

# Index

- ❖ xx
- \? 139
- \<..\> 21, 25, 50, 131-132, 150
  - in *egrep* 15
  - in Emacs 100
  - mimicking in Perl 341-342
- \+ 139
- \(..\) 135
- '\+' history 87
- \0 116-117
- \1 136, 300, 303
  - (also see backreferences)
  - in Perl 41
- \A 111, 127-128
  - (also see enhanced line-anchor mode)
  - in Java 373
  - optimization 246
- \a 114-115
- \b 65, 114-115, 400
  - (also see: word boundaries; backspace)
  - backspace and word boundary 44, 46
  - in Perl 286
- \b\B 240
- \c 328
- \d 49, 119
  - in Perl 288
- \e 79, 114-115
- \E 290
  - (also see literal-text mode)
- \f 114-115
  - introduced 44
- \G 128-131, 212, 315-316, 362
  - (also see pos)
  - advanced example 130
  - in Java 373
  - in .NET 402
  - optimization 246
- \kname (see named capture)
- \l 290
- \L..\E 290
  - inhibiting 292
- \n 49, 114-115
  - introduced 44
  - machine-dependency 114
- \N{LATIN SMALL LETTER SHARP S} 290
- \N{name} 290
  - (also see pragma)
  - inhibiting 292
- \p{...} 119
  - (also see Unicode, properties)
- \p{^...} 288
- \p{a11} 380
- \p{A11} 123
  - in Perl 288
- \p{Any} 123
  - in Perl 288
- \p{Arrows} 122
- \p{Assigned} 123-124
  - in Perl 288
- \p{Basic\_Latin} 122
- \p{Box\_Drawing} 122
- \p{C} 120
- \p{Cc} 121
- \p{Cf} 121
- \p{Cherokee} 120

<code>\p{Close_Punctuation}</code>	121	<code>\p{Lo}</code>	121, 400
<code>\p{Cn}</code>	121, 123-124, 380, 401	<code>\p{Lowercase_Letter}</code>	121
<code>\p{Co}</code>	121	<code>\p{Lt}</code>	121, 400
<code>\p{Connector_Punctuation}</code>	121	<code>\p{Lu}</code>	121, 400
<code>\p{Control}</code>	121	<code>\p{M}</code>	120, 125
<code>\p{Currency}</code>	122	<code>\p{Mark}</code>	120
<code>\p{Currency_Symbol}</code>	121	<code>\p{Math_Symbol}</code>	121
<code>\p{Cyrillic}</code>	120, 122	<code>\p{Mc}</code>	121
<code>\p{Dash_Punctuation}</code>	121	<code>\p{Me}</code>	121
<code>\p{Decimal_Digit_Number}</code>	121	<code>\p{Mn}</code>	121
<code>\p{Dingbats}</code>	122	<code>\p{Modifier_Letter}</code>	121
<code>\p{Enclosing_Mark}</code>	121	<code>\p{Modifier_Symbol}</code>	121
<code>\p{Final_Punctuation}</code>	121	<code>\p{N}</code>	120, 390
<code>\p{Format}</code>	121	<code>\p{Nd}</code>	121, 380, 400
<code>\p{Gujarati}</code>	120	<code>\p{Nl}</code>	121
<code>\p{Han}</code>	120	<code>\p{No}</code>	121
<code>\p{Hangul_Jamo}</code>	122	<code>\p{Non_Spacing_Mark}</code>	121
<code>\p{Hebrew}</code>	120, 122	<code>\p{Number}</code>	120
<code>\p{Hiragana}</code>	120	<code>\p{Open_Punctuation}</code>	121
<code>\p{InArrows}</code>	122	<code>\p{Other}</code>	120
<code>\p{InBasic_Latin}</code>	122	<code>\p{Other_Letter}</code>	121
<code>\p{InBox_Drawing}</code>	122	<code>\p{Other_Number}</code>	121
<code>\p{InCurrency}</code>	122	<code>\p{Other_Punctuation}</code>	121
<code>\p{InCyrillic}</code>	122	<code>\p{Other_Symbol}</code>	121
<code>\p{InDingbats}</code>	122	<code>\p{P}</code>	120
<code>\p{InHangul_Jamo}</code>	122	<code>\p{Paragraph_Separator}</code>	121
<code>\p{InHebrew}</code>	122	<code>\p{Pc}</code>	121, 400
<code>\p{Inherited}</code>	122	<code>\p{Pd}</code>	121
<code>\p{Initial_Punctuation}</code>	121	<code>\p{Pe}</code>	121
<code>\p{InKatakana}</code>	122	<code>\p{Pf}</code>	121, 400
<code>\p{InTamil}</code>	122	<code>\p{Pi}</code>	121, 400
<code>\p{InTibetan}</code>	122	<code>\p{Po}</code>	121
<code>\p{IsCherokee}</code>	120	<code>\p{Private_Use}</code>	121
<code>\p{IsCommon}</code>	122	<code>\p{Ps}</code>	121
<code>\p{IsCyrillic}</code>	120	<code>\p{Punctuation}</code>	120
<code>\p{IsGujarati}</code>	120	<code>\p{S}</code>	120
<code>\p{IsHan}</code>	120	<code>\p{Sc}</code>	121-122
<code>\p{IsHebrew}</code>	120	<code>\p{Separator}</code>	120
<code>\p{IsHiragana}</code>	120	<code>\p{Sk}</code>	121
<code>\p{IsKatakana}</code>	120	<code>\p{Sm}</code>	121
<code>\p{IsLatin}</code>	120	<code>\p{So}</code>	121
<code>\p{IsThai}</code>	120	<code>\p{Space_Separator}</code>	121
<code>\p{IsTibetan}</code>	122	<code>\p{Spacing_Combining_Mark}</code>	121
<code>\p{Katakana}</code>	120, 122	<code>\p{Symbol}</code>	120
<code>\p{L}</code>	119-120, 131, 380, 390	<code>\p{Tamil}</code>	122
<code>\p{L&amp;}</code>	120-121, 123	<code>\p{Thai}</code>	120
in Perl	288	<code>\p{Tibetan}</code>	122
<code>\p{Latin}</code>	120	<code>\p{Titlecase_Letter}</code>	121
<code>\p{Letter}</code>	120, 288	<code>\p{Unassigned}</code>	121, 123
<code>\p{Letter_Number}</code>	121	in Perl	288
<code>\p{Line_Separator}</code>	121	<code>\p{Uppercase_Letter}</code>	121
<code>\p{Ll}</code>	121, 400	<code>\p{Z}</code>	119-120, 380, 400
<code>\p{Lm}</code>	121, 400	<code>\p{Zl}</code>	121

- `\p{Zp}` 121
- `\p{Zs}` 121
- `\Q·\E` 290
  - inhibiting 292
  - in Java 373
- `\r` 49, 114-115
  - machine-dependency 114
- `\s` 49, 119
  - introduction 47
  - in Emacs 127
  - in Perl 288
- `\S` 49, 56, 119
- `\t` 49, 114-115
  - introduced 44
- `\u` 116, 290, 400
- `\U` 116
- `\U·\E` 290
  - inhibiting 292
- `\V` 364
- `\v` 114-115, 364
- `\W` 49, 119
- `\w` 49, 65, 119
  - in Emacs 127
  - many different interpretations 93
  - in Perl 288
- `\x` 116, 400
  - in Perl 286
- `\X` 107, 125
- `\z` 111, 127-128, 316
  - (also see enhanced line-anchor mode)
  - in Java 373
  - optimization 246
- `\Z` 111, 127-128
  - (also see enhanced line-anchor mode)
  - in Java 373
  - optimization 246
- `//` 322
- `/c` 129-130, 315
- `/e` 319-321
- `/g` 61, 130, 307, 311-312, 315, 319
  - (also see `\G`)
  - introduced 51
  - with regex object 354
- `/i` 134
  - (also see: case-insensitive mode; mode modifier)
  - introduced 47
  - with `study` 359
- `/m` 134
  - (also see: enhanced line-anchor mode; mode modifier)
- `/o` 352-353
  - with regex object 354
- `/osmosis` 293
- `/s` 134
  - (also see: dot-matches-all mode; mode modifier)
- `/x` 134, 288
  - (also see: comments and free-spacing mode; mode modifier)
  - introduced 72
  - history 90
- `-Dr` 363
- `-i` as `-y` 86
- `-Mre=debug` (see use `re 'debug'`)
- `-y` old *grep* 86
- `<>` 54
  - and `$_` 79
- `!~` 309
- `#` (see comments)
- `$_` 79, 308, 311, 314, 318, 322, 353-354, 359
  - in .NET 418
- `$&` 299-300
  - checking for 358
  - mimicking 302, 357
  - naughty 356
  - in .NET 418
  - okay for debugging 331
  - pre-match copy 355
- `$$` in .NET 418
- `$*` 362
- `$` 111-112, 128
  - (also see enhanced line-anchor mode)
  - escaping 77
  - optimization 246
  - Perl interpolation 289
- `$+` 300-301, 345
  - example 202
  - .NET 202
  - in .NET 418
- `$/` 35, 78
- `$'` 300
  - checking for 358
  - mimicking 357
  - naughty 356
  - in .NET 418
  - okay for debugging 331
  - pre-match copy 355
- `$`` 300
  - checking for 358
  - mimicking 357
  - naughty 356
  - in .NET 418
  - okay for debugging 331
  - pre-match copy 355

- \$0 300
- \$-[0] (see @-)
- \$+[0] (see @+)
- \$1 135-136, 300, 303
  - introduced 41
  - in Java 388
  - in .NET 418
  - in other languages 136
  - pre-match copy 355
- \$ARGV 79
- \$HostnameRegex 76, 136, 303, 351
- \$HttpUrl 303, 305, 345, 351
- \$LevelN 330, 343
- \$^N 300-301, 344-346
- \${name} 403
- \${name~} 418
- \$NestedStuffRegex 339, 346
- \$^R 302, 327
- \$^W 297
- % Perl interpolation 289
- (?! ) 240, 333, 335, 340-341
- (?# ..) 99, 134, 414
  - in Java 373
- (?: ..) (see non-capturing parentheses)
- (...) (see parentheses)
- (?i) (see: case-insensitive mode; mode modifier)
- (?i: ..) (see mode-modified span)
- (?if then | else) (see conditional)
- (?m: ..) (see mode-modified span)
- (?m) (see: enhanced line-anchor mode; mode modifier)
- (?n) 402
- (?'name' ..) (see named capture)
- (?<name> ..) (see named capture)
- (?P=name ..) (see named capture)
- (?P<name> ..) (see named capture)
- (?s: ..) (see mode-modified span)
- (?s) (see: dot-matches-all mode; mode modifier)
- (?x: ..) (see mode-modified span)
- (?x) (see: comments and free-spacing mode; mode modifier)
- \* (see star)
- \*\* (see possessive quantifiers)
- ++ (see possessive quantifiers)
- + (see plus)
- ".\*" (see double-quoted string example)
- .\*
  - introduced 55
  - mechanics of matching 152
  - optimization 246
  - warning about 56
- .NET 399-432
  - \$+ 202
  - flavor overview 91
  - after-match data 136
  - benchmarking 236
  - JIT 404
  - line anchors 128
  - literal-text mode 135
  - MISL 404
  - object model 411
  - regex approach 96-97
  - regex flavor 401
  - search-and-replace 408, 417-418
  - URL parsing example 204
  - version covered 91
  - word boundaries 132
- =~ 308-309, 318
  - introduced 38
- ? (see question mark)
- ?..? 308
- ?+ (see possessive quantifiers)
- @+ 300, 302, 314
- @"... " 102
- @- 300, 302, 339
- @ Perl interpolation 289
- [=..=] 126
- [:<:] 92
- [:::] 125-126
- [... ] 126
- \p{...} in java.util.regex 380
- ^ 111-112, 128
  - (also see enhanced line-anchor mode)
  - optimization 245-246
- ^Subject: **example** 94, 151-152, 154, 242, 244-245, 289
  - in Java 95, 393
  - in Perl 55
  - in Perl debugger 361
  - in Python 97
  - in VB.NET 96
- {min,max} 20, 140
- | (see alternation)
- \0 116-117
- \$0 300
- \1 136, 300, 303
  - (also see backreferences)
  - in Perl 41
- \$1 135-136, 300, 303
  - introduced 41
  - in Java 388
  - in .NET 418

- \$1 (cont'd)
  - in other languages 136
  - pre-match copy 355
- 8859-1 encoding** 29, 87, 105, 107, 121
- \A** 111, 127-128
  - (also see enhanced line-anchor mode)
  - in Java 373
  - optimization 246
- @** escaping 77
- \a** 114-115
- issues overview encoding** 105
- after-match variables**
  - in Perl 299
  - pre-match copy 355
- Aho, Alfred** 86, 180
- \p{All}** 123
  - in Perl 288
- \p{all}** 380
- all-in-one object model** 369
- alternation** 138
  - and backtracking 231
  - introduced 13-14
  - efficiency 222, 231
  - greedy 174-175
  - hand tweaking 260-261
  - order of 175-177, 223, 260
    - for correctness 28, 189, 197
    - for efficiency 224
  - and parentheses 13
- analogy**
  - backtracking
    - bread crumbs 158-159
    - stacking dishes 159
  - ball rolling 261
  - building a car 31
  - charging batteries 179
  - engines 143-147
  - first come, first served 153
  - gas additive 150
  - learning regexes
    - Pascal 36
    - playing rummy 33
  - regex as a language 5, 27
  - regex as filename patterns 4
  - regex-directed match (see NFA)
  - text-directed match (see DFA)
  - transmission 148-149, 228
  - transparencies (Perl's `local`) 298
- anchor** (also see: word boundaries; enhanced line-anchor mode)
  - overview 127
  - caret 127
  - dollar 127
  - end-of-line optimization 246
  - exposing 255
  - line 87, 111-112, 150
- anchored(...)** 362
- anchored 'string'** 362
- AND** class set operations 123-124
- ANSI escape sequences** 79
- \p{Any}** 123
  - in Perl 288
- any character** (see dot)
- Apache**
  - Jakarta (see ORO)
  - `org.apache.xerces.utils.regex` 372
  - ORO 392-398
    - benchmark results 376
    - comparative description 374
  - Regexp
    - comparative description 375
    - speed 376
- appendReplacement()** 388
- appendTail()** 389
- \$ARGV** 79
- array context** (see list context)
- \p{Arrows}** 122
- ASCII encoding** 29, 105-106, 114, 121
- Asian character encoding** 29
- AssemblyName** 429
- \p{Assigned}** 123-124
  - in Perl 288
- asterisk** (see star)
- atomic grouping** (also see possessive quantifiers)
  - introduced 137-138
  - details 170-172
  - for efficiency 171-172, 259, 268-270
  - essence 170-171
  - example 198, 201, 213, 271, 330, 340-341, 346
- AT&T Bell Labs** 86
- auto-lookaheadification** 403
- automatic possessification** 251
- awk**
  - after-match data 136
  - `gensub` 183
  - history 87
  - search-and-replace 99
  - version covered 91
  - word boundaries 132

- `\b` 65, 114-115, 400
  - (also see: word boundaries; backspace)
  - backspace and word boundary 44, 46
  - in Perl 286
- ß 110, 126, 290, 366
- `<B>` `</B>` 165-167
  - unrolling 270
- `\b\B` 240
- backreferences** 117, 135
  - introduced with *egrep* 20-22
  - DFA 150, 182-183
  - vs. octal escape 406-407
  - remembering text 21
- backspace** (see `\b`)
- backtracking** 163-177
  - introduction 157-163
  - and alternation 231
  - avoiding 171-172
  - computing count 227
  - counting 222, 224
  - detecting excessive 249-250
  - efficiency 179-180
  - essence 168-169
  - exponential match 226
  - global view 228-232
  - LIFO 159
  - of lookaround 173-174
  - neverending match 226
  - non-match example 160-161
  - POSIX NFA example 229
  - saved states 159
  - simple example 160
  - simple lazy example 161
- balanced constructs** 328-331, 340-341, 430
- balancing regex issues** 186
- Balling, Derek** xxii
- Barwise, J.** 85
- base character** 107, 125
- Basic Regular Expressions** 87-88
  - `\p{Basic_Latin}` 122
  - `\b\B` 240
- `beginOffset` 396
- benchmarking** 232-239
  - comparative 248, 376-377
  - compile caching 351
  - in Java 234-236, 375-377
  - for naughty variables 358
  - in .NET 236, 404
  - with neverending match 227
  - in Perl 360
  - pre-match copy 356
  - in Python 237
  - in Ruby 238
- benchmarking** (cont'd)
  - in Tcl 239
- Bennett, Mike** xxi
- Berkeley** 86
- Better-Late-Than-Never** 234-236, 375
  - `<B>` `</B>` 165-167
  - unrolling 270
- blocks** 122, 288, 380, 400
- BLTN** 235-236, 375
- BOL** 362
- `\p{Box_Drawing}` 122
- Boyer-Moore** 244, 247
- brace** (see interval)
- bracket expressions** 125
- BRE** 87-88
- bread-crumble analogy** 158-159
- Bulletin of Math. Biophysics* 85
- bump-along**
  - introduction 148-149
  - avoiding 210
  - distrusting 215-218
  - optimization 255
  - in overall processing 241
- byte** matching 328
  - `/c` 129-130, 315
- C#** (also see .NET)
  - strings 102
- `\p{C}` 120
- `\C` 328
- € 122
- C comments**
  - matching 272-276
  - unrolling 275-276
- caching** (also see regex objects)
  - benchmarking 351
  - compile 242-244
  - in Emacs 244
  - integrated 242
  - in Java 393
  - in .NET 426
  - object-oriented 244
  - procedural 243
  - in Tcl 244
  - unconditional 350
- CANON\_EQ (Pattern flag)** 108, 380
- Capture** 431
- CaptureCollection** 432
- car analogy** 83-84
- caret anchor** introduced 8
- carriage return** 109
- case** title 109

- case folding 290, 292
  - inhibiting 292
- CASE\_INSENSITIVE (Pattern flag) 95, 109, 380, 383
- case-insensitive mode 109
  - introduced 14-15
  - egrep* 14-15
  - /i* 47
  - Ruby 109
  - with *study* 359
- cast 294-295
- categories (see Unicode, properties)
- `\p{Cc}` 121
- Celsius (see temperature conversion example)
- `\p{Cf}` 121
- character
  - base 125
  - classes 117
  - combining 107, 125, 288
    - Inherited script 122
  - vs. combining characters 107
  - control 116
  - initial character discrimination 244-246, 249, 251-252, 257-259, 332, 361
  - machine-dependent codes 114
  - multiple code points 107
  - as opposed to byte 29
  - separating with *split* 322
  - shorthands 114-115
- character class
  - introduced 9-10
  - vs. alternation 13
  - mechanics of matching 149
  - negated
    - must match character 11-12
    - and newline 118
    - Tcl 111
  - positive assertion 118
  - of POSIX bracket expression 125
  - range 9, 118
  - as separate language 10
- character equivalent 126
- CharacterIterator 372
- charnames pragma 290
- CharSequence 372, 390
- CheckNaughtiness 358
- `\p{Cherokee}` 120
- Chinese text processing 29
- chr 414
- chunk limit
  - Java ORO 395
  - java.util.regex 391
- chunk limit (cont'd)
  - Perl 323
- class
  - vs. dot 118
  - elimination optimization 249
  - initial class discrimination 244-246, 249, 251-252, 257-259, 332, 361
  - and lazy quantifiers 167
  - set operations 123-125, 375
  - subtraction 124
- Clemens, Sam 375
- Click, Cliff xxii
- client VM 234, 236
- clock clicks 239
- `\p{Close_Punctuation}` 121
- closures 339
- `\p{Cn}` 121, 123-124, 380, 401
- `\p{Co}` 121
- code point
  - introduced 106
  - beyond U+FFFF 108
  - multiple 107
  - unassigned in block 122
- coerce 294-295
- cold VM 235
- collating sequences 126
- combining character 107, 125, 288
  - Inherited script 122
- com.ibm.regex
  - comparative description 372
  - speed 377
- commafying a number example 64-65
  - introduced 59
  - in Java 393
  - without lookbehind 67
- COMMAND.COM 7
- comments 99, 134
  - in Java 98
  - matching of C comments 272-276
  - matching of Pascal comments 265
  - in .NET regex 414
- COMMENTS (Pattern flag) 99, 218, 378, 380, 386
- comments and free-spacing mode 110
- Communications of the ACM* 85
- comparison of engine types (see NFA)
- compile() 383
- compile
  - caching 242-244
  - once (/o) 352-353
  - on-demand 351
  - regex 404-405
- compile() (Pattern factory) 383

- Compiled (.NET) 236, 402, 404, 414, 421-422, 429
- Compilers – Principles, Techniques, and Tools* 180
- CompileToAssembly 427, 429
- com.stevesoft.pat
  - comparative description 374
  - speed 377
- conditional** 138-139
  - with embedded regex 327, 335
  - in Java 373
  - mimicking with lookaround 139
  - in .NET 403
- Config module 290, 299
- conflicting metacharacters** 44-46
  - \p{Connector\_Punctuation} 121
- Constable, Robert 85
- context** (also see: list context; scalar context; match, context)
  - forcing 310
  - metacharacters 44-46
  - regex use 189
- continuation lines** 178, 186-187
  - unrolling 270
- contorting an expression** 294-295
  - \p{Control} 121
- control characters** 116
- Conway, Damian 339
- cooking for HTML 68, 408
- copy for \$& (see pre-match copy)
- correctness** vs. efficiency 223-224
- counting quantifier** (see interval)
- www.cpan.org 358
- CR 109, 382
- Cruise, Tom 51
- crummy analogy** 158-159
- CSV parsing example**
  - java.util.regex 218, 386
  - .NET 429
  - ORO 397
  - Perl 212-219
  - unrolling 271
- currency**
  - € 121-122, 379, 400
  - \p{Currency} 122
  - \p{Currency\_Symbol} 121
  - \p{Sc} 121
  - Unicode block 121-122
  - \p{Currency} 122
  - \p{Currency\_Symbol} 121
- currentTimeMillis() 236
- \p{Cyrillic} 120, 122
- \d 49, 119
  - in Perl 288
- \D 49, 119
- Darth 197
- dash in character class** 9
  - \p{Dash\_Punctuation} 121
- DBIx::DWIW 258
- debugcolor 363
- debugging** 361-363
  - with embedded code 331-332
  - regex objects 305-306
  - run-time 362
- \p{Decimal\_Digit\_Number} 121
- default regex** 308
- define-key 100
- delegate** 417-418
- delimited text** 196-198
  - standard formula 196, 273
- delimiter**
  - with shell 7
  - with substitution 319
- Deterministic Finite Automaton** (see DFA)
- Devel::FindAmpersand 358
- Devel::SawAmpersand 358
- DFA**
  - introduced 145, 155
  - acronym spelled out 156
  - backreferences 150, 182-183
  - boring 157
  - compared with NFA 224, 227 (also see NFA)
  - efficiency 179
  - implementation ease 182
  - lazy evaluation 181
  - longest-leftmost match 177-179
  - testing for 146-147
  - in theory, same as an NFA 180
- dialytika 108
- \p{Dingbats} 122
- directed alternation** (see alternation, ordered)
- dish-stacking analogy** 159
- dollar** for Perl variable 37
- dollar anchor** 127
  - introduced 8
- dollar value example** 24-25, 51-52, 167-170, 175, 194-195
- DOS 7
- dot** 118
  - introduced 11-12
  - vs. character class 118
  - mechanics of matching 149
  - Tcl 112



- .NET 399-432
  - \$+ 202
  - flavor overview 91
  - after-match data 136
  - benchmarking 236
  - JIT 404
  - line anchors 128
  - literal-text mode 135
  - MISL 404
  - object model 411
  - regex approach 96-97
  - regex flavor 401
  - search-and-replace 408, 417-418
  - URL parsing example 204
  - version covered 91
  - word boundaries 132
- DOTALL (Pattern flag) 380, 382
- dot-matches-all mode 110-111
- doubled-word example**
  - description 1
  - in *egrep* 22
  - in Emacs 100
  - in Java 81
  - in Perl 35, 77-80
- double-quoted string example**
  - allowing escaped quotes 196
  - egrep* 24
  - final regex 263
  - makudonarudo 165, 169, 228-232, 264
  - sobering example 222-228
  - unrolled 262, 268
- double-word finder example**
  - description 1
  - in *egrep* 22
  - in Emacs 100
  - in Java 81
  - in Perl 35, 77-80
- Dr 363
- dragon book 180
- DWIW (DBIX) 258
- dynamic regex 327-331
  - sanitizing 337
- dynamic scope 295-299
  - vs. lexical scope 299
- /e 319-321
- \e 79, 114-115
- \E 290
  - (also see literal-text mode)
- earliest match wins 148-149
- EBCDIC 29
- ECMAScript (.NET) 400, 402, 406-407, 415, 421
- ed* 85
- efficiency** (also see optimization)
  - and backtracking 179-180
  - correctness 223-224
  - Perl-specific issues 347-363
  - regex objects 353-354
  - unlimited lookbehind 133
- egrep*
  - flavor overview 91
  - introduced 6-8
  - metacharacter discussion 8-22
  - after-match data 136
  - backreference support 150
  - case-insensitive match 15
  - doubled-word solution 22
  - example use 14
  - flavor summary 32
  - history 86-87
  - regex implementation 182
  - version covered 91
  - word boundaries 132
- electric engine analogy** 143-147
- else (see conditional)
- Emacs**
  - flavor overview 91
  - after-match data 136
  - control characters 116
  - re-search-forward 100
  - search 100
  - strings as regexes 100
  - syntax class 127
  - version covered 91
  - word boundaries 132
- email address example** 70-73, 98
  - in Java 98
  - in VB.NET 99
- embedded code**
  - local 336
  - my 338-339
  - regex construct 327, 331-335
  - sanitizing 337
- embedded string check optimization** 247, 257
- Embodiments of Mind* 85
- Empty 426
- encapsulation** (see regex objects)
- \p{Enclosing\_Mark} 121
- encoding** (also see Unicode)
  - introduced 29
  - issues overview 105
  - ASCII 29, 105-106, 114, 121

**encoding** (cont'd)

- Latin-1 29, 87, 105, 107, 121
- UCS-2 106
- UCS-4 106
- UTF-16 106
- UTF-8 106

`end()` 385

**END block** 358

**end of line** (see `anchor`, `dollar`)

**end of previous match** (see `\G`)

**end of word** (see `word boundaries`)

`endOffset` 396

**end-of-string anchor optimization** 246

**engine**

- introduced 27
- analogy 143-147
- hybrid 183, 239, 243
- implementation ease 182
- testing type 146-147
  - with neverending match 227
  - type comparison 156-157, 180-182

**English module** 357

**English vs. regex** 275

**enhanced line-anchor mode** 111-112

- introduced 69

**ERE** 87-88

**errata** xxi

**Escape** 427

**escape**

- introduced 22
- term defined 27

**essence**

- atomic grouping 170-171
- greediness, laziness, and backtracking 168-169
- NFA (see `backtracking`)

`eval` 319

**example**

- atomic grouping 198, 201, 213, 271, 330, 340-341, 346
- commafying a number 64-65
  - introduced 59
  - in Java 393
  - without lookbehind 67
- CSV parsing
  - `java.util.regex` 218, 386
  - `.NET` 429
  - `ORO` 397
  - `Perl` 212-219
  - unrolling 271
- `dollar` value 24-25, 51-52, 167-170, 175, 194-195

**example** (cont'd)

- double-quoted string
  - allowing escaped quotes 196
  - `egrep` 24
  - final regex 263
  - `makudonarudo` 165, 169, 228-232, 264
  - sobering example 222-228
  - unrolled 262, 268
- double-word finder
  - description 1
  - in `egrep` 22
  - in Emacs 100
  - in Java 81
  - in Perl 35, 77-80
- email address 70-73, 98
  - in Java 98
  - in VB.NET 99
- filename 190-192
- five modifiers 316
- floating-point number 194
- form letter 50-51
- `gr[ea]y` 9
- hostname 22, 73, 76, 98-99, 136-137, 203, 260, 267, 304, 306
  - `egrep` 25
  - Java 209
  - plucking from text 71-73, 205-208
  - in a URL 74-77
  - validating 203-205
  - VB.NET 204
- HTML
  - conversion from text 67-77
  - cooking 68, 408
  - encoding 408
  - `<HR>` 194
  - link 201-203
  - optional 139
  - paired tags 165
  - parsing 130, 315, 321
  - tag 9, 18-19, 26, 200-201, 326, 357
  - URL 74-77, 203, 205-208, 303
  - URL-encoding 320
- IP 5, 187-189, 267, 311, 314, 348-349
- `Jeffs` 61-64
- lookahead 61-64
- mail processing 53-59
- `makudonarudo` 165, 169, 228-232, 264
- pathname 190-192
- population 59
- possessive quantifiers 198, 201
- postal code 208-212
- regex overloading 341-345

**example** (cont'd)

- stock pricing 51-52, 167-168
  - with alternation 175
  - with atomic grouping 170
  - with possessive quantifier 169
- temperature conversion
  - in .NET 419
  - in Java 389
  - in Perl 37
  - Perl one-liner 283
- text-to-HTML 67-77
- this|that 132, 138, 243, 245-246, 252, 255, 260-261
- unrolling the loop 270-271
- URL 74-77, 201-204, 208, 260, 303-304, 306, 320
  - egrep* 25
  - Java 209
  - plucking 205-208
- username 73, 76, 98
  - plucking from text 71-73
  - in a URL 74-77
- variable names 24
- ZIP code 208-212

**exception**

- `IllegalArgumentException` 383, 388
- `IllegalStateException` 385
- `IndexOutOfBoundsException` 384-385, 388
- `IOException` 81
- `NullPointerException` 396
- `PatternSyntaxException` 381, 383

`Explicit (Option)` 409

`ExplicitCapture (.NET)` 402, 414, 421

**exponential match** 222-228, 330, 340

- avoiding 264-265
- discovery 226-228
- explanation 226-228
- non-determinism 264
- short-circuiting 250
- solving with atomic grouping 268
- solving with possessive quantifiers 268

**expose literal text** 255

**expression**

- context 294-295
- contorting 294-295

**Extended Regular Expressions** 87-88

`\f` 114-115

- introduced 44

**Fahrenheit** (see temperature conversion example)

**failure**

- atomic grouping 171-172
- forcing 240, 333, 335, 340-341

**FF** 109

**file globs** 4

**file-check example** 2, 36

**filename**

- example 190-192
- patterns (*globs*) 4
- prepending to line 79

`\p{Final_Punctuation}` 121

`find()` 384

`FindAmpersand` 358

**five modifiers example** 316

**Flanagan, David** xxii

**flavor**

- Perl 286-293
- superficial chart
  - general 91
  - Perl 285, 287
  - POSIX 88
- term defined 27

*flex* version covered 91

**floating regex cache** (see regex objects)

floating *'string'* 362

**floating-point number example** 194

**forcing failure** 240, 333, 335, 340-341

foreach vs. while vs. if 320

**form letter example** 50-51

`\p{Format}` 121

**freeflowing regex** 277-281

**Friedl, Alfred** 176

**Friedl, brothers** 33

**Friedl, Fumie** xxi

- birthday 11-12

**Friedl, Liz** 33

**Friedl, Stephen** xxii

**fully qualified name** 295

**functions** related to regexes in Perl 285

`\G` 128-131, 212, 315-316, 362

- (also see `pos`)
- advanced example 130
- in Java 373
- in .NET 402
- optimization 246

`/g` 61, 130, 307, 311-312, 315, 319

- (also see `\G`)
- introduced 51
- with regex object 354

**garbage collection** Java benchmarking 236

- gas engine analogy 143-147
- general categories (see Unicode, properties)
- gensub 183
- George, Kit xxii
- GetGroupNames 421-422
- GetGroupNumbers 421-422
- getMatch() 397
- global match (see /g)
- global vs. private Perl variables 295
- globs filename 4
- GNU Java packages 374
- GNU awk
  - after-match data 136
  - gensub 183
  - version covered 91
  - word boundaries 132
- GNU *egrep*
  - after-match data 136
  - backreference support 150
  - doubled-word solution 22
  - i bug 21
  - regex implementation 182
  - word boundaries 132
- GNU Emacs (see Emacs)
- GNU *grep*
  - shortest-leftmost match 183
  - version covered 91
- GNU sed
  - after-match data 136
  - version covered 91
  - word boundaries 132
- gnu.regexp
  - comparative description 374
  - speed 377
- gnu.rex 374
- Goldberger, Ray xxii
- Gosling, James 89
- GPOS 362
- gr[ea]y example 9
- greedy (also see lazy)
  - introduced 151
  - alternation 174-175
  - and backtracking 162-177
  - deference to an overall match 153, 274
  - essence 159, 168-169
  - favors match 167-168
  - first come, first served 153
  - global vs. local 182
  - in Java 373
  - vs. lazy 169, 256-257
  - localizing 225-226
  - quantifier 139-140
- greedy (cont'd)
  - too greedy 152
- green dragon 180
- grep*
  - flavor overview 91
  - as an acronym 85
  - history 86
  - regex flavor 86
  - version covered 91
  - y option 86
- grep in Perl 324
- group()
  - java.util.regex 385
  - ORO 396
- Group object (.NET) 412
  - Capture 431
  - creating 423
  - Index 424
  - Length 424
  - Success 424
  - ToString 424
  - using 424
  - Value 424
- GroupCollection 423, 432
- groupCount() 385
- grouping and capturing 20-22
- grouping-only parentheses (see non-capturing parentheses)
- GroupNameFromNumber 421-422
- GroupNumberFromName 421-422
- groups() ORO 397
- Groups Match object method 423
- \p{Gujarati} 120
- Gutierrez, David xxii
- \p{Han} 120
- hand tweaking
  - alternation 260-261
  - caveats 253
- \p{Hangul\_Jamo} 122
- HASH(0x80f60ac) 257
- \p{Hebrew} 120, 122
- Hz 108
- hex escape 116-117
  - in Java 373
  - in Perl 286
- Hietaniemi, Jarkko xxii
- highlighting with ANSI escape sequences 79
- \p{Hiragana} 120

- history**
  - '\+' 87
  - AT&T Bell Labs 86
  - awk 87
  - Berkeley 86
  - ed* trivia 86
  - egrep* 86-87
  - grep* 86
  - lex* 87
  - Perl 88-90, 308
  - of regexes 85-91
  - sed* 87
  - underscore in `\w` 89
  - `/x` 90
- hostname example** 22, 73, 76, 98-99, 136-137, 203, 260, 267, 304, 306
  - egrep* 25
  - Java 209
  - plucking from text 71-73, 205-208
  - in a URL 74-77
  - validating 203-205
  - VB.NET 204
- `$HostnameRegex` 76, 136, 303, 351
- hot VM** 235, 375
- HTML**
  - cooking 68, 408
  - matching tag 200-201
- HTML example**
  - conversion from text 67-77
  - cooking 68, 408
  - encoding 408
  - `<HR>` 194
  - link 201-203
  - optional 139
  - paired tags 165
  - parsing 130, 315, 321
  - tag 9, 18-19, 26, 200-201, 326, 357
  - URL 74-77, 203, 205-208, 303
  - URL-encoding 320
- HTTP** newlines 115
- HTTP URL example** 25, 74-77, 201-209, 260, 303-304, 306, 320
- `http://regex.info/` xxi, 7, 345, 372
- `$HttpRequest` 303, 305, 345, 351
- hybrid regex engine** 183, 239, 243
- hyphen in character class** 9
- `Hz` 108
  
- `-i` as `-y` 86
- `/i` 134
  - (also see: case-insensitive mode; mode modifier)
- `/i` (cont'd)
  - introduced 47
  - with `study` 359
  - (`?i`) (see: case-insensitive mode; mode modifier)
- IBM (Java package)**
  - comparative description 372
  - speed 377
- identifier** matching 24
- if** (see conditional)
- if** vs. **while** vs. **foreach** 320
- (`?if then | else`) (see conditional)
- `IgnoreCase (.NET)` 96, 99, 402, 413, 421
- `IgnorePatternWhitespace (.NET)` 99, 402, 413, 421
- `IllegalArgumentException` 383, 388
- `IllegalStateException` 385
- implementation of engine** 182
- implicit** 362
- implicit anchor optimization** 246
- `Imports` 407, 409, 428
- `\p{InArrows}` 122
- `\p{InBasic_Latin}` 122
- `\p{InBox_Drawing}` 122
- `\p{InCurrency}` 122
- `\p{InCyrillic}` 122
- Index**
  - Group object method 424
  - Match object method 423
- `IndexOutOfBoundsException` 384-385, 388
- `\p{InDingbats}` 122
- indispensable** TiVo 3
- `\p{InHangul_Jamo}` 122
- `\p{InHebrew}` 122
- `\p{Inherited}` 122
- initial class discrimination** 244-246, 249, 251-252, 257-259, 332, 361
- `\p{Initial_Punctuation}` 121
- `\p{InKatakana}` 122
- `\p{InTamil}` 122
- integrated handling** 94-95
  - compile caching 242
- interpolation** 288-289
  - introduced 77
  - caching 351
  - mimicking 321
  - in PHP 103
- INTERSECTION** class set operations 124
- interval** 140
  - introduced 20
  - `[X{0,0}`, 140
  - `\p{InTibetan}` 122

- IOException 81
- IP example 5, 187-189, 267, 311, 314, 348-349
- Iraq 11
- Is vs. In 120, 122-123
  - with java.util.regex 380
  - in .NET 401
  - in Perl 288
- \p{IsCherokee} 120
- \p{IsCommon} 122
- \p{IsCyrillic} 120
- \p{IsGujarati} 120
- \p{IsHan} 120
- \p{IsHebrew} 120
- \p{IsHiragana} 120
- \p{IsKatakana} 120
- \p{IsLatin} 120
- IsMatch (Regex object method) 415
- ISO-8859-1 encoding 29, 87, 105, 107, 121
- \p{IsThai} 120
- \p{IsTibetan} 122
  
- ĵ 110
- Jakarta (see Apache, ORO)
- Japanese
  - 正規表現は簡単だよ! 5
  - text processing 29
- “japhy” 246
- Java 365-398
  - (also see java.util.regex)
  - benchmarking 234-236
  - BLTN 235-236, 375
  - choosing a regex package 366
  - exposed mechanics 374
  - fastest package 377
  - JIT 235
  - list of packages 372
  - matching comments 272-276
  - object models 368-372
  - package flavor comparison 373
  - “Perl5 flavors” 375
  - strings 102
  - version covered 91
  - VM 234-236, 375
- java.util.regex 95-96, 378-391
  - after-match data 136
  - code example 383, 389
  - comparative description 372
  - CSV parsing 386
  - dot modes 111
  - doubled-word example 81
  - line anchors 128
- java.util.regex (cont'd)
  - line terminators 382
  - match modes 380
  - object model 381
  - regex flavor 378-381
  - search-and-replace 387
  - speed 377
  - split 390
  - URL parsing example 209
  - version covered 91
  - word boundaries 132
- Jeffs example 61-64
- JfiedlsRegexLibrary 428-429
- JIT
  - Java 235
  - .NET 404
- JRE 234
- jregex comparative description 374
  
- \p{Katakana} 120, 122
- keeping in sync 210-211
- Keisler, H. J. 85
- Kleene, Stephen 85
- The Kleene Symposium* 85
- \kname (see named capture)
- Korean text processing 29
- Kunen, K. 85
  
- \p{L&} 120-121, 123
  - in Perl 288
- \p{L} 119-120, 131, 380, 390
- £ 122
- \l 290
- language (also see: .NET; C#; Java; MySQL; Perl; procmail; Python; Ruby; Tcl; VB.NET)
  - character class 10, 13
  - identifiers 24
- \p{Latin} 120
- Latin-1 encoding 29, 87, 105, 107, 121
- lazy 166-167
  - (also see greedy)
  - essence 159, 168-169
  - favors match 167-168
  - vs. greedy 169, 256-257
  - in Java 373
  - optimization 249, 256
  - quantifier 140
- lazy evaluation 181, 355
- \L..\E 290
  - inhibiting 292

- lc 290
- lcfirst 290
- leftmost match** 177-179
- Length
  - Group object method 424
  - Match object method 423
- length() ORO 396
- length-cognizance optimization** 245, 247
- \p{Letter} 120, 288
- \p{Letter\_Number} 121
- \$LevelN 330, 343
- lex** 86
  - \$ 111
  - dot 110
  - history 87
  - and trailing context 182
- lexer** building 130, 315
- lexical scope** 299
- LF 109, 382
- Li, Yadong xxii
- LIFO backtracking** 159
- limit**
  - backtracking 237
  - recursion 249-250
- line** (also see string)
  - anchor optimization 246
  - vs. string 55
- line anchor** 111-112
  - mechanics of matching 150
  - variety of implementations 87
- line feed** 109
- LINE SEPARATOR 109, 121, 382
- line terminators** 108-109, 111, 128, 382
  - with \$ and ^ 111
- \p{Line\_Separator} 121
- link**
  - matching 201
    - (also see URL examples)
    - Java 204, 209
- list context** 294, 310-311
  - forcing 310
- literal string** initial string discrimination
  - 244-246, 249, 251-252, 257-259, 332, 361
- literal text**
  - introduced 5
  - exposing 255
  - mechanics of matching 149
  - pre-check optimization 244-246, 249, 251-252, 257-259, 332, 361
- literal-text mode** 112, 134-135, 290
  - inhibiting 292
- \p{Ll} 121, 400
- \p{Lm} 121, 400
- \p{Lo} 121, 400
- local 296, 341
  - in embedded code 336
  - vs. my 297
- locale** 126
  - overview 87
  - \w 119
- localizing** 296-297
- localtime 294, 319, 351
- lock up** (see neverending match)
- locking in regex literal** 352
- "A logical calculus of the ideas imminent in nervous activity" 85
- longest match** finding 334-335
- longest-leftmost match** 148, 177-179
- lookahead** 132
  - (also see lookaround)
  - introduced 60
  - auto 403
  - example 61-64
  - in Java 373
  - mimic atomic grouping 174
  - mimic optimizations 258-259
  - negated
    - <B>...</B> 167
  - positive vs. negative 66
- lookaround**
  - introduced 59
  - backtracking 173-174
  - in conditional 139
  - and DFAs 182
  - doesn't consume text 60
  - mimicking class set operations 124
  - mimicking word boundaries 132
  - in Perl 288
- lookbehind** 132
  - (also see lookaround)
  - in Java 373
  - in .NET 402
  - in Perl 288
  - positive vs. negative 66
  - unlimited 402
- lookingAt() 385
- loose matching** (see case-insensitive mode)
- Lord, Tom 182
- \p{Lowercase\_Letter} 121
- LS 109, 121, 382
- \p{Lt} 121, 400
- \p{Lu} 121, 400
- Lunde, Ken xxii, 29

- `\p{M}` 120, 125
- `(?m)` (see: enhanced line-anchor mode; mode modifier)
- `/m` 134
  - (also see: enhanced line-anchor mode; mode modifier)
- `m/.../` introduced 38
- machine-dependent character codes 114
- MacOS 114
- mail processing example 53-59
- makudonarudo example 165, 169, 228-232, 264
- `\p{Mark}` 120
- match** 306-318
  - (also see: DFA; NFA)
  - actions 95
  - context 294-295, 309
    - list 294, 310-311
    - scalar 294, 310, 312-316
  - DFA vs. NFA 224
  - efficiency 179
  - example with backtracking 160
  - example without backtracking 160
  - lazy example 161
  - leftmost-longest 335
  - longest 334-335
  - `m/.../`
    - introduced 38
  - mechanics (also see: greedy; lazy)
    - `.*` 152
    - greedy introduced 151
    - anchors 150
    - capturing parentheses 149
    - character classes and dot 149
    - consequences 156
    - literal text 149
  - modes 109-112
    - `java.util.regex` 380
  - negating 309
  - neverending 222-228, 330, 340
    - avoiding 264-265
    - discovery 226-228
    - explanation 226-228
    - non-determinism 264
    - short-circuiting 250
    - solving with atomic grouping 268
    - solving with possessive quantifiers 268
  - NFA vs. DFA 156-157, 180-182
  - position (see `pos`)
  - POSIX
    - in Perl 335
  - shortest-leftmost 183
- match** (cont'd)
  - side effects 317
    - intertwined 43
    - Perl 40
  - speed 181
  - in a string 27
  - tag-team 130
  - viewing mechanics 331-332
- `Match` `Empty` 426
- `match()` 393
- `Match(.NET)` `Success` 96
- Match object (.NET)** 411
  - `Capture` 431
  - creating 415, 423
  - `Groups` 423
  - `Index` 423
  - `Length` 423
  - `NextMatch` 423
  - `Result` 423
  - `Success` 421
  - `Synchronized` 424
  - `ToString` 422
  - using 421
  - `Value` 422
- `Match` (Regex object method) 415
- "match rejected by optimizer" 363
- match result object model** 371
- match state object model** 370
- `MatchCollection` 416
- `matcher()` (Pattern method) 384
- Matcher object** 384
  - reusing 387
- matches**
  - unexpected 194-195
  - viewing all 332
- `matches()` (Pattern method) 384, 390
- `Matches` (Regex object method) 416
- `MatchEvaluator` 417-418
- matching**
  - delimited text 196-198
  - HTML tag 200
  - longest-leftmost 177-179
- `MatchObject` object (.NET) creating 416
- `\p{Math_Symbol}` 121
- Maton, William xxii, 36
- MBOL 362
- `\p{Mc}` 121
- McCloskey, Mike xxii
- McCulloch, Warren 85
- `\p{Me}` 121
- mechanics** viewing 331-332



- metacharacter**
  - introduced 5
  - conflicting 44-46
  - differing contexts 10
  - first-class 87, 92
  - vs. metasequence 27
- metasequence** defined 27
- mimic**
  - \$` 357
  - \$' 357
  - \$& 302, 357
  - atomic grouping 174
  - class set operations 124
  - conditional with lookahead 139
  - initial-character discrimination optimization 258-259
  - named capture 344-345
  - POSIX matching 335
  - possessive quantifiers 343-344
  - variable interpolation 321
  - word boundaries 66, 132, 341-342
- minlen** *length* 362
- minus in character class** 9
- MISL** .NET 404
- \p{Mn}** 121
- mode modifier** 109, 133-135
- mode-modified span** 109, 134
- modes** introduced with *egrep* 14-15
- \p{Modifier\_Letter}** 121
- modifiers** (also see *match*, *modes*)
  - combining 69
  - example with five 316
  - /g 51
  - /i 47
  - “locking in” 304-305
  - notation 98
  - /osmosis 293
  - Perl core 292-293
  - with regex object 304-305
- \p{Modifier\_Symbol}** 121
- Mre=debug** (see *use re 'debug'*)
- Mui, Linda** xxii
- multi-character quotes** 165-166
- Multiline (.NET)** 402, 413-414, 421
- MULTILINE (Pattern flag)** 81, 380, 382
- multiple-byte character encoding** 29
- MungeRegexLiteral** 342-344, 346
- my**
  - binding 339
  - in embedded code 338-339
  - vs. *local* 297
- MySQL**
  - after-match data 136
  - DBIx::DWIW 258
  - version covered 91
  - word boundaries 132
- \n** 49, 114-115
  - introduced 44
  - machine-dependency 114
- \p{N}** 120, 390
- (?n)** 402
- \$^N** 300-301, 344-346
- named capture** 137
  - mimicking 344-345
  - .NET 402
  - with unnamed capture 403
- naughty variables** 356
  - okay for debugging 331
- \p{Nd}** 121, 380, 400
- negated class**
  - introduced 10-11
  - and lazy quantifiers 167
  - Tcl 111
- negative lookahead** (see *lookahead*, *negative*)
- negative lookbehind** (see *lookbehind*, *negative*)
- NEL** 109, 382, 400
- nervous system** 85
- nested constructs**
  - .NET 430
  - Perl 328-331, 340-341
- \$NestedStuffRegex** 339, 346
- .NET** 399-432
  - \$+ 202
  - flavor overview 91
  - after-match data 136
  - benchmarking 236
  - JIT 404
  - line anchors 128
  - literal-text mode 135
  - MISL 404
  - object model 411
  - regex approach 96-97
  - regex flavor 401
  - search-and-replace 408, 417-418
  - URL parsing example 204
  - version covered 91
  - word boundaries 132
- neurophysiologists** early regex study 85

- neverending match** 222-228, 330, 340
  - avoiding 264-265
  - discovery 226-228
  - explanation 226-228
  - non-determinism 264
  - short-circuiting 250
  - solving with atomic grouping 268
  - solving with possessive quantifiers 268
- New Regex** 96, 99, 410, 415
- newline** and HTTP 115
- NEXT LINE** 109, 382, 400
- NextMatch (Match object method)** 423
- NFA**
  - first introduced 145
  - introduction 153
  - acronym spelled out 156
  - and alternation 174-175
  - compared with DFA 156-157, 180-182
  - control benefits 155
  - efficiency 179
  - essence (see backtracking)
  - freeflowing regex 277-281
  - and greediness 162
  - implementation ease 182
  - nondeterminism 265
    - checkpoint 264
  - POSIX efficiency 179
  - testing for 146-147
  - theory 180
- Nicholas, Ethan** xxii
- \p{N1}** 121
- \N{LATIN SMALL LETTER SHARP S}** 290
- \N{name}** 290
  - (also see pragma)
  - inhibiting 292
- \p{No}** 121
- no re 'debug'** 361
- no\_match\_vars** 357
- nomenclature** 27
- non-capturing parentheses** 45, 136-137, 373
  - (also see parentheses)
- Nondeterministic Finite Automaton** (see NFA)
- None (.NET)** 415, 421
- non-greedy** (see lazy)
- nonillion** 226
- nonregular sets** 180
- \p{Non\_Spacing\_Mark}** 121
- non-word boundaries** (see word boundaries)
- "normal"** 262-266
- null** 116
  - with dot 118
- NullPointerException** 396
- \p{Number}** 120
- /o** 352-353
  - with regex object 354
- Obfuscated Perl Contest** 320
- object model**
  - Java 368-372
  - .NET 410-411
- Object Oriented Perl** 339
- object-oriented handling** 95-97
  - compile caching 244
- octal escape** 115, 117
  - vs. backreference 406-407
  - in Java 373
  - in Perl 286
- on-demand recompilation** 351
- oneseif example** 332, 334
- \p{Open\_Punctuation}** 121
- operators** Perl list 285
- optimization** 239-252
  - (also see: atomic grouping; possessive quantifiers; efficiency)
  - automatic possessification 251
  - BLTN 235-236, 375
  - with bump-along 255
  - end-of-string anchor 246
  - excessive backtrack 249-250
  - hand tweaking 252-261
  - implicit line anchor 191
  - initial character discrimination 244-246, 249, 251-252, 257-259, 332, 361
  - JIT 235, 404
  - lazy evaluation 181
  - lazy quantifier 249, 256
  - leading [.\*] 246
  - literal-string concatenation 247
  - need cognizance 252
  - needless class elimination 249
  - needless parentheses 248
  - pre-check of required character 244-246, 249, 251-252, 257-259, 332, 361
  - simple repetition
    - discussed 247-248
  - small quantifier equivalence 251-252
  - state suppression 250-251
  - string/line anchors 149, 181
  - super-linear short-circuiting 250
- Option (.NET)** 409

- optional (also see quantifier)
  - whitespace 18
- Options (Regex object method) 421
- OR class set operations 123-124
- Oram, Andy xxii, 5
- ordered alternation 175-177
  - (also see alternation, ordered)
  - pitfalls 176
- org.apache.oro.text.regex 392-398
  - benchmark results 376
  - comparative description 374
- org.apache.regexp
  - comparative description 375
  - speed 376
- org.apache.xerces.utils.regex 372
- ORO 392-398
  - benchmark results 376
  - comparative description 374
- osmosis 293
- /osmosis 293
- \p{Other} 120
- \p{Other\_Letter} 121
- \p{Other\_Number} 121
- \p{Other\_Punctuation} 121
- \p{Other\_Symbol} 121
- our 295, 336
- overload pragma 342
  
- \p{ } 119
- \p{P} 120
- \p{^ } 288
- \p{All} 123
  - in Perl 288
- \p{all} 380
- panic: top\_env 332
- \p{Any} 123
  - in Perl 288
- Papen, Jeffrey xxii
- PARAGRAPH SEPARATOR 109, 121, 382
- \p{Paragraph\_Separator} 121
- parentheses
  - as \(-\ ) 86
  - and alternation 13
  - balanced 328-331, 340-341, 430
    - difficulty 193-194
  - capturing 135-136, 300
    - introduced with *egrep* 20-22
    - and DFAs 150, 182
    - mechanics 149
    - in Perl 41
  - capturing only 152
  - counting 21
- parentheses (cont'd)
  - elimination optimization 248
  - grouping-only (see non-capturing parentheses)
  - limiting scope 18
  - named capture 137, 344-345, 402-403
  - nested 328-331, 340-341, 430
  - non-capturing 45, 136-137
    - in Java 373
  - non-participating 300
  - with split
    - Java ORO 395
    - .NET 403, 420
    - Perl 326
- \p{Arrows} 122
- parsing regex 404
- participate in match 139
- Pascal 36, 59, 182
  - matching comments of 265
- \p{Assigned} 123-124
  - in Perl 288
- Pat (Java Package)
  - comparative description 374
  - speed 377
- patch* 88
- path (see backtracking)
- pathname example 190-192
- Pattern
  - CANON\_EQ 108, 380
  - CASE\_INSENSITIVE 95, 109, 380, 383
  - COMMENTS 99, 218, 378, 380, 386
  - compile() 383
  - DOTALL 380, 382
  - matcher() 384
  - matches() 384, 390
  - MULTILINE 81, 380, 382
  - UNICODE\_CASE 380, 383
  - UNIX\_LINES 380, 382
- PatternSyntaxException 381, 383
- \p{Basic\_Latin} 122
- \p{Box\_Drawing} 122
- \p{Pc} 121, 400
- \p{C} 120
- \p{Cc} 121
- \p{Cf} 121
- \p{Cherokee} 120
- \p{Close\_Punctuation} 121
- \p{Cn} 121, 123-124, 380, 401
- \p{Co} 121
- \p{Connector\_Punctuation} 121
- \p{Control} 121

**PCRE**

lookbehind 132  
 version covered 91  
 \p{Currency} 122  
 \p{Currency\_Symbol} 121  
 \p{Cyrillic} 120, 122  
 \p{Pd} 121  
 \p{Dash\_Punctuation} 121  
 \p{Decimal\_Digit\_Number} 121  
 \p{Dingbats} 122  
 \p{Pe} 121  
 PeakWebhosting.com xxii  
 \p{Enclosing\_Mark} 121

**people**

Aho, Alfred 86, 180  
 Balling, Derek xxii  
 Barwise, J. 85  
 Bennett, Mike xxi  
 Clemens, Sam 375  
 Click, Cliff xxii  
 Constable, Robert 85  
 Conway, Damian 339  
 Cruise, Tom 51  
 Flanagan, David xxii  
 Friedl, Alfred 176  
 Friedl, brothers 33  
 Friedl, Fumie xxi  
   birthday 11-12  
 Friedl, Liz 33  
 Friedl, Stephen xxii  
 George, Kit xxii  
 Goldberger, Ray xxii  
 Gosling, James 89  
 Gutierrez, David xxii  
 Hietaniemi, Jarkko xxii  
 Keisler, H. J. 85  
 Kleene, Stephen 85  
 Kunen, K. 85  
 Li, Yadong xxii  
 Lord, Tom 182  
 Lunde, Ken xxii, 29  
 Maton, William xxii, 36  
 McCloskey, Mike xxii  
 McCulloch, Warren 85  
 Mui, Linda xxii  
 Nicholas, Ethan xxii  
 Oram, Andy xxii, 5  
 Papen, Jeffrey xxii  
 Perl Porters 90  
 Pinyan, Jeff 246  
 Pitts, Walter 85  
 Purcell, Shawn xxii  
 Reed, Jessamyn xxii

**people (cont'd)**

Reinhold, Mark xxii  
 Rudkin, Kristine xxii  
 Savarese, Daniel xxii  
 Sethi, Ravi 180  
 Spencer, Henry 88, 182-183, 243  
 Thompson, Ken 85-86, 110  
 Trapszo, Kasia xxii  
 Tubby 264  
 Ullman, Jeffrey 180  
 Wall, Larry 88-90, 138, 363-364  
 Wilson, Dean xxii  
 Woodward, Josh xxii  
 Zawodny, Jeremy xxii, 258

**Perl**

\$/ 35  
 flavor overview 91, 287  
 introduction 37-38  
 context (also see match, context)  
   contorting 294  
 efficiency 347-363  
 greatest weakness 286  
 history 88-90, 308  
 in Java 375, 392  
 line anchors 128  
 modifiers 292-293  
 motto 348  
 option  
   -o 36  
   -c 361  
   -Dr 363  
   -e 36, 53, 361  
   -i 53  
   -M 361  
   -Mre=debug 363  
   -n 36  
   -p 53  
   -w 38, 296, 326, 361  
 regex operators 285  
 Σ 110  
 version covered 91  
 warnings 38  
   (\$^w variable) 297  
   use warnings 326, 363

**Perl Porters** 90

Perl5Util 392, 396  
 perladmin 299  
 \p{Pf} 121, 400  
 \p{Final\_Punctuation} 121  
 \p{Format} 121  
 \p{Gujarati} 120  
 \p{Han} 120  
 \p{Hangul\_Jamo} 122

- \p{Hebrew} 120, 122
- \p{Hiragana} 120
- PHP
  - after-match data 136
  - line anchors 128
  - lookbehind 132
  - mode modifiers 133
  - strings 103
  - version covered 91
  - word boundaries 132
- \p{Pi} 121, 400
- \p{InArrows} 122
- \p{InBasic\_Latin} 122
- \p{InBox\_Drawing} 122
- \p{InCurrency} 122
- \p{InCyrillic} 122
- \p{InDingbats} 122
- \p{InHangul\_Jamo} 122
- \p{InHebrew} 122
- \p{Inherited} 122
- \p{Initial\_Punctuation} 121
- \p{InKatakana} 122
- \p{InTamil} 122
- \p{InTibetan} 122
- Pinyan, Jeff 246
- \p{IsCherokee} 120
- \p{IsCommon} 122
- \p{IsCyrillic} 120
- \p{IsGujarati} 120
- \p{IsHan} 120
- \p{IsHebrew} 120
- \p{IsHiragana} 120
- \p{IsKatakana} 120
- \p{IsLatin} 120
- \p{IsThai} 120
- \p{IsTibetan} 122
- Pitts, Walter 85
- \p{Katakana} 120, 122
- \p{L} 119-120, 131, 380, 390
- \p{L&} 120-121, 123
  - in Perl 288
- \p{Latin} 120
  - (?P<*name*>...) (see named capture)
- \p{Letter} 120, 288
- \p{Letter\_Number} 121
- \p{Line\_Separator} 121
- \p{Ll} 121, 400
- \p{Lm} 121, 400
- \p{Lo} 121, 400
- \p{Lowercase\_Letter} 121
- \p{Lt} 121, 400
- \p{Lu} 121, 400
- plus
  - as \+ 139
  - introduced 18-20
  - backtracking 162
  - greedy 139
  - lazy 140
  - possessive 140
- \p{M} 120, 125
- \p{Mark} 120
- \p{Math\_Symbol} 121
- \p{Mc} 121
- \p{Me} 121
- \p{Mn} 121
- \p{Modifier\_Letter} 121
- \p{Modifier\_Symbol} 121
- \p{N} 120, 390
  - (?P=*name*...) (see named capture)
- \p{Nd} 121, 380, 400
- \p{Nl} 121
- \p{No} 121
- \p{Non\_Spacing\_Mark} 121
- \p{Number} 120
- \p{Po} 121
- \p{Open\_Punctuation} 121
- population example 59
- pos 128-131, 313-314, 316
  - (also see \G)
- positive lookahead (see lookahead, positive)
- positive lookbehind (see lookbehind, positive)
- POSIX
  - [:::] 125
  - [...] 126
  - Basic Regular Expressions 87-88
  - bracket expressions 125
  - character class 125
  - character class and locale 126
  - character equivalent 126
  - collating sequences 126
  - dot 118
  - empty alternatives 138
  - Extended Regular Expressions 87-88
  - superficial flavor chart 88
  - in Java 374
  - locale 126
    - overview 87
  - longest-leftmost rule 177-179, 335
- POSIX NFA
  - backtracking example 229
  - testing for 146-147

- possessive quantifiers 140, 172-173
  - (also see atomic grouping)
  - automatic 251
  - for efficiency 259, 268-270
  - example 198, 201
  - mimicking 343-344
  - optimization 250-251
- postal code example 208-212
- postMatch() 397
- \p{Other} 120
- \p{Other\_Letter} 121
- \p{Other\_Number} 121
- \p{Other\_Punctuation} 121
- \p{Other\_Symbol} 121
- & 122
- \p{P} 120
- \p{Paragraph\_Separator} 121
- \p{Pc} 121, 400
- \p{Pd} 121
- \p{Pe} 121
- \p{Pf} 121, 400
- \p{Pi} 121, 400
- \p{Po} 121
- \p{Private\_Use} 121
- \p{Ps} 121
- \p{Punctuation} 120
- pragma**
  - charnames 290
    - (also see \N{name})
  - overload 342
  - re 361, 363
  - strict 295, 336, 345
  - warnings 326, 363
- pre-check of required character 244-246, 249, 251-252, 257-259, 361
  - mimic 258-259
  - viewing 332
- preMatch() 397
- pre-match copy 355
- prepending filename to line 79
- price rounding example 51-52, 167-168
  - with alternation 175
  - with atomic grouping 170
  - with possessive quantifier 169
- Principles of Compiler Design* 180
- printf 40
- private vs. global Perl variables 295
- \p{Private\_Use} 121
- procedural handling 95-97
  - compile caching 243
- procmail 94
  - version covered 91
- Programming Perl* 283, 286, 339
- promote 294-295
- properties 119-121, 123-124, 288, 380
  - \p{S} 120
- PS 109, 121, 382
  - \p{Ps} 121
  - \p{Sc} 121-122
  - \p{Separator} 120
  - \p{Sk} 121
  - \p{Sm} 121
  - \p{So} 121
  - \p{Space\_Separator} 121
  - \p{Spacing\_Combining\_Mark} 121
  - \p{Symbol} 120
  - \p{Tamil} 122
  - \p{Thai} 120
  - \p{Tibetan} 122
  - \p{Titlecase\_Letter} 121
- publication**
  - Bulletin of Math. Biophysics* 85
  - Communications of the ACM* 85
  - Compilers—Principles, Techniques, and Tools* 180
  - Embodiments of Mind* 85
  - The Kleene Symposium* 85
  - “A logical calculus of the ideas imminent in nervous activity” 85
  - Object Oriented Perl* 339
  - Principles of Compiler Design* 180
  - Programming Perl* 283, 286, 339
  - Regular Expression Search Algorithm* 85
  - “The Role of Finite Automata in the Development of Modern Computing Theory” 85
- \p{Unassigned} 121, 123
  - in Perl 288
- \p{Punctuation} 120
- \p{Uppercase\_Letter} 121
- Purcell, Shawn** xxii
- Python**
  - after-match data 136
  - benchmarking 237
  - line anchors 128
  - mode modifiers 133
  - regex approach 97
  - strings 103-104
  - version covered 91
  - word boundaries 132
  - \z 111
- \p{Z} 119-120, 380, 400
  - \p{Zl} 121
  - \p{Zp} 121
  - \p{Zs} 121

- Qantas** 11
- `\Q-\E` 290
  - inhibiting 292
  - in Java 373
- qed** 85
- `qr/./` (also see regex objects)
  - introduced 76
- quantifier** (also see: plus; star; question mark; interval; lazy; greedy; possessive quantifiers)
  - and backtracking 162
  - factor out 255
  - grouping for 18
  - multiple levels 265
  - optimization 247, 249
  - and parentheses 18
  - possessive quantifiers 140, 172-173
    - for efficiency 259, 268-270
    - mimicking
    - optimization
      - automatic
  - question mark
    - as `\?` 139
    - introduced 17-18
    - backtracking 160
    - greedy 139
    - lazy 140
    - possessive 140
    - smallest preceding subexpression 29
- question mark**
  - as `\?` 139
  - backtracking 160
  - greedy 139
  - lazy 140
  - possessive 140
- quoted string** (see double-quoted string example)
- quotes** multi-character 165-166
  
- `r"..."` 103
- `$$R` 302, 327
- `\r` 49, 114-115
  - machine-dependency 114
- `re` 361
- `re 'debug'` 363
- `re pragma` 361, 363
- reality check** 226-228
- recursive matching** (see dynamic regex)
- red dragon** 180
- Reed, Jessamyn** xxii
- Reflection** 429
  
- regex**
  - balancing needs 186
  - compile 179-180, 350
  - default 308
  - delimiters 291-292
  - DFA (see DFA)
  - encapsulation (see regex objects)
  - engine analogy 143-147
  - vs. English 275
  - frame of mind 6
  - freeflowing design 277-281
  - history 85-91
  - library 76, 207
  - longest-leftmost match 177-179
    - shortest-leftmost 183
  - mechanics 241-242
  - NFA (see NFA)
  - nomenclature 27
  - operands 288-292
  - overloading 291, 328
    - inhibiting 292
    - problems 344
  - subexpression
    - defined 29
- regex literal** 288-292, 307
  - inhibiting processing 292
  - locking in 352
  - parsing of 292
  - processing 350
  - regex objects 354
- Regex (.NET)**
  - `CompileToAssembly` 427, 429
  - creating
    - options 413-415
  - `Escape` 427
  - `GetGroupNames` 421-422
  - `GetGroupNumbers` 421-422
  - `GroupNameFromNumber` 421-422
  - `GroupNumberFromName` 421-422
  - `IsMatch` 407, 415, 425
  - `Match` 96, 408, 410, 415, 425
  - `Matches` 416, 425
  - object
    - creating 96, 410, 413-415
    - exceptions 413
    - using 96, 415
  - `Options` 421
  - `Replace` 408-409, 417-418, 425
  - `RightToLeft` 421
  - `Split` 419-420, 425
  - `ToString` 421
  - `Unescape` 427

- regex objects 303-306
  - (also see `qr/.../`)
  - efficiency 353-354
  - `/g` 354
  - match modes 304-305
  - `/o` 354
  - in regex literal 354
  - viewing 305-306
- regex overloading 292
  - (also see `use overload`)
  - example 341-345
- `http://regex.info/` xxii, 7, 345, 358
- `RegexCompilationInfo` 429
- regex-directed matching 153
  - (also see NFA)
  - and backreferences 303
  - and greediness 162
- `Regex.Escape` 135
- `RegexOptions`
  - `Compiled` 236, 402, 404, 414, 421-422, 429
  - `ECMAScript` 400, 402, 406-407, 415, 421
  - `ExplicitCapture` 402, 414, 421
  - `IgnoreCase` 96, 99, 402, 413, 421
  - `IgnorePatternWhitespace` 99, 402, 413, 421
  - `Multiline` 402, 413-414, 421
  - `None` 415, 421
  - `RightToLeft` 402, 405-406, 414, 420-421, 423-424
  - `Singleline` 402, 414, 421
- `Regexp` (Java package)
  - comparative description 375
  - speed 376
- `regsub` 100
- regular expression origin of term 85
- Regular Expression Search Algorithm* 85
- regular sets 85
- Reinhold, Mark xxii
- removing whitespace 199-200
- `Replace` (Regex object method) 417-418
- `replaceAll` 387
- `replaceFirst()` 387-388
- reproductive organs 5
- required character pre-check 244-246, 249, 251-252, 257-259, 332, 361
- `re-search-forward` 100
- `reset()` 387
- `Result` (Match object method) 423
- `RightToLeft` (Regex property) 421-422
- `RightToLeft` (.NET) 402, 405-406, 414, 420-421, 423-424
- “The Role of Finite Automata in the Development of Modern Computing Theory” 85
- Ruby
  - `$` and `^` 111
  - after-match data 136
  - benchmarking 238
  - `\G` 131
  - line anchors 128
  - mode modifiers 133
  - version covered 91
  - word boundaries 132
- Rudkin, Kristine xxii
- rule
  - earliest match wins 148-149
  - standard quantifiers are greedy 151-153
- `rx` 182
- `s/.../.../` 50, 318-321
- `\s` 49, 56, 119
- `\p{S}` 120
- `\s` 49, 119
  - introduction 47
  - in Emacs 127
  - in Perl 288
- `(?s)` (see: dot-matches-all mode; mode modifier)
- `/s` 134
  - (also see: dot-matches-all mode; mode modifier)
- Savarese, Daniel xxii
- saved states (see backtracking, saved states)
- `SawAmpersand` 358
- say what you mean 195, 274
- SBOL 362
- `\p{Sc}` 121-122
- scalar context 294, 310, 312-316
  - forcing 310
- `schaffkopf` 33
- scope lexical vs. dynamic 299
- scripts 120-122, 288
- search-and-replace
  - awk 99
  - Java 387, 394
  - .NET 408, 417-418
  - Tcl 100
- sed
  - after-match data 136
  - dot 110
  - history 87
  - version covered 91



- sed (cont'd)
  - word boundaries 132
  - 正規表現は簡単だよ! 5
  - \p{Separator} 120
- server VM 234, 236, 375
- set operations (see class, set operations)
- Sethi, Ravi 180
- shell 7
- Σ 110
- simple quantifier optimization 247-248
- single quotes delimiter 292, 319
- Singleline (.NET) 402, 414, 421
- \p{Sk} 121
- \p{Sm} 121
- small quantifier equivalence 251-252
- \p{So} 121
- \p{Space\_Separator} 121
- \p{Spacing\_Combining\_Mark} 121
- span (see: mode-modified span; literal-text mode)
- “special” 262-266
- Spencer, Henry 88, 182-183, 243
- split() java.util.regex 390
- split ORO 394-396
- split
  - with capturing parentheses
    - Java ORO 395
    - .NET 403, 420
    - Perl 326
  - chunk limit
    - Java ORO 395
    - java.util.regex 391
    - Perl 323
  - into characters 322
  - in Perl 321-326
  - trailing empty items 324
  - whitespace 325
- Split (Regex object method) 419-420
- ß 110, 126, 290, 366
- standard formula for matching delimited text 196
- star
  - introduced 18-20
  - backtracking 162
  - greedy 139
  - lazy 140
  - possessive 140
- start() 385
- start of match (see \G)
- start of word (see word boundaries)
- start-of-line/string (see anchor, caret)
- start-of-string anchor optimization 245-246, 255-256, 315
- states (also see backtracking, saved states)
  - flushing (see: atomic grouping; look-around; possessive quantifiers)
- stdClass 'list' 362
- stock pricing example 51-52, 167-168
  - with alternation 175
  - with atomic grouping 170
  - with possessive quantifier 169
- Strict (Option) 409
- strict pragma 295, 336, 345
- String
  - matches() 384
  - replaceAll 387
  - replaceFirst() 388
  - split() 390
- string (also see line)
  - double-quoted (see double-quoted string example)
  - initial string discrimination 244-246, 249, 251-252, 257-259, 332, 361
  - vs. line 55
  - match position (see pos)
  - pos (see pos)
- StringBuffer 388
- strings
  - C# 102
  - Emacs 100
  - Java 102
  - PHP 103
  - Python 103-104
  - as regex 101-105, 305
  - Tcl 104
  - VB.NET 102
- stripping whitespace 199-200
- study 359-360
  - when not to use 359
- subexpression defined 29
- substitute() 394
- substitution
  - delimiter 319
  - s/--/--/ 50, 318-321
- substring initial substring discrimination 244-246, 249, 251-252, 257-259, 332, 361
- subtraction class set operations 124
- Success
  - Group object method 424
  - Match object method 421
- Sun's regex package (see java.util.regex)
- super-linear (see neverending match)
- super-linear short-circuiting 250
- \p{Symbol} 120

- Synchronized Match object method 424
- syntax class Emacs 127
- System.currentTimeMillis() 236
- System.Reflection 429
- System.Text.RegularExpressions 407, 409
- 
- \t 49, 114-115
  - introduced 44
- tag matching 200-201
- tag-team matching 130, 315
- \p{Tamil} 122
- Tcl
  - [<::] 92
  - flavor overview 91
  - benchmarking 239
  - dot 111-112
  - hand-tweaking 243, 259
  - line anchors 112, 128
  - mode modifiers 133
  - regex implementation 182
  - regsub 100
  - search-and-replace 100
  - strings 104
  - version covered 91
  - word boundaries 132
- temperature conversion example
  - in .NET 419
  - in Java 389
  - in Perl 37
  - Perl one-liner 283
- terminators (see line terminators)
- testing engine type 146-147
- text-directed matching 153
  - (also see DFA)
  - regex appearance 162
- text-to-HTML example 67-77
- \p{Thai} 120
- then (see conditional)
- theory of an NFA 180
- There's more than one way to do it* 349
- this|that example 132, 138, 243, 245-246, 252, 255, 260-261
- Thompson, Ken 85-86, 110
- thread scheduling Java benchmarking 236
- \p{Tibetan} 122
- tied variables 299
- time() 232
- time of day 26
- Time::HiRes 232, 358, 360
- Time.new 238
- Timer() 237
- 
- title case 109
- \p{Titlecase\_Letter} 121
- TiVo 3
- tokenizer building 130, 315
- toothpicks scattered 100
- tortilla 126
- ToString
  - Group object method 424
  - Match object method 422
  - Regex object method 421
- toString ORO 396
- Traditional NFA testing for 146-147
- trailing context 182
- transmission (also see \G)
  - optimizations 245-247
- Trapszo, Kasia xxii
- Tubby 264
- typographical conventions xix
- 
- \u 116, 290, 400
- \U 116
- \U-\AE 290
  - inhibiting 292
- uc 290
- U+00B5 106
- ucfirst 290
- UCS-2 encoding 106
- UCS-4 encoding 106
- Ullman, Jeffrey 180
- \p{Unassigned} 121, 123
  - in Perl 288
- unconditional caching 350
- underscore in \w history 89
- Unescape 427
- Unicode
  - overview 106-108
  - block 122
    - Java 380
    - .NET 400
    - Perl 288
  - categories (see Unicode, properties)
  - character
    - combining 107, 122, 125, 288
  - code point
    - introduced 106
    - beyond U+FFFF 108
    - multiple 107
    - unassigned in block 122
  - combining character 107, 122, 125, 288
  - in Java 380
  - line terminators 108-109, 111
    - in Java 382

- Unicode (cont'd)
  - loose matching (see case-insensitive mode)
  - in .NET 401
  - properties 119, 288
    - java.util.regex 380
    - list 120-121
    - \p{All} 123, 288
    - \p{Any} 123, 288
    - \p{Assigned} 123-124, 288
    - \p{Unassigned} 121, 123, 288
  - script 120-122, 288
  - support in Java 373
  - Version 3.1 108, 380, 401
  - Version 3.2 288
  - whitespace and /x 288
- UNICODE\_CASE (Pattern flag) 380, 383
- UnicodeData.txt* 290
- unicore* 290
- UNIX\_LINES (Pattern flag) 380, 382
- unmatch 152, 161, 163
  - .\* 165
  - atomic grouping 171
- unrolling the loop 261-276
  - example 270-271
  - general pattern 264
- \p{Uppercase\_Letter} 121
- URL encoding 320
- URL example 74-77, 201-204, 208, 260, 303-304, 306, 320
  - egrep 25
  - Java 209
  - plucking 205-208
- use charnames 290
- use Config 290, 299
- use English 357
- use overload 342
  - (also see regex overloading)
- use re 361, 363
- use re 'debug' 361, 363
- use re 'eval' 337
- use strict 295, 336, 345
- use Time::HiRes 358, 360
- use warnings 326, 363
- username example 73, 76, 98
  - plucking from text 71-73
  - in a URL 74-77
- using System.Text.RegularExpressions 410
- UTF-16 encoding 106
- UTF-8 encoding 106
- \v 364
- \v 114-115, 364
- Value
  - Group object method 424
  - Match object method 422
- variable names example 24
- variables
  - after match
    - pre-match copy 355
  - binding 339
  - fully qualified 295
  - interpolation 344
  - naughty 356
  - tied 299
- VB.NET (also see .NET)
  - comments 99
  - regex approach 96-97
  - strings 102
  - URL parsing example 204
- verbatim strings 102
- Version 7 regex 182
- Version 8 regex 182
- versions covered in this book 91
- vertical tab 109
  - in Perl \s 288
- vi after-match data 136
- Vietnamese text processing 29
- virtual machine 234-236, 375
- Visual Basic (see VB.NET)
- Visual Studio .NET 428
- VM 234, 236, 375
  - warming up 235
- void context 294
- VT 109
- \w 49, 119
- W 297
- \w 49, 65, 119
  - in Emacs 127
  - many different interpretations 93
  - in Perl 288
- Wall, Larry 88-90, 138, 363-364
- warming up Java VM 235
- warnings 296
  - temporarily turning off 297
- warnings pragma 326, 363
- while vs. foreach vs. if 320
- whitespace
  - allowing optional 18
  - removing 199-200
- wildcards filename 4
- Wilson, Dean xxii

Woodward, Josh xxii  
word anchor mechanics of matching 150  
word boundaries 131  
    \`<...>`  
        *egrep* 15  
        introduced 15  
        in Java 373  
        many programs 132  
        mimicking 66, 132, 341-342  
        in Perl 132, 288  
www.cpan.org 358  
www.PeakWebhosting.com xxii  
www.regex.info 358

`\x` 107, 125  
`\x` 116, 400  
    in Perl 286  
(?`x`) (see: comments and free-spacing  
    mode; mode modifier)  
`/x` 134, 288  
    (also see: comments and free-spacing  
    mode; mode modifier)  
    introduced 72  
    history 90

Xerces  
    org.apache.xerces.utils.regex  
    372

`-y` old *grep* 86  
¥ 122  
Yahoo! xxi, 74, 130, 190, 205, 207, 258,  
    314

`\z` 111, 127-128, 316  
    (also see enhanced line-anchor mode)  
    in Java 373  
    optimization 246  
`\Z` 111, 127-128  
    (also see enhanced line-anchor mode)  
    in Java 373  
    optimization 246  
`\p{Z}` 119-120, 380, 400  
Zawodny, Jeremy xxii, 258  
zero-width assertions (see: anchor; look-  
    ahead; lookbehind)  
ZIP code example 208-212  
`\p{Zl}` 121  
`\p{Zp}` 121  
`\p{Zs}` 121