

Symbols

- " (double quote)
 - not interchangeable with single quote, 18
 - string literals, 19
- % (modulus) for integer types, 21
 - not floating-point types, 23
- & (ampersand) as pointer address operator, 121
- ' (single quote)
 - not interchangeable with double quote, 18
 - rune literals, 18
- * (asterisk)
 - pointer indirection operator, 121
 - pointer type declaration, 121
- (dash) for ignored field in marshaling or unmarshaling, 328
- ... operator
 - slice appended onto another, 41
 - variadic input parameter, 95
- . prefix for hidden file names, 277
- ./..., 5
- 0b (binary prefix), 18
- 0o (octal prefix), 18
 - 0 alone should not be used, 18
- 0x (hexadecimal prefix), 18
- := versus var, 28-30
 - shadowing variables, 68-71
- ;(semicolon) insertion rule, 6
- <- for channel access, 291
- = (equal sign)
 - := versus var, 28-30
 - arithmetic operators, 22
 - comparison operators
 - arrays, 38
 - comparable interface, 186, 196
 - complex types, 24
 - floating-point types, 23
 - integer types, 22
 - interfaces comparable, 165
 - maps not comparable, 56
 - slices not comparable, 40
 - string types, 25
 - structs comparable if fields comparable, 64
 - time.Time methods, 325, 326
- @latest
 - installing third-party tools, 264
 - updating a tool, 266
 - retracted version not matching, 255
- @version
 - installing third-party tools, 264
 - updating a tool, 266
- [] (brackets) for slices versus arrays, 39
- _ (underscore)
 - blank imports, 239
 - first iota value when ignored, 154
 - floating-point literals, 18
 - hidden file name prefix, 277
 - integer literals, 18
 - return value ignored, 78, 97
 - return value nameless among named, 98
 - targeted code build tags, 284
 - unused variable and constant names, 77
 - variable and constant names not using, 34
- ` (backquote)
 - raw string literals, 19
 - struct tags, 328
- { } (braces)
 - defining a block, 4, 67

- empty interface, 166
- not fixed by go fmt, 6
- opening brace placement, 5
- ~ (tilde) for types with underlying type, 191

A

- abstract types, 144
 - interfaces, 157
- addresses
 - pointers pointing to, 120
 - variable storage in memory, 119
- aliases, 238
- alice module for http.Server, 342
- Amdahl, Gene, 288
- Amdahl's law, 288
- ampersand (&) as pointer address operator, 121
- anonymous functions, 103
 - as closures, 105-109
- anonymous structs, 63
 - comparing structs, 65
 - unmarshaling and marshaling data, 64
- Antinyan, Vard, 114
- any as type alias for empty interface, 167
 - comparable interface, 166, 186, 196
 - interfaces as type constraints, 188
 - generic function creating a stack, 185
- APIs
 - concurrency-free, 297
 - database/sql/driver package evolution, 163, 171
 - exported identifiers in package API, 227
 - internal packages not part of, 234
 - language servers, 8
 - log/slog package documentation URL, 346
 - maps for input parameters or return values, 131
 - optional interfaces for evolving APIs, 171
 - panics must not escape public APIs, 220
 - renaming and reorganizing, 238
 - sentinel errors a part of public API, 206
 - testing your public API, 377
 - type assertions and type switches, 170
- APL (A Programming Language) and iota, 152
- append function
 - not assigning return value as error, 41
 - slices grown via, 41
 - capacity of slices, 42

- full slice expression protecting subslices, 48
- slice for input parameter then using append, 132
- arithmetic operators
 - complex types, 24
 - dividing by zero
 - floating-point types, 23
 - integer types, 22
- floating-point types, 23
- integer types, 21
 - bit manipulation operators, 22
 - combined with =, 22
 - comparison operators, 22
 - Go specification URL, 22
- type conversion, 26
- array literals, 37
- arrays, 37-39
 - converting to slices, 50
 - declaration, 37
 - function input parameters, 134
 - len function for length, 38
 - multidimensional arrays simulated, 38
 - reading and writing, 38
 - size affecting array type, 38
 - converting array to slice to array, 52
 - slices contrasted, 39
 - stack and, 136
 - slices converted to, 51
- The Art of Concurrency (Breshears), 288
- Artifactory proxy server support, 259
- asterisk (*)
 - pointer indirection operator, 121
 - pointer type declaration, 121
- Athens Project open source proxy server, 259
 - authentication configuration documentation URL, 260
- atomic variables, 317
- automatic type promotion, 26

B

- B suffix, 140
- backpressure in buffered channels, 302
- backquote (`)
 - raw string literals, 19
 - struct tags, 328
- backslash rune literal ('\'), 18
- backslash-escaped rune literals, 18

- backward compatibility of Go, [xvii](#)
 - Go Compatibility Promise, [13](#), [163](#)
- Bazaar for version control, [252](#)
- benchmarks, [395](#)
- Bendersky, Eli, [200](#), [237](#)
- Bhargava, Aditya, [57](#)
- binary prefix (0b), [18](#)
 - uses for binary, [18](#)
- binary trees
 - generic functions with generic data structures, [194-196](#)
 - generic types for, [182](#)
 - nil values for value receiver, [148](#)
- bit manipulation operators, [22](#)
- blank imports, [239](#)
 - embed package import, [275](#)
- blank returns never should be used, [99](#)
- blank switches, [87](#)
- blocks, [67](#)
 - file block, [67](#)
 - package blocks, [30](#), [67](#)
 - universe block, [70](#)
- book supplemental material URL, [xix](#)
 - code repository
 - Chapter 1, [13-14](#)
 - Chapter 2, [24](#), [27](#), [30](#), [33-35](#)
 - Chapter 3, [43](#), [46-50](#), [52-54](#), [57](#), [60](#), [65](#)
 - Chapter 4, [68-71](#), [74](#), [77-82](#), [84](#), [86-87](#), [89-90](#), [92](#)
 - Chapter 5, [94-95](#), [97](#), [99](#), [101](#), [104-105](#), [107-109](#), [111](#), [113-116](#)
 - Chapter 6, [123](#), [130](#), [141](#)
 - Chapter 7, [146](#), [149](#), [156](#), [158](#), [164](#), [166](#), [168](#), [172](#), [177-178](#)
 - Chapter 8, [183](#), [186](#), [188](#), [190-193](#), [196-197](#), [200-201](#)
 - Chapter 9, [204-206](#), [208-211](#), [213-214](#), [216](#), [218-219](#), [221](#)
 - Chapter 10, [225](#), [231](#), [235](#)
 - Chapter 11, [264](#), [268-271](#), [274](#), [276-277](#), [280-281](#), [286](#)
 - Chapter 12, [291](#), [295-296](#), [300-308](#), [313](#), [315](#), [317](#)
 - Chapter 13, [322](#), [331-333](#), [335](#), [337-338](#), [340](#), [341](#), [343](#), [346](#), [347](#)
 - Chapter 14, [352](#), [355](#), [358](#), [361](#), [363](#), [365-366](#), [368](#)
 - Chapter 15, [371](#), [377-378](#), [380](#), [383-384](#), [393](#), [397](#), [399](#), [402-406](#)
 - Chapter 16, [413](#), [417-418](#), [423](#), [425](#), [428-429](#), [433-436](#), [438](#)
 - goroutine_for_loop repository, [298](#)
 - money repository, [241](#)
 - package examples, [227](#)
- bool type, [19](#)
 - explicit type conversion and booleans, [27](#)
 - zero value as false, [19](#)
- braces ({ })
 - defining a block, [4](#), [67](#)
 - empty interface, [166](#)
 - not fixed by go fmt, [6](#)
 - opening brace placement, [5](#)
- brackets ([]) for slices versus arrays, [39](#)
- Branson, Rick, [139](#)
- break keyword
 - for statement, [75](#)
 - switch statement, [86](#)
- Breshears, Clay, [288](#)
- brew (see Homebrew installation of Go)
- BSD installation of Go, [1](#)
 - updating Go tools, [14](#)
- buffers for reading data via slices, [135](#)
 - reducing garbage collector's workload, [135](#), [136](#)
- bufio package
 - bufio.NewReader, [171](#)
 - bufio.Reader, [171](#)
 - *os.File instance with Scanner type, [324](#)
- bug detection via go vet, [7](#)
 - (see also testing)
- build tags, [284](#)
 - integration tests and, [405](#)
- builds
 - before building
 - go fmt, [6](#)
 - go vet, [7](#)
 - binaries for other platforms, [283](#)
 - go build, [4](#)
 - go generate, [278-280](#)
 - go install, [264-266](#)
 - go run to build and execute, [263](#)
 - Hello World, [3-7](#)
 - Makefiles, [12](#)
 - semicolons added by compiler, [6](#)
 - single native binary, [2](#), [240](#)
 - Go runtime compiled into, [42](#)
 - Go tool updates and, [14](#)
 - reading build info inside, [281](#)

- byte for uint8, 20, 26
 - for statements iterating over strings, 80
 - strings as sequences of, 52-55
 - string bytes versus UTF-8 multi-byte code points, 53
 - type conversions to strings and runes, 54
- C**
- C library integration
 - C pseudopackage, 434-438
 - cgo, 433-438
- calculator example code, 101
- call by value language, 114-116, 125
 - map passed to a function, 131
 - pointer passed to a function, 125-127
- call-by-value language, 41
- camelCase instead of snake_case, 34
- cancellation, 358-363
 - in your own code, 367
- cap function
 - channel buffer capacity, 292
 - slice capacity, 43
 - example code, 43
- capacity of slices, 42
 - never less than length, 44
- capitalization determining export of package-level identifiers, 227
- case clause, 84-87
- cat simplified example code, 109, 113
- cgo package for C library integration, 433-438
 - cgo.Handle, 436
 - cgo.NewHandle, 436
- chan keyword, 291
- channels in concurrency
 - about, 291
 - passed as pointer to functions, 291
 - backpressure in buffered channels, 302
 - behavior chart, 293
 - closing, 292
 - selecting a closed channel, 304
 - zero value when read while closed, 293
 - mutexes used instead, 313-316
 - critical section, 313
 - passed via pointer, 316
 - reading, writing, and buffering, 291
 - read once, 291
 - reading via for-range loop, 292
 - zero value and comma ok idiom, 293
 - when to use buffered versus unbuffered, 301
 - zero value as nil, 291
- characters via rune type, 26
- checksum database, 259
- Cheney, Dave, 205, 207
- chi request router, 342
- Chocolatey
 - Go installation, 1
 - make installation, 13
 - updating Go tools, 14
- circular dependencies, 235
- clear function
 - map length set to zero, 59
 - slice elements to zero value, 44
- Client type, 336-343
- close function to close a channel, 292
- Close method of file via defer, 110
- closures, 105-109
 - passing functions as parameters, 107
 - returning functions from functions, 108
- code
 - book code repository (see book supplemental material URL)
 - documented with Go Doc comments, 231-234
 - error detection via go vet, 7
 - (see also testing)
 - formatting, 5
 - go generate, 278-280
 - Hello World, 4
 - importance of error checking, 103
 - modules, 3, 223
 - (see also modules)
 - packages, 4
 - (see also packages)
 - third-party code imported, 240
 - timing out in concurrency, 304
- code points
 - string bytes versus UTF-8 multi-byte code points, 53
 - Unicode using four bytes for each, 55
- collaboration
 - code formatting for, 5, 32
 - Go Playground Share button, 10
 - unused variables, 32
- comma ok idiom
 - channel returning zero value, 293
 - handling wrong type assertion, 168, 169
 - map key 0 or not in map, 58

- zero value versus no value indication, 131
- comments in Go Doc format, 231-234
- Communicating Sequential Processes (CSP), 287
- comparable interface, 166, 186, 196
 - interfaces as type constraints, 188
- comparison operators
 - arrays, 38
 - boolean types via, 27
 - comparable interface, 166, 186, 196
 - complex types, 24
 - floating-point types, 23
 - integer types, 22
 - interfaces comparable, 165, 196
 - maps not comparable, 56
 - maps.Equal, 59
 - maps.EqualFunc, 59
 - slices not comparable, 40
 - DeepEqual in reflect package, 40, 411
 - slices.Equal, 40
 - slices.EqualFunc, 40
 - strings, 25
 - structs comparable if fields comparable, 64
 - anonymous structs, 65
 - test results compared with go-cmp, 378-380
 - time.Time methods, 326
- compiling
 - before compiling
 - go fmt, 6
 - go vet, 7
 - cross-compiling, 283
 - go build, 4
 - go install, 264-266
 - go run to build and execute, 263
 - Makefiles, 12
 - semicolons added by compiler, 6
 - single native binary, 2, 240
 - Go runtime compiled into, 42
 - Go tool updates and, 14
 - reading build info inside, 281
- complex types, 23-25
 - arithmetic operators, 24
 - complex function to declare, 24
 - extracting real and imaginary portions, 24
 - inexactness of, 24
 - why supported, 25
 - zero value as 0, 24
- composite types
 - arrays, 37-39
 - maps, 56-61
 - slices, 39-50
 - strings, runes, bytes, 52-55
 - structs, 61
- composition and promotion, 154
- concrete types, 144
 - examples, 143
 - function return values, 162
 - interface implicit implementation, 158
 - type assertions, 167
 - uses for, 170-172
- concurrency
 - about, 287
 - atomic variables, 317
 - channels
 - about, 291
 - backpressure in buffered channels, 302
 - behavior chart, 293
 - closing, 292, 304
 - mutexes used instead, 313-316
 - reading via for-range loop, 292
 - reading, writing, and buffering, 291
 - using buffered versus unbuffered, 301
 - zero value as nil, 291
 - data race detector, 406
 - goroutines, 289
 - channels for communication, 291
 - launching, 290, 299
 - scheduler, 289
 - terminating, 300, 301, 358-363, 367
- practices and patterns
 - always clean up goroutines, 299
 - APIs concurrency-free, 297
 - backpressure, 302
 - buffered versus unbuffered channels, 301
 - context to terminate goroutines, 300
 - for loop index variable, 298-299
 - putting the tools together, 309-313
 - running code exactly once, 308-309
 - select case turned off, 304
 - timing out code, 304
 - waiting for goroutines to finish, 306-307
- select keyword, 294-297
 - case turned off, 304
 - default clause, 296
 - for loop for multiple channels, 296
- sync.Map type, 316
- when to use, 287

- Concurrency in Go (Cox-Buday), 315, 317
- const keyword, 30-32
 - const block for iota set of values, 152
 - constants
 - generic type assignments, 193
 - immutability of, 31
 - io.Seeker whence constants, 323
 - as sentinel errors, 207
 - unused constants, 33
 - naming constants, 33
 - camelCase instead of snake_case, 34
 - uppercase names not used, 34
 - typed and untyped constants, 32
 - untyped constant type conversion, 29
- containers with Go app image, 2
 - blog post resource, 2
- content embedded into program, 274-277
- context package
 - about metadata problem, 349
 - cancellation, 301, 358-363
 - in your own code, 367
 - context.Background, 350
 - context.CancelFunc, 358
 - context.Cause, 361
 - context.TODO, 350
 - context.WithCancel, 358
 - context.WithDeadline, 364
 - context.WithTimeout, 364
 - context.WithValue, 352
 - contexts with deadlines, 363-367
 - description of context, 349-352
 - first parameter passed, 349
 - terminating goroutines, 300
 - context cancellation, 301, 358-363
 - Value method, 352
 - key, 352
 - values, 352-358
- continue keyword, 75
- control structures
 - choosing between if and switch, 89
 - for statement, 72-84
 - about, 72
 - break statement, 75
 - choosing the right statement, 83
 - complete for statement, 72
 - condition-only for statement, 73
 - continue statement, 75
 - for-range statement, 76-82
 - for-select loop, 296
 - goroutines and for loop index, 298-299
 - infinite for statement, 74
 - labeling, 82, 86
 - nested, 82
 - shadowing variables, 73
 - single-letter variable names, 35
 - goto generally not used, 89-92
 - if/else if/else, 71
 - switch statement, 84-87
 - blank switches, 87
- copy function for slices, 49
- Cox, Russ, 5, 13, 184, 238
- Cox-Buday, Katherine, 315, 317
- cross-compiling, 283
- csv package
 - csv.NewReader, 417
 - csv.NewWriter, 417
- Ctrl-C to stop execution, 75

D

- dash (-) for ignored field in marshaling or unmarshaling, 328
- data embedded into program, 274-277
- data race detector, 406
 - documentation URL, 408
- database/sql/driver package
 - database support, 112
 - evolution of, 163, 171
 - interfaces defined for database driver, 163
- deadlock, 294
- Dean, Jeff, 138
- decimal module (ShopSpring), 241
- declaration list for variables, 28
- decorator pattern with standard interfaces, 161
- DeepEqual in reflect package, 40, 411
- default keyword, 84-87
- default type, 19
- defer keyword, 109-114
 - actions taken on returned error, 112
 - function, method, or closure with, 111
 - multiple functions, 111
 - io.Closer Close function called via, 323
 - loops and, 323
 - Mutex Unlock called after Lock or RLock, 315
 - named return values, 112
 - returned values cannot be read, 112
 - wrapping errors with, 217

- delete function for map key-value, 59
- Delve debugger, 8
- dependencies
 - circular dependencies, 235
 - module versions
 - dependency copies inside module, 250
 - resolving different module version dependencies, 247
 - updating to compatible versions, 248
 - updating to incompatible versions, 248, 253
 - vendor, 250
 - overriding, 254
 - require section of go.mod, 226, 242
- dependency injection via implicit interfaces, 174-178
 - Wire helper instead, 178
- dereferencing, 121
 - non-nil pointers only, 121
- Design Patterns (Gamma, Helm, Johnson, and Vlissides), 154, 158
- development environment
 - go commands, 2
 - go build, 4
 - go fmt, 5
 - go mod, 3
 - go version, 2
 - installing Go tools, 1
 - go command, 2
 - troubleshooting installation, 2
 - updating Go tools, 14
 - validating with go version, 2
- Makefiles, 12
- project organization, 3
- tools
 - about, 8
 - Go development tools URL, 8
 - The Go Playground, 10-11
 - GoLand, 9
 - Visual Studio Code, 8
 - updating, 14
- Dijkstra, Edsger, 89
- dividing by zero
 - floating-point types, 23
 - integer types, 22
- do loop via infinite for loop, 75
- Docker containers with Go app image, 2
 - blog post resource, 2
- Dogan, Jaana, 265

- double quote (")
 - not interchangeable with single quote, 18
 - string literals, 19
- duck typing, 158

E

- Echo web framework, 342
- Effective Go
 - idiomatic Go resource, 7
 - semicolon insertion rule, 6
- else keyword, 71
 - variables scoped to if and else blocks, 71
- embed package, 275
- embedding an interface in an interface, 162
- embedding content into program, 274-277
 - hidden files, 277
 - virtual filesystem, 275
- embedding for composition, 154
 - embedded field, 155
 - embedding is not inheritance, 156
- encoding/json package, 327-335
 - custom JSON parsing, 332-335
 - JSON
 - about, 327
 - marshaling versus unmarshaling, 327
 - struct tags to add metadata, 327
 - zero value versus no value via pointers, 131
 - json.Decoder, 330-331
 - json.Encoder, 330-331
 - multiple JSON structs at once, 331
 - json.Marshal, 329
 - json.Unmarshal, 329
 - about unmarshaling, 327
 - dash for name of ignored field, 328
 - populating variable from, 129
- enumerations via iota, 152
 - printable via stringer, 280
- environment variables
 - go help environment to list, 265
 - GOARCH, 283
 - GOBIN, 265
 - GOGC, 139-141
 - GOMEMLIMIT, 140, 364
 - GOOS, 283
 - GOPATH, 265
 - GOPRIVATE, 260
 - GOPROXY, 259, 266

- GOROOT, 265
- GOTOOLCHAIN, 225
- GOWORK, 258
- testing using, 376
- equal sign (=)
 - := versus var, 28-30
- arithmetic operators, 22
- comparison operators
 - arrays, 38
 - comparable interface, 186, 196
 - complex types, 24
 - floating-point types, 23
 - integer types, 22
 - interfaces comparable, 165
 - maps not comparable, 56
 - slices not comparable, 40
 - string types, 25
 - structs comparable if fields comparable, 64
 - time.Time methods, 325, 326
- errgroup package Group method for WaitGroup errors, 307
- documentation URL, 307
- error interface type, 163, 204
 - wrapped errors, 171
- errors
 - actions taken on returned error via defer, 112
 - compile-time errors
 - anonymous function name, 104
 - append return value not assigned, 41
 - declared variable not used, 32
 - defer function parentheses, 113
 - importing a package but not using exported identifiers, 228
 - importing a package without go.mod references, 242
 - impossible type parameter interface instantiated, 192
 - multiple return values assigned to one variable, 97
 - parentheses around returned values, 96
 - slice capacity less than length, 44
 - struct pointer field assigned a literal, 122
 - type elements as other than type constraint, 190
 - variadic input parameter dot placement, 96
 - workspace not published, 257
 - detecting in code via go vet, 7
 - (see also testing)
 - documentation URL, 221
 - error handling, 203-204
 - error messages not capitalized or punctuated, 203
 - multiple return values, 203-204
 - sentinel errors, 205-207
 - strings for simple errors, 205
 - values for errors, 208-210
 - wrong type assertion, 168, 169
 - error interface type, 163, 204
 - wrapped errors, 171
 - Go as call-by-value language, 41
 - importance of error checking, 103
 - io.Reader
 - io.EOF, 321
 - io.ErrUnexpectedEOF, 321
 - panic and recover, 218-220
 - (see also panics)
 - returning errors via return value, 96
 - actions taken via defer, 112
 - last return value, 203
 - nil returned if no error, 96, 203
 - other return values to zero values on error, 203
 - why not throwing exceptions, 204
 - stack trace from, 220
 - wrapping errors, 171, 210-212
 - defer, 217
 - sentinel errors not checkable by ==, 214
 - wrapping multiple errors, 212
- errors package
 - documentation URL, 221
 - errors.As, 171, 216
 - errors.Is, 171, 214
 - errors.New, 203, 205
- escape analysis, 137
 - URLs to more information, 139
- exclude directive, 254
 - retract directive versus, 255
- exporting package-level identifiers, 227
 - aliases, 238

F

- fallthrough keyword avoided, 86
- false, 19
 - booleans only, 27

- zero value for bool type, 19
- file block, 67
- first program, 3-7
 - building executable, 4
 - go fmt before, 6
 - code, 4
 - formatting, 5
 - packages, 4
 - creating go module, 3
 - nonunique name, 224
- floating-point literals, 18
 - 0x prefix for hexadecimal, 18
 - default type of float64, 22
 - p indicating exponent, 18
 - underscores for readability, 18
- floating-point types, 22
 - arithmetic operators, 23
 - dividing by zero, 23
 - division, 23
 - decimal module by ShopSpring, 241
 - The Floating Point Guide, 23
 - comparing floats, 23
 - inexactness of, 23
 - comparing floats, 23
 - using float64 simple, 22
 - zero value as 0, 22
- fmt package, 4
 - fmt.Errorf, 205
 - fmt.Printf, 7
 - stack trace via verbose output, 221
 - fmt.Println, 4
 - passing error to, 205
 - fmt.Stringer, 325
- for statement, 72-84
 - about, 72
 - break statement, 75
 - choosing the right statement, 83
 - complete for statement, 72
 - condition-only for statement, 73
 - continue statement, 75
 - for-range statement, 76-82
 - channel read via, 292
 - iterating over maps, 78
 - iterating over strings, 80
 - underscore (_) for unused variable name, 77
 - value variable as a copy, 81
 - value variable copy control, 81
 - for-select loop, 296
 - goroutines and for loop index, 298-299
 - infinite for statement, 74
 - labeling, 82, 86
 - nested, 82
 - shadowing variables, 73
 - single-letter variable names, 35
- forks and overriding dependencies, 254
- formatting of code, 5
 - go fmt, 5
- Fossil for version control, 252
- Fowler, Martin, 402
- func keyword
 - function declarations, 93
 - method declarations, 144
 - new function within a function, 103
- functions
 - accept interfaces, return structs, 162-164
 - anonymous functions, 103
 - as closures, 105-109
 - automating repetitive tasks via reflection, 422
 - call by value language, 114-116, 125
 - channel passed to a function, 291
 - map passed to a function, 131
 - pointer passed to a function, 125-127
 - call-by-value language, 41
 - calling, 93
 - anonymous functions, 104
 - consecutive input parameters of same type, 94
 - emulating named and optional parameters, 94
 - input parameters must be supplied, 94
 - no input parameters, 94
 - stack, 136
 - variadic input parameters, 95
 - closures, 105-109
 - passing functions as parameters, 107
 - returning functions from functions, 108
 - declaring, 4, 93
 - methods versus, 144
 - new function within a function, 103
 - defer, 109-114
 - functions are values, 100-103
 - calculator example code, 101
 - signature of function, 100
 - zero value of function variable as nil, 101
 - generic functions (see generic functions)
 - goroutines in concurrency, 289

- channels for communication, 291
 - launching, 290
 - scheduler, 289
 - terminating, 300, 301, 358-363, 367
 - helper functions (see helper functions)
 - “higher-order functions” definition, 109
 - input parameters not modified, 41, 114-116, 125
 - pointer passed to a function, 125-127
 - interface implementation, 173
 - methods as replacements for, 149
 - when to use methods versus functions, 150
 - no overloading, xvii, 144
 - passing functions as parameters, 107
 - when to use interface instead, 173
 - programs starting with main function, 4
 - returning values, 93
 - blank returns never should be used, 99
 - concrete types returned, 162
 - error handling, 203-204
 - errors, 96
 - errors as last return values, 203
 - goroutine return values ignored, 290
 - ignoring returned values, 78, 97
 - maps for return values, 131
 - multiple return values, 96
 - multiple return values via multiple variables, 97
 - named return values, 98
 - named return values used by defer, 112
 - no return value, 94
 - parentheses around returned values, 96
 - return keyword, 93
 - returning functions from functions, 108
 - reused err variable, 205
 - stack for, 136
 - value types favored over pointers, 130
 - triggering to run after time duration, 327
 - type declarations, 103
 - fuzzing, 386-393
- ## G
- Gamma, Erich, 154, 158
 - Gang of Four book, 154
 - garbage collector
 - algorithm, 138
 - implementation details talk URL, 138
 - Go runtime, 42
 - “A Guide to the Go Garbage Collector”
 - URL, 141
 - heap, 137
 - pointer memory management, 121
 - reducing workload, 136-139
 - buffers, 135, 136
 - interface trade-off, 163
 - Java versus Go, 138
 - slices growing, 42
 - stack, 136
 - thrashing, 140
 - tuning, 139-141
 - using values versus pointers, 125
 - generic data structures with generic functions, 194-196
 - generic functions
 - abstract algorithms, 187-188
 - actions made possible by, 184
 - adding to standard library, 201
 - features not implemented in Go, 198-199
 - fixing struct pointer error, 122
 - future feature possibilities, 201
 - generic data structures with, 194-196
 - idiomatic Go and, 199
 - interfaces and, 188
 - introducing, 184-186
 - stack created via, 184
 - type constraint, 185
 - Type Parameters Proposal URL, 184
 - why not initially included, 184
 - operators via type terms, 190-192
 - comparable interface with generics, 196
 - repetitive code reduced, 181-184
 - type inference and, 193
 - GiB suffix, 140
 - Gin web framework, 342
 - Git for version control, 252
 - git restore go.mod, 244
 - needed to download from GitHub, 266
 - GitHub
 - Git needed for downloading, 266
 - pushing workspace files to, 257
 - Go
 - backward compatibility, xvii
 - Go Compatibility Promise, 13, 163
 - boring, xviii
 - but practical, 178
 - call by value language, 114-116, 125

- map passed to a function, 131
- pointer passed to a function, 125-127
- call-by-value language, 41
- cgo for C library integration, 433-438
- compiling to a single native binary, 2, 240
 - Go runtime compiled into, 42
 - Go tool updates and, 14
 - reading build info inside, 281
- creators of, 55
- Effective Go
 - idiomatic Go resource, 7
 - semicolon insertion rule, 6
- errors returned instead of exceptions
 - thrown, 204
 - (see also errors)
- first program, 3-7
 - building executable, 4
 - code, 4
 - creating go module, 3, 224
 - formatting of code, 5
 - packages, 4
- formatting of code, 5
- Go Programming Language Specification, 17
- improvements, xvii
- installing Go tools, 1
 - (see also development environment)
- project organization, 3
- requirements for learning, xviii
- semantic versioning, 246
 - specifications URL, 247
- The Go Playground, 10-11
- updating managed by go directive, 225
- wiki Code Review Comments page, 7
- go commands
 - about, 2
 - ./..., 5
 - go build, 4
 - o flag for name or path change, 5
 - reading build info inside binary, 281
 - third-party packages, 242-244
 - go clean -modcache, 244
 - Go Compatibility Promise and, 14
 - go fmt, 5
 - braces not fixed by, 6
 - Go Playground Format button, 10
 - goimports as enhanced version, 266
 - go generate, 278-280
 - Makefiles and, 281
 - go get
 - go get ./..., 242, 256, 258
 - module path passed to, 244
 - proxy server, 259
 - updating to compatible versions, 248
 - updating to incompatible versions, 249
 - version specified, 246
 - go help, 286
 - go help environment, 265
 - go help importpath, 286
 - go install, 234, 264-266
 - secondary Go environment, 285
 - go list, 246
 - go mod, 3
 - go mod init, 224, 256
 - go mod tidy, 245, 250
 - go mod vendor, 250
 - go run, 263
 - go test, 130, 371-373, 374
 - benchmarks, 395
 - caching test results, 377
 - code coverage, 384-386
 - data race detector, 406
 - fuzzing, 386-393
 - go-cmp to compare results, 378-380
 - integration tests, 405
 - short flag, 406
 - storing sample test data, 376
 - go version, 2
 - m flag to specify filename, 282
 - go vet, 7
 - shadowing not reported, 71
 - struct tag validation, 328
 - third-party tools, 267-272
 - go work, 257
 - Makefiles to automate, 12
- Go Doc comments, 231-234
 - documentation URL, 234
- The Go Playground, 10-11
 - Google security for content removal, 11
- Go Programming Language Specification, 17
- Go Project golang.org/x packages, 267
- Go runtime, 42
 - compiling to a single native binary, 2, 240
 - Go tool updates and, 14
 - reading build info inside, 281
 - error runtime information via fmt.Errorf, 205
 - garbage collector tuning, 139-141

- generic function impact on performance, 200
- hash algorithms implemented, 57
- scheduler, 289
 - more information URL, 290
- “Go To Statement Considered Harmful” (Dijkstra), 89
- go-cmp to compare test results, 378-380
- go.mod file, 3, 224
 - contents of, 224
 - exclude directive, 254
 - retract directive versus, 255
 - go directive
 - for-range value as copy, 81, 225
 - managing Go build versions, 225
 - module path versioning tool, 253
 - replace directive, 254
 - workspaces for simultaneous module modifications, 256
 - require directives, 226
 - dependencies, 226
 - imports marked as indirect, 245
 - retract directive, 255
 - exclude directive versus, 255
 - rolling back with git restore, 244
 - synchronizing with go.sum, 245
 - third-party code references, 242
 - toolchain directive for version control, 225
- go.sum file, 243
 - module listed though removed, 246
 - synchronizing with go.mod, 245
- go.work file, 257
- go/bin directory, 265
- go:embed, 274-277
- GOARCH environment variable, 283
- GOBIN environment variable, 265
- GOGC environment variable, 139-141
- goimports tool improving formatting, 266
- GoLand (JetBrains), 9
- golang.org/x packages, 267
 - errgroup.Group for WaitGroup errors, 307
 - golang.org/x/sys, 426
- golanci-lint linter, 270-272
- GOMEMLIMIT environment variable, 140, 364
 - soft limit, 140
- Gonum package for numerical computing, 25
- Google proxy server, 259
 - checksum database, 259
- Google security for Playground content, 11
- Google Wire helper, 178
- GOOS environment variable, 283
- GOPATH environment variable, 265
- gopls language server, 8
- GOPRIVATE environment variable, 260
- GOPROXY environment variable, 259, 266
- gorilla mux request router, 342
- GOROOT environment variable, 265
- goroutines in concurrency, 289
 - always clean up goroutines, 299
 - cancellation in your own code, 367
 - context cancellation, 301, 358-363
 - context to terminate goroutines, 300
 - goroutine leak, 299
 - channels for communication, 291
 - for loop index variable, 298-299
 - launching, 290
 - must eventually exit, 299
 - scheduler, 289
 - more information URL, 290
 - terminating, 300
 - cancellation in your own code, 367
 - context cancellation, 301, 358-363
 - waiting for others to finish, 306-307
- goto keyword generally not used, 89-92
- GOTOOLCHAIN environment variable, 225
- govulncheck tool for security vulnerabilities, 272-274
- GOWORK environment variable, 258
- Grokking Algorithms (Bhargava), 57
- “A Guide to the Go Garbage Collector” URL, 141
- gzip package
 - gzip.NewReader function, 322
 - gzip.Reader pointer to io.Reader, 322

H

- hash map implementation of maps, 57
 - Go runtime hash algorithm implementation, 57
- Haskell having higher-order functions, 109
- heap, 137
 - garbage collector tuning, 139-141
 - garbage collector workload reduced, 136-139
 - interface trade-off, 163
 - goroutine leak, 299

- heap size control environment variable, 139-141
- more information on heap versus stack, 139
- Hello World, 3-7
 - building executable, 4
 - go fmt before, 6
 - code, 4
 - formatting, 5
 - packages, 4
 - creating go module, 3
 - nonunique name, 224
- Helm, Richard, 154, 158
- helper functions
 - building packages, 230
 - fixing struct pointer error, 122
 - io package, 324
 - maps
 - maps.Equal, 59
 - maps.EqualFunc, 59
 - running a function exactly once, 309
- hexadecimal prefix (0x), 18
 - uses for hexadecimal, 18
- hey tool to load test HTTP servers, 265
- hidden files in embedded content, 277
- “higher-order functions” definition, 109
- Hoare, Tony, 287
- Homebrew installation of Go, 1
 - updating Go tools, 14
- http support (see net/http package)
- Hudson, Rick, 138
- Hyrum’s law, 237

I

- IDEs, 8
 - Go development tools URL, 8
 - GoLand, 9
 - Visual Studio Code, 8
- if keyword, 71
 - choosing between switch and if, 89
 - shadowing variables, 68
 - variables scoped to if and else blocks, 71
- imag function, 24
- ImageMagick, 438
- imaginary literals, 25
- immutable values in Go, 125
 - constant immutability, 31
- import compatibility rule, 248
- import keyword, 4, 227
- blank imports, 239
- import path, 228
 - tool to automate versioning, 253
- shadowing variables, 69
- third-party code imported, 240
- import path, 228
 - relative path import paths not working with modules, 229
- index expression for strings, 53
- ±Inf when dividing by zero, 23
- inheritance not a part of Go, 150
 - code reuse via composition and promotion, 154
 - embedding is not inheritance, 156
- init function avoided, 239, 308
 - blank imports, 239
- “Inside the Map Implementation” video, 57
- int for int32 or int64, 20
 - uint as unsigned int, 21
- int32 via rune type, 26
- integer literals, 18
 - defaulting to int type, 20
 - prefixes indicating bases, 18
 - underscores for readability, 18
- integer types, 20-22
 - arithmetic operators, 21
 - dividing by zero, 22
 - choosing which to use, 21
 - special integer types, 20
 - zero value as 0, 20
- integration tests, 405
- interface keyword, 157
 - interface{}, 166
 - any as type alias for, 167
- interface literals, 157
- interfaces, 157
 - accept interfaces, return structs, 162-164
 - error interface type exception, 163
 - heap allocation trade-off, 163
 - as comparable, 165, 196
 - comparable interface for any, 186, 196
 - interfaces as type constraints, 188
 - declaring, 157
 - named with “er” endings, 158
 - embedding and, 162
 - empty interface for variable value of any type, 166
 - any as type alias for, 167
 - functions implementing, 173

- generic functions and, 188
 - implicit implementation, 158
 - dependency injection easier, 174-178
 - safety and decoupling, static and dynamic, 158-161
 - nil and, 164
 - reflection checking for nil value, 417
 - optional interfaces, 170
 - passing functions as parameters or interface, 173
 - standard interfaces, 160
 - interface instance returning type implementing same interface, 161
 - io package, 160, 320, 323
 - type assertions, 167
 - uses for, 170-172
 - type elements, 190-192
 - only valid as type constraints, 190
 - type terms, 190-192
 - type switches, 169
 - uses for, 170-172
 - zero value as nil, 164
 - internal packages, 234
 - io package, 319-324
 - helper functions, 324
 - io.Closer, 323
 - io.Copy, 170, 322
 - io.LimitReader, 322
 - io.MultiReader, 322
 - io.MultiWriter, 322
 - io.NopCloser, 324
 - io.ReadAll, 324
 - io.ReadCloser, 323
 - io.Reader, 319-323
 - io.EOF, 321
 - io.ErrUnexpectedEOF, 321
 - passing to bufio.NewReader, 171
 - standard interface, 160, 320
 - strings.NewReader function, 321
 - io.ReadSeeker, 323
 - io.ReadWriteCloser, 323
 - io.ReadWriter, 323
 - io.ReadWriteSeeker, 323
 - io.Seeker, 323
 - whence constants, 323
 - io.WriteCloser, 323
 - io.Writer, 319-323
 - standard interface, 160, 320
 - io.WriteSeeker, 323
 - standard interfaces, 160, 320, 323
 - io/fs package
 - fs.embed.FS type, 275
 - fs.FS, 275
 - fs.ReadDirFS, 275
 - fs.ReadFileFS, 275
 - fs.WalkDir, 276
 - iota for enumerations, 152-154
 - advice on, 153
 - origin of iota, 152
 - It's FOSS resource for open source licenses, 252
- ## J
- JetBrains GoLand, 9
 - Johnson, Ralph, 154, 158
 - Joshi, Kavya, 290
 - JSON support in encoding/json package
 - json.Unmarshal
 - populating variable, 129
 - zero value versus no value via pointers, 131
 - JSON support in encoding/json package, 327-335
 - about JSON, 327
 - custom JSON parsing, 332-335
 - json.Decoder, 330-331
 - multiple JSON structs at once, 331
 - json.Encoder, 330-331
 - multiple JSON structs at once, 331
 - json.Marshal, 329
 - json.Unmarshal, 329
 - about unmarshaling, 327
 - dash for name of ignored field, 328
 - marshaling versus unmarshaling, 327
 - struct tags to add metadata, 327
 - json tag with struct field, 328
- ## K
- Kennedy, Bill, 139
 - keywords
 - break
 - for statement, 75
 - switch statement, 86
 - case, 84-87
 - chan, 291
 - const, 30-32
 - continue, 75
 - default, 84-87
 - defer, 109-114, 217

- else, 71
- fallthrough avoided, 86
- for, 72-84
- func
 - function declarations, 93
 - method declarations, 144
- go, 2
 - (see also concurrency; goroutines in concurrency)
- goto generally not used, 89-92
- if, 71
- import, 4, 227
- interface, 157
 - interface{}, 166
- map, 56
- package, 228
- predeclared identifiers versus, 70
- range, 76-82
- return, 93
- select, 294-297
- struct, 61
- switch, 84-87
- type, 143
 - alias declaration, 238
 - functions, 103
 - interface, 157
 - structs, 61, 143
- var, 28-30
- KiB suffix, 140
- Kubernetes containers with Go app image, 2
 - blog post resource, 2

L

- language servers, 8
- Layher, Matt, 426
- lazy load, 308
- len function
 - array length, 38
 - channel buffer contents, 292
 - map key-value pair count, 56
 - slice length, 41
 - example code, 43
 - string length in bytes, 53
- lib.go file, 256, 258
- LICENSE file for open source modules, 252
 - It's FOSS resource, 252
- Lindamood, Jack, 162
- linters, 267-272

- golangci-lint, 270
- revive, 269
- staticcheck, 268-269
- Linux installation of Go, 1
 - updating Go tools, 14
- literals, 18
 - array literals, 37
 - constants for naming literals, 31
 - floating-point literals, 18
 - imaginary literals, 25
 - integer literals, 18
 - interface literals, 157
 - map literals, 56
 - rune literals, 18
 - slice literals, 39
 - string literals, 19
 - raw string literals, 19
 - struct literals, 62
 - untyped, 19, 27
 - default type, 19
 - type conversion, 29
- log/slog package, 344-346
 - API documentation URL, 346
 - slog.Debug, 344
 - slog.Error, 344
 - slog.Info, 344
 - slog.NewLogLogger, 346
 - slog.Warn, 344
- looping (see for statement)

M

- Mac Homebrew installation of Go, 1
 - updating Go tools, 14
- main package, 4
 - main function, 4, 93
 - no input parameters, 94
 - module organization, 236
- make function
 - channels created, 291
 - buffer capacity, 292
 - map declaration, 56
 - slice declaration, 43
 - append function after, 44, 46
- Makefiles, 12
 - go generate and, 281
 - make fmt, 13
 - make vet, 13
 - tutorial URL, 13

- map literals, 56
- maps, 56-61
 - call by value language, 115
 - maps for input parameters or return values, 131
 - pointer implementation of maps, 116, 131
 - comma ok idiom for keys not in map, 58
 - declaration, 56
 - make function for sized map that grows, 56
 - deleting key-value pairs, 59
 - for statement iterating over, 78
 - hash map implementation, 57, 79
 - keys as any comparable type, 57
 - len function for number of key-value pairs, 56
 - clear function to set length to zero, 59
 - not comparable, 56
 - maps.Equal, 59
 - maps.EqualFunc, 59
 - reading and writing a map, 57
 - 0 for value for key not in map, 58
 - determining if key not in map, 58
 - slices versus maps, 57
 - using as sets, 60
 - struct{ } for value, 61
 - zero value as nil, 56
 - empty versus nil map literals, 56
 - writing to nil map causing panic, 56
- maps package
 - maps.Equal, 59
 - maps.EqualFunc, 59
- marshaling
 - about, 327
 - dash for name of ignored field, 328
 - reflection for data marshaler, 417-422
- Marti, Vicent, 200
- mathematical operators (see arithmetic operators)
- mechanical sympathy, 138
- memory
 - B, KiB, MiB, GiB, TiB suffixes, 140
 - garbage collector tuning, 139-141
 - goroutine leak, 299
 - heap, 137
 - heap versus stack information URL, 139
 - memory allocation via Go runtime, 42
 - memory allocation via stack pointer value, 136
 - memory limit environment variable, 140
 - reducing garbage collector's workload, 136-139
 - interface trade-off, 163
 - slices growing, 42
 - stack, 136
 - stack pointer, 136
 - thrashing, 140
 - variable storage, 119
- Mercurial for version control, 252
- metadata
 - context
 - about the problem, 349
 - cancellation, 358-363
 - cancellation in your own code, 367
 - contexts with deadlines, 363-367
 - description, 349-352
 - values, 352-358
 - struct tags for, 327
- method expression, 150
- method set, 147
 - interface implicit implementation, 158
 - methods on embedded field and containing struct, 157
- method value, 150
- methods
 - about, 144
 - adding to types, 324
 - declaring, 144
 - functions versus, 144
 - receiver specification, 144
 - functions replaced by methods, 149
 - when to use functions versus methods, 150
 - interface declaration, 157
 - invoking, 145
 - nil instance handling, 148
 - no inheritance from type based on type, 150
 - executable documentation, 151
 - no overloading, 144
 - as parameters or variables, 150
 - pointer receivers, 145-147
 - nil instance handling, 148
 - reflection cannot add to a type, 424
 - struct field direct access encouraged, 147
 - value receivers, 145-147
 - nil instance handling, 148

- Miara, Richard, 114
- MiB suffix, 140
- middleware pattern of HTTP server, 340-341
 - context
 - cancellation, 358-363
 - cancellation in your own code, 367
 - contexts with deadlines, 363-367
 - description, 349-352
 - values, 352-358
- module cache, 244
 - deleting, 244
- module path, 224
 - tool to automate versioning, 253
- modules
 - about, 3, 223
 - API renamed and reorganized, 238
 - code documented with Go Doc comments, 231-234
 - creating a Go module, 3
 - documentation URL, 260
 - go.mod file, 3, 224
 - module path, 224
 - tool to automate versioning, 253
 - open source Go module site, 251
 - organization of, 236-238
 - proxy server, 259
 - checksum database, 259
 - publishing, 251
 - retracting versions, 255
 - third-party code imported, 240
 - versions, 245
 - dependencies replaced with specified fork, 254
 - dependency copies inside module, 250
 - excluding module versions, 254
 - go list to see versions, 246
 - import compatibility rule, 248
 - minimal version selection, 247
 - resolving different version dependencies, 247
 - semantic import versioning rule, 249
 - semantic versioning in Go, 246
 - semantic versioning specifications URL, 247
 - updating to compatible versions, 248
 - updating to incompatible versions, 248, 253
 - vendoring, 250
 - version control software, 252

- version control system documentation URL, 252
- versioning your module, 252
- workspaces for simultaneous modifications, 255-259

- modulus (%) for integer types, 21
 - not floating-point types, 23
- monotonic time, 326
- Musseman, Joyce, 114
- mutexes instead of channels in concurrency, 313-316
 - critical section, 313
 - passed via pointer, 316

N

- naming variables and constants, 33
 - camelCase instead of snake_case, 34
 - case of first letter in package-level declaration, 34
 - smaller the scope, shorter the name, 35
- NaN (Not a Number) when dividing by zero, 23
- Navarro, Juan, 114
- nested code harder to read, 114
- net/http package, 336-343
 - Client type, 336-343
 - context
 - about metadata problem, 349
 - http.DefaultServeMux and functions avoided, 339
 - http.Flusher interface, 342
 - http.Handler interface, 337
 - http.Hijacker interface, 342
 - http.NewRequestWithContext, 336
 - http.NewServeMux, 339
 - http.Request context-related methods, 350
 - http.ResponseController, 342
 - *http.ServeMux, 339
 - http.Server, 337
 - middleware pattern, 340-341
 - third-party modules to enhance, 342
 - testing via httptest, 402-404
- new function for pointer variable creation rarely used, 121
- newline rune literal ('\n'), 18
- nil
 - about, 40, 120
 - defined in universe block, 120
 - can be shadowed, 120

- error return value when no error, 96, 203
- function variable zero value, 101
- interfaces and, 164
 - reflection checking for nil value, 417
- methods coded for nil instances, 148
- zero value
 - channels, 291
 - maps, 56
 - pointers, 120
 - slices, 40
- numeric types, 20-25
 - complex types, 23-25
 - explicit type conversion, 26
 - floating-point types, 22
 - integer types, 20-22
 - numerical computing in Go, 25
 - decimal module by ShopSpring, 241
 - Gonum third-party package, 25

O

- octal prefix (0o), 18
 - 0 alone should not be used, 18
 - uses for octal, 18
- online resources (see resources online)
- open source Go module site, 251
- operator generic functions via type terms, 190-192
- operator overloading not part of Go, xvii, 198
- os package
 - os.Args, 110
 - os.Create, 324
 - *os.File instance returned, 324
 - os.File with io.Reader interface, 161
 - os.NewFile, 324
 - os.Open, 110, 161, 324
 - os.OpenFile, 324
 - os.ReadFile, 324
 - os.WriteFile, 324

P

- p for exponent in floating-point literals, 18
- package blocks, 30, 67
 - anonymous functions, 105
 - descriptive variable and constant names, 35
 - method definitions, 144
 - sentinel errors declaration, 206
- package clause, 228
 - identical for every Go file in a directory, 229

- package keyword, 228
- packages, 223
 - building
 - circular dependencies, 235
 - comments in Go Doc format, 231-234
 - creating and accessing, 227
 - import path, 228
 - imports and exports, 227
 - init function avoided, 239
 - internal package, 234
 - naming packages, 230
 - overriding a package's name, 230
 - package clause, 228
 - package main, 228, 229
 - package name matching directory containing it, 229
 - renaming and reorganizing API, 238
 - case of first letter in package-level declaration, 34
 - declaration, 4
 - import statement, 4
 - shadowing variables, 69
 - main package, 4
 - module organization, 236
 - module organization, 236-238
- panics
 - about panics, 218-220
 - recover, 219
 - stack trace, 218
 - array access out of bounds, 38
 - channel behavior chart, 293
 - comparing uncomparable types, 196
 - dereferencing a nil pointer, 121
 - dividing by zero, 22
 - function variable of nil value running, 101
 - map of nil value written to, 56
 - slice capacity less than length, 44
 - type assertion wrong, 168
 - value receiver method called with nil value pointer instance, 146
- parameters
 - call by value language, 114-116, 125
 - pointer passed to a function, 125-127
 - call-by-value language, 41
 - context as first, 349
 - functions declared and called, 93
 - emulating named and optional parameters, 94
 - input parameters must be supplied, 94

- no input parameters, 94
- variadic input parameters, 95
- goroutine invocation, 290
- input parameters not modified, 41, 114-116, 125
 - maps for input parameters, 131
 - pointer passed to a function, 125-127
 - slices for input parameters, 132
- method passed to, 150
- passing functions as parameters, 107
 - when to use interface instead, 173
- pointers indicating mutable parameters, 125-127, 145
 - pointers improving performance, 130
 - pointers used as last resort, 129
- personally identifiable information on Go Playground, 11
- Pike, Rob, 55
- pkg.go.dev open source Go module site, 251
- pkgsite powering, 234
- pkgsite tool
 - documentation formatting viewer, 234
 - pkg.go.dev powered by, 234
- pointer receivers, 145-147
 - nil instance handling, 148
- pointer type, 121
 - pointer receivers, 145-147
 - methods with value receiver methods in method set, 147
- pointers, 119
 - address pointing to, 120
 - call by value behavior of maps and slices, 116
 - channels passed to functions as, 291
 - creating
 - & before struct literal, 122
 - & before struct literal exception, 122
 - new function rarely used, 121
 - numbers, booleans, strings, constants need variable, 122
 - dereferencing, 121
 - non-nil pointers only, 121
 - indicating mutable parameters, 125-127, 145
 - pointers improving performance, 130
 - pointers used as last resort, 129, 139
 - pointer passed to a function, 125-127
 - pointer type, 121
 - pointer receivers, 145-147
- syntax
 - & as address operator, 121
 - * as indirection operator, 121
 - used as last resort, 129
 - but performance improvements, 130
 - using values versus pointers, 123
 - lack of immutable declarations and, 125
 - zero value as nil, 120
 - zero value versus no value indication, 131
- pprof code profiler, 397
- pre-releases in semantic versioning, 252
- predeclared types
 - about, 17
 - booleans, 19
 - complex types, 23-25
 - explicit type conversion, 26
 - floating-point types, 22
 - keywords versus, 70
 - numeric, 20-25
- “Preemptive Interface Anti-Pattern in Go” (Lindamood), 162
- Printf (fmt package), 7
 - stack trace via verbose output, 221
- Println (fmt package), 4
 - passing error to, 205
- private repositories, 260
- promotion and composition, 154
- Protocol Buffers (protobufs), 278
 - schema, 278
- proxy server for modules, 259
 - checksum database, 259
 - disabling via GOPROXY, 259
 - running own proxy server, 259
- publishing your module, 251
 - LICENSE file, 252
 - retracting versions, 255

R

- race checker, 406
- range keyword, 76-82
- Read method of file, 110
- real function, 24
- receiver specification of methods, 144
- reflect package, 409, 411
 - reflect.DeepEqual, 40, 411
 - reflect.New, 415
 - reflect.Type, 415
 - reflect.TypeOf, 411-413

- reflect.Value, 415
- reflect.ValueOf, 414
- reflection
 - about, 410
 - automating repetitive tasks, 422
 - avoid building structs with, 423
 - cannot add methods to a type, 424
 - checking for nil interface value, 417
 - data marshaler, 417-422
 - making new values, 415
 - types and kinds, 411-413
 - use only if worthwhile, 424
 - values, 414
- relative path import paths not working with modules, 229
- repetitive code reduced via generic functions, 181-184
- replace directive, 254
 - workspaces for simultaneous module modifications, 256
- repositories, 223
 - GitHub
 - Git needed for downloading, 266
 - pushing workspace files to, 257
 - module path, 224
 - nonunique module names, 224
 - private repositories, 260
 - proxy server for modules, 259
 - publishing your module, 251
 - version control software, 252
 - Git needed for GitHub downloads, 266
- require directives, 226
 - dependencies, 226, 245
 - imports marked as indirect, 245
- resources online
 - arithmetic operators, 22
 - book supplemental material, xix
 - Chapter 1, 13-14
 - Chapter 2, 24, 27, 30, 33-35
 - Chapter 3, 43, 46-50, 52-54, 57, 60, 65
 - Chapter 4, 68-71, 74, 77-78, 80, 84, 86-87, 89-90, 92
 - Chapter 5, 94-95, 97, 99, 101, 104-105, 107-109, 111, 113-116
 - Chapter 6, 123, 130, 141
 - Chapter 7, 146, 149, 156, 158, 164, 166, 168, 172, 177-178
 - Chapter 8, 183, 186, 188, 190-193, 196-197, 200-201
 - Chapter 9, 204-206, 208-211, 213-214, 216, 218-219, 221
 - Chapter 10, 225, 231, 235
 - Chapter 11, 264, 268-271, 274, 276-277, 280-281, 286
 - Chapter 12, 291, 295-296, 300-313, 315, 317
 - Chapter 13, 322, 331-333, 335, 337-341, 343-347
 - Chapter 14, 352, 355, 358, 361, 363, 365-366, 368
 - Chapter 15, 371, 377-378, 380, 383-384, 393, 397, 399, 402-406
 - Chapter 16, 413, 417-418, 423, 425, 428-429, 433-436, 438
 - goroutine_for_loop repository, 298
 - money repository, 241
 - package examples, 227
- containers with Go app image, 2
- data race detector documentation, 408
- Effective Go
 - idiomatic Go resource, 7
 - semicolon insertion rule, 6
- errors package documentation, 221
- The Floating Point Guide, 23
 - comparing floats, 23
- garbage collector algorithm implementation details, 138
- generic functions
 - Type Parameters Proposal, 184
 - why not initially included, 184
- Go Compatibility Promise, 13
 - Russ Cox GopherConn 2022 keynote talk, 13
- Go development tools, 8
- Go Doc comments documentation, 234
- Go Programming Language Specification, 17
- Go runtime
 - performance and generics, 200
 - scheduler information, 290
- Go tools on Go website, 1
- Go wiki Code Review Comments page, 7
- Google security for Playground content, 11
- “A Guide to the Go Garbage Collector”, 141
- heap versus stack and escape analysis, 139
- indentation of code and readability, 114
- “Inside the Map Implementation” video, 57
- language server protocols, 8

- log/slog package documentation, 346
 - Makefile tutorial, 13
 - memory efficiency blog post, 138
 - mocks covered in blog post, 402
 - module documentation, 260
 - module organization, 237
 - module path versioning tool, 253
 - monotonic time blog post, 327
 - open source Go module site pkg.go.dev, 251
 - open source licenses, 252
 - pprof code profiler, 397
 - “Preemptive Interface Anti-Pattern in Go” (Lindamood), 162
 - semantic versioning specifications, 247
 - standard library documentation, 319
 - unsafe blog post, 426
 - unsafe documentation, 433
 - updating code to incompatible versions, 253
 - retract directive, 255
 - exclude directive versus, 255
 - return keyword, 93
 - blank returns never should be used, 99
 - concrete types returned, 162
 - defer functions running after, 111
 - goroutine return values ignored, 290
 - ignoring returned values, 78, 97
 - maps for return values, 131
 - multiple return values, 96
 - error handling, 203-204
 - errors as last return value, 203
 - errors returned, 96
 - ignoring returned values, 78, 97
 - multiple variables for, 97
 - named return values, 98
 - defer using, 112
 - no return value, 94
 - parentheses around returned values, 96
 - returning functions from functions, 108
 - stack for returning values, 136
 - value types favored over pointer return values, 130
 - revive linter, 269
 - Roussel, Achille, 139
 - rune integer type, 21, 26, 52-55
 - characters via, 26
 - for statements iterating over strings, 80
 - type conversions to strings and bytes, 54
 - rune literals, 18
 - backslash-escaped rune literals, 18
 - default type as rune, 26
 - string literals, 19
- ## S
- Sandberg, Anna, 114
 - The Scheduler Saga (talk by Joshi), 290
 - schema for Protocol Buffer, 278
 - “scripting” via go run, 264
 - security vulnerabilities tool govulncheck, 272-274
 - select keyword, 294-297
 - default clause, 296
 - for loop for multiple channels, 296
 - selecting a closed channel, 304
 - semantic import versioning rule, 249
 - semantic versioning in Go (SemVer), 246
 - pre-releases, 252
 - semantic import versioning rule, 249
 - specification URL, 247
 - versioning your module, 252
 - semicolon insertion rule, 6
 - sentinel errors, 205-207
 - constants as, 207
 - declared in package block, 206
 - origin of term, 205
 - testing for, 206
 - wrapped not checkable by ==, 214
 - errors.Is, 214
 - sets, 60
 - maps as sets, 60
 - struct{} for value, 61
 - shadowing variables, 68-71
 - type switch making shadowing good, 170
 - Shneiderman, Ben, 114
 - ShopSpring decimal module, 241
 - signature of function, 100
 - single quote (')
 - not interchangeable with double quote, 18
 - rune literals, 18
 - single quote rune literal ("\'"), 18
 - slice expressions, 46-49
 - full slice expressions, 48
 - sharing memory, 47
 - slice literals, 39
 - slices, 39-50
 - about, 39
 - all elements to zero value, 44
 - append function to grow, 41

- full slice expression protecting subslices, 48
- arrays converted to, 50
- buffers for reading data, 135
- call by value language, 115
 - pointer implementation of slices, 116
- capacity, 42
 - cap function for capacity, 43
 - example code, 43
 - never less than length, 44
- converted to arrays, 51
- copy function, 49
- declaration, 39
 - make function for specifics, 43
 - make function with append function
 - after, 44, 46
 - which to use when, 45
 - without using literal, 40
- full slice expressions, 48
- function input parameters, 96, 132
 - how slices passed to functions, 134
 - slice copy appended to, 133
- len function for length, 41
 - example code, 43
- maps versus slices, 57
- multidimensional slices simulated, 39
- not comparable, 40
 - DeepEqual in reflect package, 40, 411
 - slices.Equal, 40
 - slices.EqualFunc, 40
- reading and writing, 39
- slices from slices, 46-49
 - full slice expression protecting from
 - append, 48
 - sharing memory, 47
- slices package in standard library, 40
- sort.Slice, 107
- strings converted to, 54
- zero value as nil, 40

slices package in standard library, 40

Smith, Forrest, 138

Sonatype proxy server support, 259

sort.Slice, 107

sparse arrays, 37

SQLite embedded in Go application, 438

stack, 136, 184

- garbage collector workload reduced, 136-139
- generic functions to create, 184
- goroutine leak, 299
- more information on heap versus stack, 139
- pointer type stored on stack, 136
 - data pointer pointing to, 137
- size increase during execution, 137
- stack frame, 136
- stack pointer, 136
- stack traces
 - errors providing, 220
 - panics providing, 218

standard library

- C pseudopackage, 434-438
- context package, 349
 - (see also context package)
- database/sql/driver package, 112, 163
- documentation URL, 319
- encoding/csv package, 417
- encoding/json package, 327-335
- errors package
 - documentation URL, 221
 - errors.As, 171, 216
 - errors.Is, 171, 214
 - errors.New, 203, 205
- fmt package, 4
- generic functions added to, 201
 - implementations pre-generics, 184
- Go Compatibility Promise, 13, 163
- io package, 160, 170, 319-324
- log/slog package, 344-346
- maps package, 59
- net/http package, 336-343
- os package, 324
- reflect package, 409, 411
- slices package, 40
- sort package, 107
- strconv package, 21, 91, 102
- strings package, 55
- sync package, 306, 314
- sys package, 426
- syscall package, 426
- testing package, 371
 - (see also testing)
- time package, 324-327
- unicode/utf8 package, 55
- unsafe package, 121, 425

Staron, Mirosław, 114

staticcheck linter, 268-269

strconv package

- strconv.Atoi, 102

- strconv.FormatInt, 21
- string literals
 - default type as string, 26
 - raw string literals, 19
 - string literals, 19
 - UTF-8, 52
 - string bytes versus UTF-8 multi-byte code points, 53
- string type, 25, 52-55
 - bytes representing string, 52
 - converted to slices, 54
 - error handling, 205
 - extracting single value from, 53
 - for statements iterating over, 80
 - immutable, 25
 - len function for length in bytes, 53
 - slice expression notation working with, 53
 - strings.NewReader function creating
 - io.Reader, 321
 - time converted to and from, 325
 - time duration strings, 325
 - type conversions to runes and bytes, 54
 - zero value as empty string, 25
- stringer tool for printable enumerations, 280
- strings package in standard library, 55
- struct keyword, 61, 143
- struct literals, 62
- structs, 61
 - accept interfaces, return structs, 162-164
 - anonymous structs, 63
 - comparing structs, 65
 - unmarshaling and marshaling data, 64
 - comparing structs, 64
 - anonymous structs, 65
 - declaration, 61
 - as user-defined type with underlying struct, 143
 - direct field access encouraged, 147
 - embedded field methods promoted to containing struct, 155
 - embedding is not inheritance, 156
 - fields or methods with same name, 155
 - emulating named and optional parameters, 94
 - input parameters, 131
 - maps used as sets, 61
 - pointer input parameter or return value for, 130
 - reading and writing, 63
 - reflection avoided for building, 423
 - return values, 131
 - struct tags for metadata, 327
 - type conversions, 64
 - zero value as field's zero value, 62
- structured logging, 344-346
- Subversion for version control, 252
- Sulaiman, Marwan, 253
- switch keyword, 84-87
 - blank switches, 87
 - break keyword, 86
 - case clause, 84-87
 - choosing between if and switch, 89
 - default, 84-87
 - fallthrough keyword avoided, 86
- sync package
 - sync.Map type, 316
 - sync.Mutex, 314
 - sync.Once type for running code once, 308
 - helper functions, 309
 - sync.RWMutex, 314
 - sync.WaitGroup, 306-307
- sys package, 426
- syscall package, 426

T

- tab rune literal ('\t'), 18
- table tests, 380-382
- The Tail at Scale (Dean), 138
- testing
 - about, 371
 - basics of, 371-373
 - caching test results, 377
 - environment variables in testing, 376
 - go-cmp to compare test results, 378-380
 - reporting test failures, 373
 - storing sample test data, 376
 - test setup and teardown, 373-375
 - testing your public API, 377
 - benchmarks, 393-397
 - profiling your code, 397
 - code coverage, 384-386
 - concurrency problems, 406
 - data race detector, 406
 - fuzzing, 386-393
 - go test, 130, 371-373, 374
 - benchmarks, 395
 - caching test results, 377

- code coverage, 384-386
- data race detector, 406
- fuzzing, 386-393
- go-cmp to compare results, 378-380
- integration tests, 405
- short flag, 406
- storing sample test data, 376
- go vet, 7
 - shadowing not reported, 71
 - struct tag validation, 328
 - third-party tools, 267-272
- httptest, 402-404
- integration tests and build tags, 405
- race checker, 406
- running tests concurrently, 382-383
- sentinel errors, 206
- stubs, 397-402
 - mocks versus, 402
- table tests, 380-382
- text package and using test data, 377
- versions of Go, 285
- text package and using test data, 377
- third-party code imported, 240
- third-party modules for http.Server, 342
- Thompson, Ken, 25, 55
- Thompson, Martin, 138
- thrashing, 140
- TiB suffix, 140
- tilde (~) for types with underlying type, 191
- time package, 324-327
 - Day, Month, Year, Hour, Minute, Second, Weekday methods, 326
 - formatting of date and time, 325
 - methods not modifying time.Time instance, 326
 - monotonic time, 326
 - importance of handling correctly, 327
 - time.Add, 326
 - time.AddDate, 326
 - time.After, 327
 - time.AfterFunc, 327
 - time.Clock, 326
 - time.Format, 325
 - time.NewTicker, 327
 - time.Now, 325
 - time.Parse, 325
 - time.ParseDuration, 325
 - time.Round, 325, 326
 - time.Sub, 326
 - time.Tick, 327
 - time.NewTicker instead, 327
 - time.Truncate, 325, 326
 - timeouts, 327
 - timing out code, 304
 - timers, 327
 - types
 - time.Duration, 324
 - time.Time, 324
 - time.Time comparisons, 325, 326
- time zone in time.Time type, 325
 - Equal method for comparison, 325
- timing out code in concurrency, 304
- toolchain directive for version control, 225
- tools
 - about, 263
 - (see also go commands)
 - building binaries for other platforms, 283
 - build tags, 284
 - code style check and bug scan, 267-272
 - golanci-lint, 270-272
 - revive, 269
 - staticcheck, 268-269
 - concurrency tools working together, 309-313
 - embedding content into program, 274-277
 - hidden files, 277
 - Go development tools URL, 8
 - go generate, 278-280
 - Makefiles and, 281
 - go help, 286
 - go help environment, 265
 - go help importpath, 286
 - go install, 264-266
 - GOBIN to set install path, 265
 - goimports tool install command, 266
 - GOPROXY for proxy or repository, 266
 - installing hey, 265
 - pkgsite installation, 234
 - secondary Go environment, 285
 - updating a tool to newer version, 266
 - The Go Playground, 10-11
 - go run to build and execute, 263
 - goimports improving formatting, 266
 - govulncheck scan for security vulnerabilities, 272
 - hey to load test HTTP servers, 265
 - IDEs, 8
 - GoLand, 9

- stringer for printable enumerations, 280
- testing versions of Go, 285
- updating to newer version, 266
- troubleshooting
 - code error detection via go vet, 7
 - (see also testing)
 - installation of Go tools, 2
 - make function slice with append after, 44, 46
 - universe block predeclared identifiers, 70
 - variables
 - declaring new with var, 29
 - package-level avoided, 30
 - shadowing variables, 68-71
- true, 19
 - booleans only, 27
- type assertions, 167-169
 - type conversions versus, 169
 - uses for, 170-172
- type conversion, 26
 - arrays of different sizes, 38
 - strings, runes, and bytes, 54
 - structs, 64
 - type assertion versus, 169
 - untyped constants or literals, 29
- type elements, 190-192
 - constants that can be assigned to variables, 193
 - type terms, 190-192
 - ~ for types with underlying type, 191
- type inference and generic functions, 193
- type keyword, 143
 - alias declaration, 238
 - functions, 103
 - interface, 157
 - structs, 61, 143
- type switches, 169
 - uses for, 170-172
- types
 - about Go, 143
 - types in Go, 143
 - abstract types, 144
 - interfaces, 157
 - adding a method to, 324
 - aliases, 238
 - all types as value types, 116
 - array size affecting type, 38
 - converting array to slice to array, 52
 - slices contrasted, 39
 - stack and, 136
 - byte for uint8, 20, 26
 - for statements iterating over strings, 80
 - string bytes versus UTF-8 multi-byte code points, 53
 - strings as sequences of, 52-55
 - type conversions to strings and runes, 54
 - channels, 291
 - composite types
 - arrays, 37-39
 - maps, 56-61
 - slices, 39-50
 - strings, runes, bytes, 52-55
 - structs, 61
 - concrete types, 144
 - interface implicit implementation, 158
 - constants typed and untyped, 32
 - context.Context, 350
 - default type, 19
 - duck typing, 158
 - empty interface for variable of any type, 166
 - any as type alias for, 167
 - error interface type, 163, 204
 - wrapped errors, 171
 - explicit type conversion, 26
 - automatic type promotion versus, 26
 - boolean treatment of variable and, 27
 - untyped constants or literals, 29
 - generic function abstract algorithms, 187-188
 - generic functions and type inference, 193
 - int for int32 or int64, 20
 - uint as unsigned int, 21
 - int32 via rune type, 26
 - iota enumeration type, 152
 - literals, 18
 - untyped, 19, 27
 - names of variables, 35
 - net/http package Client type, 336-343
 - nil untyped, 40, 120
 - pointer type, 121
 - pointer receivers, 145-147
 - predeclared types
 - about, 17
 - booleans, 19
 - complex types, 23-25
 - floating-point types, 22
 - numeric, 20-25
 - strings, 25

- reflection
 - about, 410
 - automating repetitive tasks, 422
 - avoid building structs with, 423
 - cannot add methods to a type, 424
 - checking for nil interface value, 417
 - data marshaler, 417-422
 - making new values, 415
 - types and kinds, 411-413
 - use only if worthwhile, 424
 - values, 414
- rune integer type, 21, 26, 52-55
 - characters via, 26
 - for statements iterating over strings, 80
 - type conversions to strings and bytes, 54
- Scanner type for `*os.File` instance, 324
- storage in memory, 119
- `sync.Map`, 316
- `sync.Once` type for running code once, 308
- time package
 - `time.Duration`, 324
 - `time.Time`, 324
- type assertions, 167-169
 - uses for, 170-172
- type constraint via generic functions, 188
- type elements, 190-192
 - constants that can be assigned to variables, 193
 - type terms, 190-192
 - ~ for types with underlying type, 191
- type switches, 169
 - uses for, 170-172
- `uint` as unsigned int, 21
- `uint8` as byte, 20, 26
- `uintptr` integer type, 21
- `unsafe.Pointer` type, 426
- user-defined types, 143
 - based on another user-defined type, 150
 - executable documentation, 151
 - methods, 144-151
 - underlying type of int or string, 173
 - value type of value receivers, 145-147
 - zero value of unassigned variables, 17

U

- `uint` as unsigned int, 21
- `uint8` as byte, 20, 26
- `uintptr` integer type, 21

- underscore (`_`)
 - blank imports, 239
 - first iota value when ignored, 154
 - floating-point literals, 18
 - hidden file name prefix, 277
 - integer literals, 18
 - return value ignored, 78, 97
 - return value nameless among named, 98
 - targeted code build tags, 284
 - unused variable and constant names, 77
 - variable and constant names not using, 34
- Unicode support
 - naming variables and constants, 33
 - rune literals, 18
 - strings, 25
 - UTF-8 encoding, 55
 - UTF-16 for one or two 2-byte sequences, 55
 - UTF-32 for four bytes per code point, 55
 - unicode/utf8 package in standard library, 55
- universe block, 70
 - nil defined, 120
 - predeclared identifiers defined, 70
- Unmarshal function
 - about unmarshaling, 327
 - dash for name of ignored field, 328
 - populating variable from JSON, 129
 - pointer input parameter, 129
- unsafe package, 121, 425
 - converting external binary data, 428-432
 - documentation URL, 433
 - `-gcflags=-d=checkptr`, 433
 - unexported field access, 432
 - `unsafe.Offsetof`, 427-428
 - `unsafe.Pointer` type, 426
 - `unsafe.Sizeof`, 426-428
- updating Go managed by go directives, 225
- updating modules
 - updating to compatible versions, 248
 - updating to incompatible versions, 248
 - workspaces for, 255-259
- updating third-party tools, 266
- user-defined types, 143
 - based on another user-defined type, 150
 - not inheritance, 150
 - executable documentation, 151
 - methods, 144
 - declaring, 144
 - invoking, 145

- underlying type of int or string, 173
- UTF-8 for string literals, 52
 - encoding for Unicode, 55
 - UTF-16 for one or two 2-byte sequences, 55
 - UTF-32 for four bytes per code point, 55
 - invented by Thompson and Pike, 55
 - string bytes versus UTF-8 multi-byte code points, 53

V

- value receivers, 145-147
 - methods in method set, 147
 - nil instance handling, 148
 - time package methods, 326
- value type of value receivers, 145-147
- van Heumen, Danny, 153
- var versus :=, 28-30
 - shadowing variables, 68-71
- variables
 - atomic variables, 317
 - bool type, 19
 - call by value language, 114-116, 125
 - pointer passed to a function, 125-127
 - call-by-value language, 41
 - declaration
 - about, 67
 - arrays, 37
 - blocks, 67
 - declaration list, 28
 - if and else block scope, 71
 - package-level variables avoided, 30, 33
 - pointers, 121
 - sentinel errors in package block, 206
 - shadowing variables, 68-71
 - shadowing variables good with type switch, 170
 - var versus :=, 28-30
 - empty interface to indicate value of any type, 166
 - any as type alias for, 167
 - environment variables (see environment variables)
 - err variable for errors, 205
 - explicit type conversion, 26
 - automatic type promotion versus, 26
 - method assigned to, 150
 - naming, 33

- camelCase instead of snake_case, 34
- smaller the scope, shorter the name, 35
- pointers, 119
 - (see also pointers)
- slices from slices sharing memory, 47
 - (see also slices)
- storage in memory, 119
- strings immutable, 25
 - (see also string type)
- unassigned variable zero value, 17
 - (see also zero value of unassigned variables)
- unused, 32
 - underscore (_) for name, 77
- variadic input parameter (. . .), 95
 - slice as input, 96
- variadic type parameters not part of Go, 199
- vendor, 250
- version control software, 252
 - version control system documentation URL, 252
 - versioning your module, 252
- virtual filesystem in embedded content, 275
 - example code for treating like real, 276
 - hidden files, 277
- Visual Studio Code, 8
 - Go extension, 8
- Vlissides, John, 154, 158
- vulnerabilities scanned for by govulncheck tool, 272-274

W

- WaitGroups in concurrency, 306-307
 - errgroup.Group for WaitGroups errors, 307
- whence constants for io.Seeker, 323
- while via condition-only for loop, 73
- whitespace formatting fixed via go fmt, 5
- wiki Code Review Comments page, 7
- Windows
 - Chocolatey
 - Go installation, 1
 - updating Go tools, 14
 - make installation, 13
- Wire helper (Google), 178
- workspaces for simultaneous module modifications, 255-259
- wrapping errors, 171, 210-212
 - sentinel errors not checkable by ==, 214

wrapping multiple errors, 212

Z

zero value of unassigned variables, 17

bool types as false, 19

channels as nil, 291

complex numbers as 0, 24

floating-point types as 0, 22

function variables as nil, 101

integer types as 0, 20

interface types as nil, 164

maps as nil, 56

pointers as nil, 120

slices as nil, 40

strings as empty string, 25

struct fields as field's zero value, 62

using var for initializing, 29