155

pound sign (#), 33, 71, 172
 predicate functions, 89, 194
 prefer-method function, 140
 :prefix argument, gen-class macro, 154
 prefix lists, 119
 primitive arrays, 195
 primitive types, 52–62
 Boolean values, 60–61
 characters, 61
 keywords, 61
 loop primitives, 193–194
 numbers, 52–57
 strings, 57–60
 primitive-aware functions, 193
 print-contacts function, 103
 printing functions (print & println), 42, 58
 print-meta, 127
 :private metadata, 124
 private Vars, 123–124, 131
 profiling, on JVM, 189–190
 program flow, controlling, 29–49
 program state. *See* state
 promise function, 165
 promises, 164–165
 protocols, 179–181
 for preexisting datatypes, 183–184
 working with, 185–186
 proxy classes, Java, 150–151
 proxy macro, 150–151, 185
 proxy methods, 151
 proxy-handler function, 151
 proxy-super macro, 151
 public symbols, 116, 125
 public Vars, 123–124
 pure functions, 4–6, 10
 pvalues function, 161–162

■ Q

querying hierarchies, 137–139
 querying namespaces, 124–125
 quote form, 63
 quoted forms, 168
 quotient function (quot), 55

■ R

range function, 85
 -rangechecker function, 46
 ratios, 52

reader macros, 33
 read-string function, 149
 read-time metadata, 129
 recur form, 39–40
 recursion, 3, 36–42

- guidelines for, 36
- implementing macros using, 176
- tail, 39–42
- using cons or conj functions, 77
- using loop, 41–42

 recursive macros, 176
 reduce function, 93
 ref function, 98
 :refer-clojure form, 121
 refer function, 116–117, 120
 reference types

- metadata on, 131
- of identities, 96

 references, namespaces as, 126
 re-find function, 59
 reflection, 191–193
 refs, 97

- coordinated access to multiple, 103
- creating and accessing, 98
- defined, 97
- updating, 98–104
 - examples, 101–104
 - tools for, 99–101

 ref-set function, 99–100
 regex pattern, 59
 re-groups function, 60
 regular expression functions, 58–60
 reify macro, 184–186
 :reload flag, 119
 :reload-all flag, 119
 remainder function (rem), 55
 re-matcher function, 59
 re-matches function, 59
 remove function, 86
 remove-method function, 140
 remove-ns function, 126
 remove-watch function, 113
 re-pattern function, 59
 repeat function, 84
 repeatedly function, 84
 REPL (Read Evaluate Print Loop), 17–18
 require function, 119
 :require keyword, 26
 re-seq function, 60
 reset! function, 105
 resolve function, 125

rest function, 73, 88
 restart-agent function, 108
 return values, 196
 reusability, 5, 10
 reverse function, 90
 Reversible interface, 182
 root binding, 149
 rseq function, 83, 182
 RT class, 149
 RT.load method, 149
 RT.loadResourceScript method, 149
 RT.maybeLoadResourceScript method, 149
 RT.readString method, 149
 RT.var method, 149
 running source files, 20–21
 runtime polymorphic dispatch, 133

■ S

scope, symbol, 24
 second function, 92
 send function, 106, 160
 send-off function, 106, 159–160
 Seqable interface, 182
 seq function, 32, 75, 83, 182
 sequence API, 83–95
 sequence functions, 128
 sequence generator functions, 81
 sequences, 73–95

- constructing, 76–77
- creating, 83–95
- infinite, 81
- introduction to, 73–75
- lazy, 77–83
- sequencable types, 75
- structure of, 75–76

 set difference function, 72
 set functions, 72
 set intersection function, 72
 set union function, 72
 set! function, 110, 144
 set-error-handler! function, 107
 sets, 8, 71–72
 setter functions, 148
 set-val function, 110
 set-validator! function, 111
 shorthand functions, 33–34
 shutdown-agents function, 108
 side effects, 4–6, 42–43

- single quote character, 63, 168
- software transactional memory (STM), 1, 12–15, 97
- some function, 94
- sort function, 90
- sort-by function, 90
- sorted maps, 8, 67
- sorted sets, 8
- source code, 168
- source files
 - structuring, 26
 - writing and running, 20–21
- special forms, 19
- splicing unquote, 171
- split-at function, 91
- split-wth function, 91
- sqrt function, 38
- square brackets [], 64
- square function, 79
- square roots, 37, 41
- stack size, 39
- StackOverflowError, 81
- start() method, 165
- state, 5–6
 - adding, to generated classes, 154
 - coordinated vs. independent, 97
 - eliminating, 95
 - failure, 107–108
 - identity and, 12–13, 96–97
 - reality of, 95
 - synchronous vs. asynchronous updates, 97
 - thread-local, 109–111
 - vars, 109–111
- :state argument, 154
- state effects, 4
- state management, 11–12, 95–113, 159
 - asynchronous agents, 105–109
 - atoms, 104–105
 - explicit, 95
 - refs, 97–104
 - validators, 111–112
 - watches, 112–113
- static methods, 145, 155
- statically typed languages, 191
- string concatenation function (str), 58
- string functions, 57–58
- string printing functions (print & println), 58
- string test function (string?), 58
- String.replace method, 192

- strings, 57–60, 75
- struct maps, 68–69
- struct-map function, 68
- StructMaps, 180
- stub methods, 154
- subroutines, 3
- substring function (subs), 58
- subtraction function (–), 54
- sub-vector function (subvec), 66
- superclass members, exposing, 156
- superclass methods, 151
- swap! function, 104
- symbol bindings, 110
- symbol function, 123
- symbol resolution, 23–24
- symbols, 19, 23–24, 27
 - binding functions to, 31
 - constructing, 123
 - defining within namespace, 25
 - forward declarations and, 122
 - functions and, 30
 - generating, 172
 - global, 24
 - names, 23
 - namespaces and, 121–124
 - public, 116, 125
 - redefinition of, 23
 - scope of, 24
 - unqualified, 122
- synchronous updates, 97
- synchronous, coordinated identities, 97
- syntax quoting, 171
- syntax-quote character, 171

■ T

- :tag keyword, 191
- :tag metadata key, 131
- tags, 136
- tail position, 39
- tail recursion, 39–42
- tail-call optimization, 39–42
- take function, 89
- take-nth function, 89
- take-while function, 90
- target argument, 143
- the-ns function, 124
- this argument, 182
- thread pools, 108, 159–160
- threading, Java-based, 165–166

thread-local state, 109–111
 thread-local var bindings, 109
 threads, creating in Java, 165–166
 time macro, 161, 190
 to-array function, 147
 transactional behavior, 101–104
 transactions, 14, 97–100
 transient function, 196
 transients, 195–196
 Turing Machine, 4
 Turing, Alan, 4
 type argument, 147
 type coercions, 192
 type function, 141
 type hierarchies, 136
 type hints, 131, 185, 191–193
 :type metadata, 141
 type tags, 131, 141

■ U

unchecked arithmetic functions, 194–195
 unchecked-add function, 194
 unchecked-dec function, 194
 unchecked-divide function, 194
 unchecked-inc function, 194
 unchecked-multiply function, 194
 unchecked-negate function, 194
 unchecked-remainder function, 194
 unchecked-subtract function, 194
 union function, 72
 unit testing, pure functions and, 6
 unqualified symbols, 122
Unsolvable Problem of Elementary Number Theory, An (Church), 4
 updates, semantics of, 106
 use function, 120
 :use parameter, 25
 user namespace, 25
 user-defined hierarchies, 141–142

■ V

validators, 111–112
 vals function (vals), 71, 83
 Var.invoke function, 150
 variable arity, 32
 variables, 23–24
 variadic arguments, 174–175
 var-name, 21
 Vars, 21–24, 27, 109–111
 evaluating, 25
 interning, 125
 Java and, 149
 lookups, 196
 metadata on, 129–131
 private, 123–124, 131
 public, 123–124
 root binding of, 149
 var-value, 21
 vary-meta function, 127
 vector association function (assoc), 66
 vector conversion function (vec), 65
 vector creation function (vector), 64
 vector functions, 64–66
 vector test function (vector?), 65
 vectors, 8, 30, 64–66
 :verbose flag, 119

■ W

watches, 112–113
 with-meta function, 127

■ X

xml macro, 177–178

■ Z

zero test function (zero?), 56