Index

A

Ansible's inventory files, 99

B

Backtracking, 109–111

C

Cron wrapper
- asynchronous interface, 60–63
- concurrent/parallel programs, 66–69
- extensions, 69
- higher-level primitives, 66
- mocking/testing, 72–78
- module installation, 79–80
- placing random points, 68
- random numbers, 68–69
- refactoring test, 70–72
- reliability/timing, 78–79
- run external commands, 60–63
- smart matching operator, 65
- STDOUT/STDERR streams, 61
- timeouts implementation, 64–66
- wrapped program, 59

Cro services
- authentication, 173–176
- command-line option, 164
- HTTP server, 164
- minimalistic web, 172
- parameter, 174
- signal() function, 163
- testing, 167–169
- UNIX timestamp, 161–162
- web page, 170–172

D, E

Datetime conversion
- automated tests, 46–51
- dealing time, 41
- debugging output, 33
- formatting, 36–38
- implicit variable/topic program, 42–44
- libraries, 32–33
- MAIN subs, 45
- numeric value, 33
- switch statement, 43
- timestamp, 38–41
- truncated-to method, 35
- UNIX timestamps, 31

M. Lenz, *Raku Fundamentals*, https://doi.org/10.1007/978-1-4842-6109-5

F

Flame graphs, 138–141
Functional programming
  argument, 148
  configuration, 146
  deterministic color generation, 147
  divide/conquer, 151
  enum/enumeration, 150
  flame graphs, 144
  file/directory classes, 149
  inner function, 145
  refactoring function, 145
  terminate function, 144
  tree/flame graph, 140–144

G

Grammars
  features, 112
  matches
    action class, 118
    capturing group, 115
    data structure, 113
    match tree, 115
    parse-ini method, 119
    rule method, 118
    submatches, 117
  regex definition, 113
Graph file/directory
  flame graphs, 138–141
  function (*see* Functional programming)
  reading file sizes, 129–132
  tree-map, 132–137

H, I, J, K, L, M, N, O

heredoc, 85

P, Q

Package application
  Dockerfile, 180
  modules/software, 179
  Windows installation, 180
  zef module, 178
Paring INI files, 99
  backtracking, 110–112
  error messages, 120
    context, 123–125
    expect method, 124
    failure, 120–121
    goal-matching syntax, 125–126
    parsing error messages, 122–123
    tracer module, 121
  grammars (*see* Grammars)
  key/value pairs/comment lines, 108
  primitives, 105–109
  regexes (*see* Regexes)
  stand-alone regex, 108
Perl Compatible Regular Expressions (PCRE), 101

R

Rakudo
  Docker, 9–11
  documentation, 12
  installation process, 7–8
  module installer, 11
  PATH environment, 8–9
  source code, 10–11
  testing, 11
Raku programming language
  classes/object declarations, 94–96
  concurrency/parallel execution, 96
  conservative Perl 5 developers, 3
  history of, 4
  intended audience, 2
  libraries, 4
  Perl influence, 1
  subroutines, 92–94
  technical benefits, 5
  variables/scoping, 91
Regexes
  character classes, 102–103
  either/alternatives, 104
  parsing value pairs, 100–101
  quantifiers, 103–104
  regular expressions, 100

S

Scaling code base, 177
Sink context, 76
Storage back end
  API code, 87–88
  database handle, 82
  development, 83–87
  expansion, 89
  insert method, 84
  persistent storage, 81–83
  statement handle, 82
Sudoku puzzle
  ASCII code, 18
  constant, 23
  for loop, 16
  horizontal lines, 21
  I/O operation/tragedies, 24
  join method, 20
  lexical variable declaration, 15
  loop statement, 20
  ++ postfix operator, 22
  Raku program, 14
  shortcuts, 22
  solving puzzle, 17
  string substitution, 18
  substr function, 16
  SVG text-based vector graphics, 27–29
  UNIX programs, 26
  unsolved form, 13
  working process, 19

T

Test Anything Protocol (TAP), 48
Testing say()
  array variable, 56

Testing say() (*cont.*)
dynamic variables, 56
IO:Handle declaration, 55
OutputCapture, 57
ready-made module, 53–54

U, V

Unicode characters tool
bytes, 157
chr method, 154–155
code points, 154–156
collation algorithm, 159
grapheme clusters, 156
number literals, 158
properties, 159
search string, 153
uni tool, 153

W, X, Y, Z

Web service *See* Cro services