Co słychać w Perlu 6

Opowieść o Perlu, Rakudo i wydarzeniach bieżących

Tadeusz Sośnierz

19 lipca 2010

Dlaczego warto dziś mówić o Perlu 6?

Larry Wall

„Perl 5 was my rewrite of Perl. I want Perl 6 to be the community's rewrite of Perl and of the community."

Dlaczego warto dziś mówić o Perlu 6?

Larry Wall

„Perl 5 was my rewrite of Perl. I want Perl 6 to be the community's rewrite of Perl and of the community."

- 19 lipca 2000 – ogłoszenie Perla 6 – Równo 10 lat temu!

Dlaczego warto dziś mówić o Perlu 6?

Larry Wall

„Perl 5 was my rewrite of Perl. I want Perl 6 to be the community's rewrite of Perl and of the community."

- 19 lipca 2000 – ogłoszenie Perla 6 – Równo 10 lat temu!
- Perl 6 nie jest już tylko specyfikacją

Dlaczego warto dziś mówić o Perlu 6?

Larry Wall

„Perl 5 was my rewrite of Perl. I want Perl 6 to be the community's rewrite of Perl and of the community."

- 19 lipca 2000 – ogłoszenie Perla 6 – Równo 10 lat temu!
- Perl 6 nie jest już tylko specyfikacją
- Za 10 dni (29 lipca) wydane zostanie Rakudo Star – dystrybucja Perla 6 zdatna do użytku

Dlaczego warto dziś mówić o Perlu 6?

Larry Wall

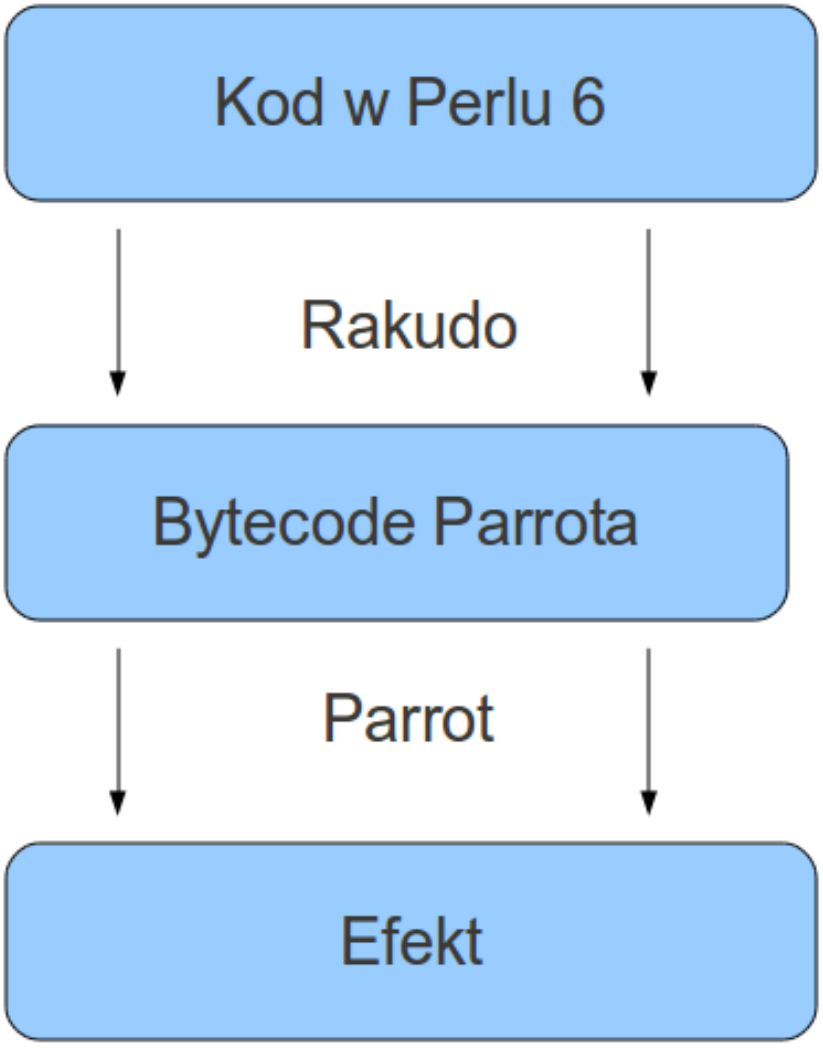
„Perl 5 was my rewrite of Perl. I want Perl 6 to be the community's rewrite of Perl and of the community."

- 19 lipca 2000 – ogłoszenie Perla 6 – Równo 10 lat temu!
- Perl 6 nie jest już tylko specyfikacją
- Za 10 dni (29 lipca) wydane zostanie Rakudo Star – dystrybucja Perla 6 zdatna do użytku
- W ten czwartek (22 lipca) zostanie wydane Rakudo #31, na którym będzie oparte Rakudo Star

Camelia

»ö«

Rakudo i Parrot



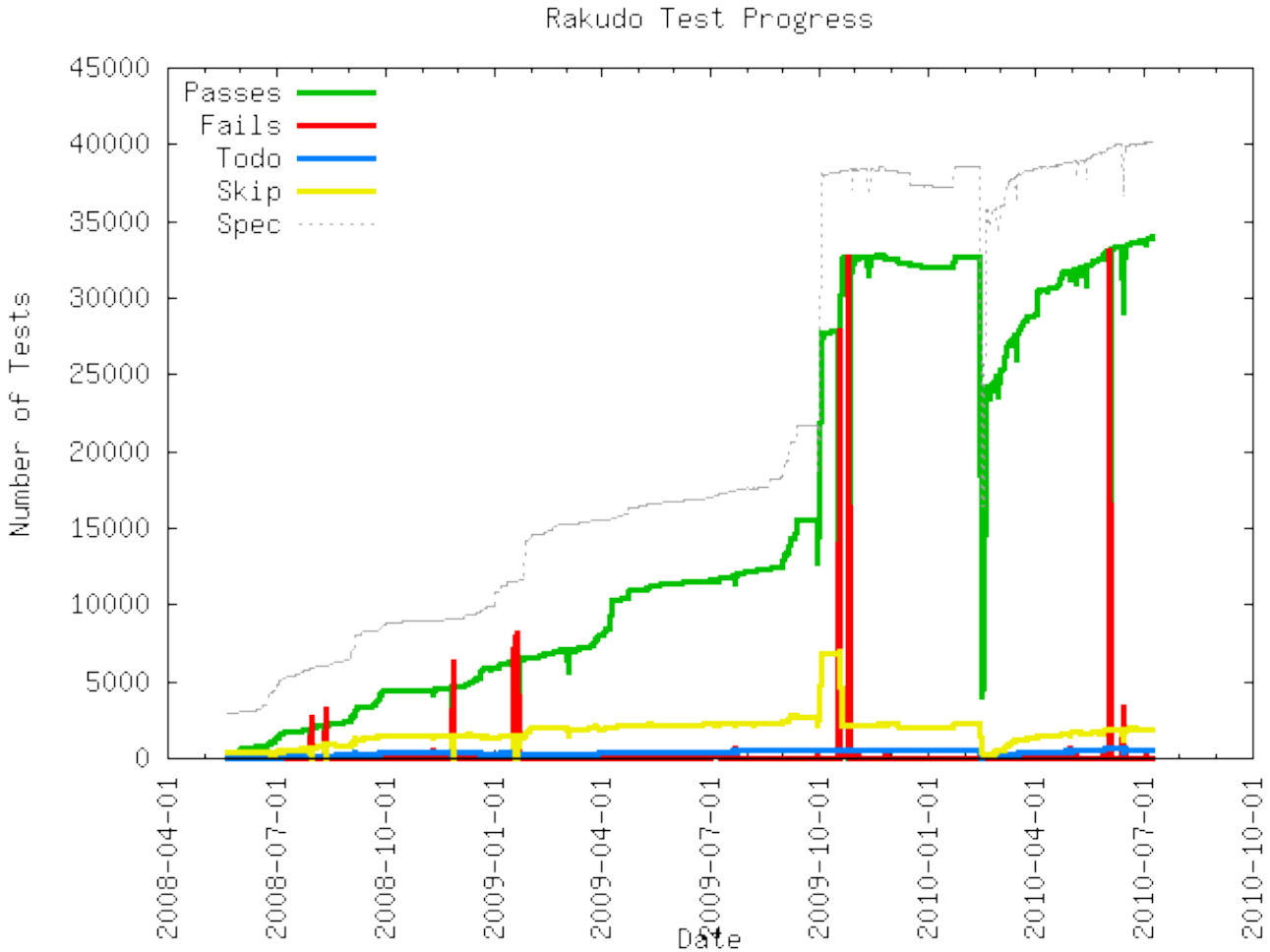
Parrot

- Maszyna wirtualna dla języków dynamicznych
- Dostępne implementacje Perla 6, Pythona, Ruby i innych
- Wydawana regularnie co miesiąc, jutro (20.07) wersja 2.6.0

Rakudo

- (Jap.) „The Way Of The Camel”, tudzież „Paradise”
- Kompilator Perla 6 w Perlu 6
- Również wydawany co miesiąc, dwa dni po wydaniu Parrota
- Aktualnie spełnia około 83% testów ze specyfikacji

Rakudo – postęp



Rakudo Star

- „useful and usable”
- Nie będzie to kompletna implementacja, ale zdatna do użytku
- Ma przyciągnąć uwagę i zachęcić do portowania modułów i pisania kodu
- Planowane wydanie – 29 lipca br.

Rakudo Star

A wraz z nim:

- Parrot
- Blizkost (o nim za chwilę)
- Moduły
  - Zavolaj (native call interface)
  - MiniDBI (subset DBI)
  - ...
- „The Perl 6 Book” (http://github.com/perl6/book)

Blizkost

Daje możliwość używania Perla 5 jako jednego z języków w Parrocie, co pozwala nam używać modułów z Perla 5 w kodzie w Perlu 6

blikost/examples/cgi.pl

```
use v6;
use CGI:from<perl5>;
my $q = CGI.new;

print $q.header, $q.start_html('Hello World'),
      $q.h1('Hello World'), $q.end_html;
```

Zavolaj

„Wołanie” funkcji z C bezpośrednio w Perlu 6

zavolaj/examples/sqlite3.p6 (fragmenty)

```
use NativeCall;
sub sqlite3_open( Str $filename, OpaquePointer $ppDB )
    returns Int
    is native('libsqlite3')
    { ... }

my OpaquePointer $db;
my $status = sqlite3_open("test.db", $db);
```

Moduły – proto, pls

- proto – instalator modułów Perla 6
- nowy projekt: pls (ma zastąpić proto)
- http:/modules.perl6.org – baza modułów
  - Math::Model
  - MiniDBI
  - LWP::Simple
  - SVG
  - ufo
  - URI
  - XML, XML::Writer
  - Web
  - HTTP::Server::Simple, HTTP::Server::Simple::PSGI (niedostępne w proto)

Inne implementacje

- PUGS (nierozwijany)
- YAPSI – Yet Another Perl Six Implementation (http://github.com/masak/yapsi)
- Niecza – kompilator Perla 6 pod .NET/Mono (http://github.com/sorear/niecza)
- Bennu – kompilator Perla 6 pod LLVM (http://github.com/ekiru/Bennu)

Zmiany, zmiany, zmiany

Co się zmieniło w samym języku?

Czego nie lubimy w Perlu 5

- Surowego OOP
- Dostępu do parametrów w funkcjach
- Łapania wyjątków
- Momentami nienajpiękniejszej składni (referencje)

Co się zmieniło

- OOP – wszystko jest obiektem, nowa składnia tworzenia klas (podobna do tej z MooseX::Declare)
- Regexpy, na inne, niekompatybilne z tymi z Perla 5, acz o dużo większych możliwościach
- Składnia – miejscami uproszczona, ale i dodane mnóstwo nowych elementów

Ale to wciąż Perl

Jakich modułów nie będziemy już potrzebować

- Moose
- Try::Tiny
- Data::Dumper – każda zmienna ma metodę .perl
- Devel::REPL – REPLa mamy w standardzie
- Getopt::* – możemy ustalić listę parametrów dla funkcji MAIN (o tym wkrótce)

Typowanie

Jest możliwość nadania zmiennej typu

```
> my Int $a = 5; $a = "foo"
Type check failed for assignment

> my Str $a = "foo"; $a = 5
Type check failed for assignment
# ale...
> my Str $a = "foo"; $a = 5.Str; $a.perl.say
"5"

> say ~[5.WHAT, 'string'.WHAT, (3/7).WHAT, /foo .*/.WHAT]
Int() Str() Rat() Regex()
```

Wszystko jest obiektem

```
> 'a string'.^methods.sort[40..45]
bytes can capitalize ceiling chars chomp
> "the big brown fox".split(' ').grep(/^b/).join(' and ')
big and brown
```

Prefixy zmiennych

W Perlu 6 prefix zmiennej (sigil) nie zmienia się przy wydobywaniu pojedynczego elementu

Perl 5	Perl 6
my @arr = (1, 2, 3) my $elem = $arr[1]	my @arr = (1, 2, 3) my $elem = @arr[1]
my %hash = (foo => 'bar') my $elem = $hash{'foo'}	my %hash = (foo => 'bar') my $elem = %hash{'foo'}

Wszystko jest referencją

Nie ma rozróżnienia na zmienne i referencje do nich, a więc nie ma już problemów z dereferowaniem tychże.

```
my $a = [ { foo => [1, 2, 3] } ];
```

Perl 5

```
say $a->[0]->{foo}->[1] # 2
```

Perl 6

```
say $a[0]<foo>[1] # 2
```

Funkcje i parametry

```
sub foo (Str $what, Int $times = 1) {
    say $what x $times
}

foo "hello", 5; # hellohellohellohellohello
foo "hello";    # hello
foo;            # Not enough positional parameters passed;
                # got 0 but expected between 1 and 2
```

Funkcja MAIN

```
sub MAIN($v?, :$arg!, :$arg2 = 'wartość domyślna') {
    if ($v) { # verbose
        say "Zaczynamy"
    }
    say "arg: $arg, arg2: $arg2"
}

$ perl6 main.pl
Usage:
./main.pl [-arg2=value-of-arg2] -arg=value-of-arg [v]

$ perl6 main.pl -v -arg=5
Zaczynamy
arg: 5, arg2: wartość domyślna

$ perl6 main.pl -arg=5 -arg2=foo
arg: 5, arg2: foo
```

Junctions

```
> say 'ok' if 5 == any(3, 5, 7) # albo 5 == 3 | 5 | 7
ok
> say 'ok' if 5 == none(4, 6, 8)
ok
> my Junction $x = 3 | 5; say 'ok' if $x == 5
ok
> my @scores = 32, 41, 73, 99, 52;
> say 'ok' if all(@scores) > 30;
ok
```

Laziness

```
> my $even = (2, 4 ... *); $even[^10].perl.say
(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
> my $sq = gather for 0..Inf { take $_ * $_ }; $sq[5].say
25
```

Try-CATCH

```
try {
    die "Oh noes!";
    CATCH {
        say "Something went wrong: $_";
    }
}
```

OOP

Perl 5 z Moose

```
package Point;
use Moose;

has ['x','y'] => (is => 'rw', isa => 'Int');

sub clear {
    my $self = shift;
    $self->x(0);
    $self->y(0);
}

package Point3D;
use Moose;
extends 'Point';

has 'z' => (is => 'rw', isa => 'Int');

after 'clear' => sub {
    my $self = shift;
    $self->z(0);
};
```

Perl 6

```
class Point {

  has Int $.x is rw;
  has Int $.y is rw;

  method clear {
    $.x = $.y = 0;
  }

}

class Point3D is Point {

  has Int $.z is rw;

  method clear {
    nextsame;
    $.z = 0;
  }

}
```

Gramatyki

```
grammar URI {
    token TOP {
        <schema> '://'
        [<hostname> | <ip> ]
        [ ':' <port>]?
        '/' <path>?
    }
    token byte {
        (\d**{1..3}) <?{ $0 < 256 }>
    }
    token ip {
        <byte> [\. <byte> ] ** 3
    }
    token schema {
        \w+
    }
    token hostname {
        (\w+) ( \. \w+ )*
    }
    token port {
        \d+
    }
    token path {
        <[ a..z A..Z 0..9 _\-.!~*'():@&=+$,/ ]>+
    }
}
my $match = URI.parse('http://perl6.org/documentation/');
say $match<hostname>;  # perl6.org
say $match<path>;      # documentation
```

Co możemy zrobić teraz

Co możemy zrobić poza bezczynnym czekaniem na Rakudo Star?

- Pisać kod, szukać bugów
- Robić pozytywny szum o Perlu 6 :)

Linki

- http://perl6.org/
- http://parrot.org
- http://rakudo.org
- http://perlgeek.de/en/article/5-to-6 – wprowadzenie do Perla 6 dla programistów Perla 5
- http://perl6advent.wordpress.com/ – Perl 6 Advent Calendar, cykl ciekawych artykułów o nowościach w Perlu 6
- Kanał #perl6 na irc.freenode.net