Regexes-and-Rules

Introducción a " Regexes and Rules" Synopsis 05, Perl6

renatolrr

Historia

Un poco de historia para rellenar.

Apocalipsis => Sinopsis

Exégesis

Sinopsis S05

Título:

Synopsis 05: Regexes and Rules

66 paginas

Sinopsis S05

- Autores:
  - Damian Conway <damian@conway.org>
  - Allison Randal <al@shadowed.net>
  - Patrick Michaud <pmichaud@pobox.com>
  - Larry Wall <larry@wall.org>
  - Moritz Lenz <moritz@faui2k3.org>
  - Tobias Leich <email@froggs.de>

Un poco más

- Perl 5 y perl6 son dos lenguajes diferentes, con una sintasis y semantica diferente, pero es que Perl:

- Filosofía: "hay más de un camino para hacerlo", " las cosas faciles..."
- Esructura (idioma, test...)
- Un edificio: Cpan.
- Una comunidad.

- Distintas implementaciones de perl. (Pugs, Rakudo, Elf, KindaPerl6, v6.pm,SMOP, STP.pm
- Rosseta code: http://rosettacode.org/wiki/Category:Perl_6

“Perl programmers tend to be lazy”

Una parada

- Espacios
- ~
  - Como operador.
  - Como operador binario.

Periodic Table of the Operators

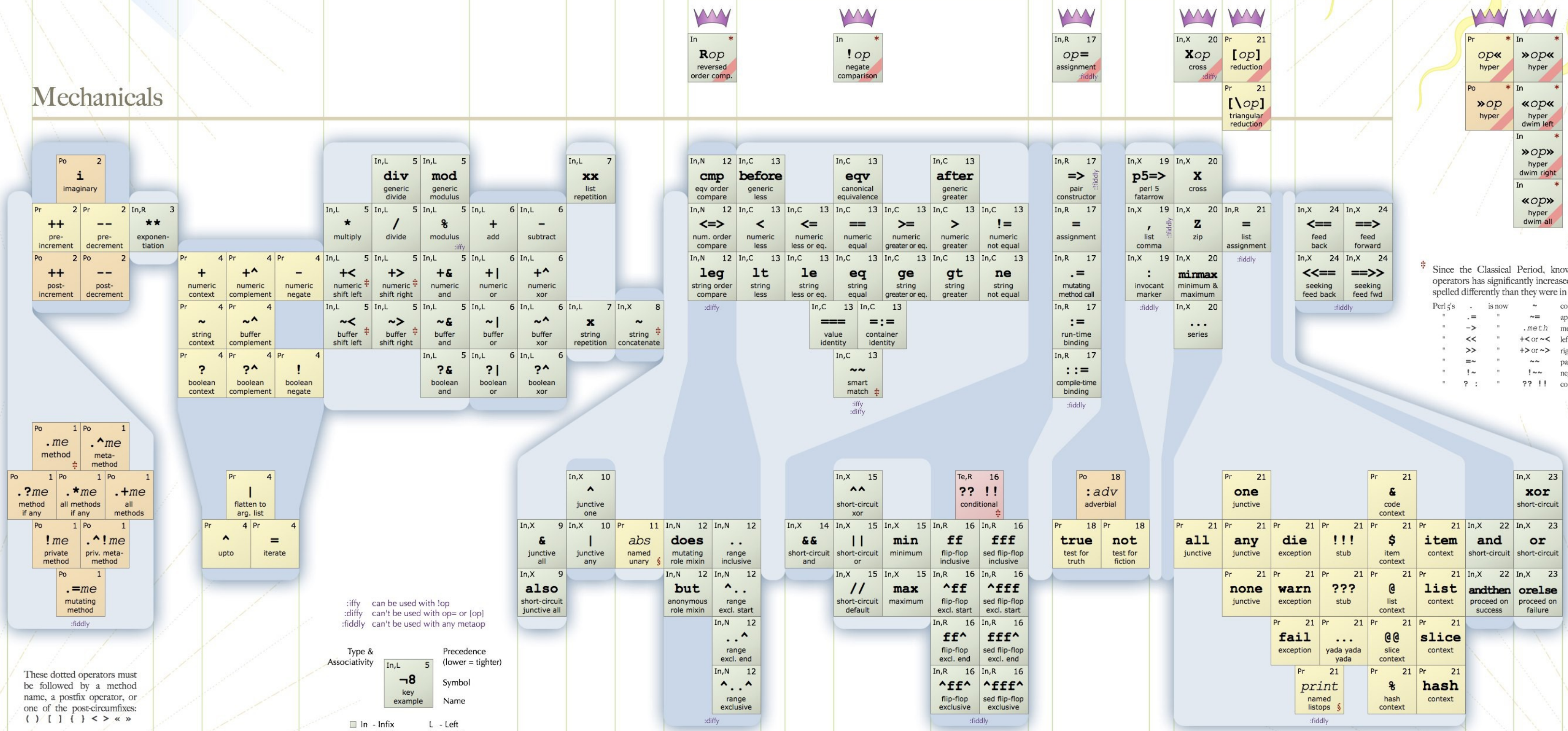
Being a comprehensive and complete enumeration of the Operatic Elements of the Perl 6 Language, assembled and drawn with dedication and diligence by M. Lentczner as a service to both the Community and the Republic.

May this simple presentation with various illustrative devices increase Knowledge & Understanding amongst practitioners in the art of Software. Forthwith the Third Edition, February 14th, Two Thousand Nine

EXEGESIS & SYNOPSIS

Perl 6

APOCALYPSE



Noble Metaops

XII Comparatist
- In * `Rop` reversed order comp.

XIII Relative
- In * `!op` negate comparison

XVII Assignogen
- In,R 17 `op=` assignment :fiddly

XX Lisztified
- In,X 20 `Xop` cross :diffy

XXI Lisztogen
- Pr 21 `[op]` reduction
- Pr 21 `[\op]` triangular reduction

- Pr * `op«` hyper
- In * `»op«` hyper
- Po * `»op` hyper
- In * `«op«` hyper dwim left
- In * `»op»` hyper dwim right
- In * `«op»` hyper dwim all

Mechanicals

II Incremental
- Po 2 `i` imaginary
- Pr 2 `++` pre-increment
- Pr 2 `--` pre-decrement
- Po 2 `++` post-increment
- Po 2 `--` post-decrement

III Expansive
- In,R 3 `**` exponentiation

IV Complementary
- Pr 4 `+` numeric context
- Pr 4 `+^` numeric complement
- Pr 4 `-` numeric negate
- Pr 4 `~` string context
- Pr 4 `~^` buffer complement
- Pr 4 `?` boolean context
- Pr 4 `?^` boolean complement
- Pr 4 `!` boolean negate

V Multiplicoid
- In,L 5 `div` generic divide
- In,L 5 `mod` generic modulus
- In,L 5 `*` multiply
- In,L 5 `/` divide
- In,L 5 `%` modulus :iffy
- In,L 5 `+<` numeric shift left ‡
- In,L 5 `+>` numeric shift right ‡
- In,L 5 `+&` numeric and
- In,L 5 `~<` buffer shift left ‡
- In,L 5 `~>` buffer shift right ‡
- In,L 5 `~&` buffer and
- In,L 5 `?&` boolean and

VI Addenoid
- In,L 6 `+` add
- In,L 6 `-` subtract
- In,L 6 `+|` numeric or
- In,L 6 `+^` numeric xor
- In,L 6 `~|` buffer or
- In,L 6 `~^` buffer xor
- In,L 6 `?|` boolean or
- In,L 6 `?^` boolean xor

VII Replicant
- In,L 7 `xx` list repetition
- In,L 7 `x` string repetition

VIII Concatenoid
- In,X 8 `~` string concatenate ‡

XII Comparatist
- In,N 12 `cmp` eqv order compare
- In,N 12 `<=>` num. order compare
- In,N 12 `leg` string order compare

:diffy

XIII Relative
- In,C 13 `before` generic less
- In,C 13 `eqv` canonical equivalence
- In,C 13 `after` generic greater
- In,C 13 `<` numeric less
- In,C 13 `<=` numeric less or eq.
- In,C 13 `==` numeric equal
- In,C 13 `>=` numeric greater or eq.
- In,C 13 `>` numeric greater
- In,C 13 `!=` numeric not equal
- In,C 13 `lt` string less
- In,C 13 `le` string less or eq.
- In,C 13 `eq` string equal
- In,C 13 `ge` string greater or eq.
- In,C 13 `gt` string greater
- In,C 13 `ne` string not equal
- In,C 13 `===` value identity
- In,C 13 `=:=` container identity
- In,C 13 `~~` smart match ‡

:iffy
:diffy

XVII Assignogen
- In,R 17 `=>` pair constructor :fiddly
- In,R 17 `=` assignment
- In,R 17 `.=` mutating method call
- In,R 17 `:=` run-time binding
- In,R 17 `::=` compile-time binding

:fiddly

XIX Comic
- In,X 19 `p5=>` perl 5 fatarrow
- In,X 19 `,` list comma :fiddly
- In,X 19 `:` invocant marker

:fiddly

XX Lisztified
- In,X 20 `X` cross
- In,X 20 `Z` zip
- In,X 20 `minmax` minimum & maximum
- In,X 20 `...` series

XXI Lisztogen
- In,R 21 `=` list assignment

:fiddly

XXIV Locomotive
- In,X 24 `<==` feed back
- In,X 24 `==>` feed forward
- In,X 24 `<<==` seeking feed back
- In,X 24 `==>>` seeking feed fwd

:fiddly

‡ Since the Classical Period, knowledge of these operators has significantly increased. They are now spelled differently than they were in ancient times:

Perl 5's	.	is now	~	concatenate
"	.=	"	~=	append
"	->	"	.meth	method call
"	<<	"	+< or ~<	left shift
"	>>	"	+> or ~>	right shift
"	=~	"	~~	pattern match
"	!~	"	!~~	negated pat. match
"	? :	"	?? !!	conditional

Transmissives

I Methodic
- Po 1 `.me` method ‡
- Po 1 `.^me` meta-method
- Po 1 `.?me` method if any
- Po 1 `.*me` all methods if any
- Po 1 `.+me` all methods
- Po 1 `!me` private method
- Po 1 `.^!me` priv. meta-method
- Po 1 `.=me` mutating method

:fiddly

These dotted operators must be followed by a method name, a postfix operator, or one of the post-circumfixes: ( ) [ ] { } < > « »

IV Raveler
- Pr 4 `|` flatten to arg. list
- Pr 4 `^` upto
- Pr 4 `=` iterate

:iffy can be used with !op
:diffy can't be used with op= or [op]
:fiddly can't be used with any metaop

Type & Associativity — In,L — Precedence (lower = tighter) — 5

Symbol — ¬8

Name — key example

- ▢ In - Infix
- ▢ Pr - Prefix
- ▢ Po - Postfix
- ▢ Te - Ternary

- L - Left
- R - Right
- N - None
- C - Chaining
- X - List

* - Metaops that take on precedence and associativity of enclosed op.

§ The named unary and named listop entries represent a vast expanse of alphabetically named functions. Messrs. Adams & Wall are leading the expedition to uncover the exotic properties and extent of this range. Please see their dispatches to the Republic entitled "Synopsis 29: Builtin Functions"

Philosophics

IX Jungian
- In,X 9 `&` junctive all
- In,X 9 `also` short-circuit junctive all

X Jungior
- In,X 10 `^` junctive one
- In,X 10 `|` junctive any

XI Nominal
- Pr 11 `abs` named unary §

XII Emotive
- In,N 12 `does` mutating role mixin
- In,N 12 `..` range inclusive
- In,N 12 `but` anonymous role mixin
- In,N 12 `^..` range excl. start
- In,N 12 `..^` range excl. end
- In,N 12 `^..^` range exclusive

:diffy

XIV Andite
- In,X 14 `&&` short-circuit and

XV Orite
- In,X 15 `^^` short-circuit xor
- In,X 15 `||` short-circuit or
- In,X 15 `min` minimum
- In,X 15 `//` short-circuit default
- In,X 15 `max` maximum

XVI Ambivalent
- Te,R 16 `?? !!` conditional ‡
- In,R 16 `ff` flip-flop inclusive
- In,R 16 `fff` sed flip-flop inclusive
- In,R 16 `^ff` flip-flop excl. start
- In,R 16 `^fff` sed flip-flop excl. start
- In,R 16 `ff^` flip-flop excl. end
- In,R 16 `fff^` sed flip-flop excl. end
- In,R 16 `^ff^` flip-flop exclusive
- In,R 16 `^fff^` sed flip-flop exclusive

:fiddly

XVIII Polygraphic
- Po 18 `:adv` adverbial
- Pr 18 `true` test for truth
- Pr 18 `not` test for fiction

XXI Liszt Op.
- Pr 21 `one` junctive
- Pr 21 `&` code context
- Pr 21 `all` junctive
- Pr 21 `any` junctive
- Pr 21 `die` exception
- Pr 21 `!!!` stub
- Pr 21 `$` item context
- Pr 21 `item` context
- Pr 21 `none` junctive
- Pr 21 `warn` exception
- Pr 21 `???` stub
- Pr 21 `@` list context
- Pr 21 `list` context
- Pr 21 `fail` exception
- Pr 21 `...` yada yada yada
- Pr 21 `@@` slice context
- Pr 21 `slice` context
- Pr 21 `print` named listops §
- Pr 21 `%` hash context
- Pr 21 `hash` context

:fiddly

XXII Looser Andite
- In,X 22 `and` short-circuit
- In,X 22 `andthen` proceed on success

XXIII Looser Orite
- In,X 23 `xor` short-circuit
- In,X 23 `or` short-circuit
- In,X 23 `orelse` proceed on failure

PROGRAMMING REPUBLIC OF PERL

Regexes

- Diferencia.
  - No se utiliza ni PCRE (Perl compatible Regular Espressions) ni POSIX (Portable Operating System Interface for Unix).
  - Regex: expresiones regulares, como patrones de busqueda.

Regexes

- Explicación, buscando patrones
- Síntasis más clara.
- Necesidad de capturar y encontrar objetos.
- Nombrando "regexes" y "grammars"

- Tablas.

- Modificadores.

Regexes

- Sustitución.

```
my $spacey = 'with   many spaces';`
say $spacey.subst(rx/ \s+/, '',:g);
\# output: with many superfluous spaces
```

Regexes

- $/
-

Perl 5...

```
#
#  $1-----------------------  $4-----------  $5-------------------
#  |   $2----------------| |              |  | $6----  $7------- |
#  |   |         $3---   | |  |           |  | |     | |       | |
#  |   |         |   |   | |  |           |  | |     | |       | |
m/ ( A (guy|gal|g(\S+)   ) )  (sees|calls)  ( (the|a) (gal|guy) ) /x;
```

Perl 6:

```
# Perl 6...
#
#  $0-----------------------  $1-----------  $2-------------------
#  |   $0[0]-----------    |  |           |  | $2[0]-  $2[1]---- |
#  |   |      $0[0][0]  |  |  |           |  | |     | |       | |
#  |   |         |   |  |  |  |           |  | |     | |       | |
m/ ( A (guy|gal|g(\S+)  )  )  (sees|calls)  ( (the|a) (gal|guy) ) /;
```

Grammar

- Grammar
- Rules como frases

```
rule TOP {
 <protocol>'://'<address>
 }
```

- Token como palabras

```
token number {
  \d+ ['.' \d+]?
}
```

- Protos define un tipo de rule o token

```
proto token protocol {*}
 token protocol:sym<http> {
     <sym>
  }
  token protocol:sym<https> {
      <sym>
  }
```

- Parse (TOP)

Grammar

```
#!/usr/bin/perl6
use v6;
grammar URL {
    token TOP {
      <schema> '://'
      [<ip> | <hostname> ]
      [ ':' <port>]?
      '/' <path>?
   }
    token byte {
      (\d**1..3) <?{ $0 < 256 }>
    }
    token ip {
      <byte> [\. <byte> ] ** 3
    ...
    token path {
      <[ a..z A..Z 0..9 \-_.!~*'():@&=+$,/ ]>+
    }
  }

  my $match = URL.parse('http://perl6.org/documentation/');
  say $match<path>;       # perl6.org
```

Modulos grammar

- Json
- Xml
- Grammar

Perl6::grammars

- Limitaciones

- Regexp::Grammars

Enlaces

- "Apocalypse 5: Pattern Matchin": http://www.perl6.org/archive/doc/design/apo/A05.html
- "Exegesis 5: Pattern Matching": http://www.perl6.org/archive/doc/design/exe/E05.html
- "Synopsis 5: Regexes and Rules": http://perlcabal.org/syn/S05.html
- "Introduction to Perl 6 Regex": https://github.com/perlpilot/perl6-docs/blob/master/intro/p6-regex-intro.pod
- "Regexes -- Pattern matching against strings": http://doc.perl6.org/language/regexes
- "Perl 5 to Perl 6": http://perlgeek.de/en/article/5-to-6
- "Perl 6 and Parrot Essentials, 2nd Edition": http://shop.oreilly.com/product/9780596007379.do

Ruegos y preguntas.

Contestaciones ??

Gracias