Perl och Raku - de första 50 åren

Creating possibilities together

Jonas Linde <jonas.linde@b3.se>

Agenda

- Perl
  - Historik
  - Logotyper
  - Bakåtkompatibilitet
  - Objektorientering
  - Webbramverk
  - Perl 44
- Raku
  - Historik
  - Sigill
  - Twigils
  - Typer
  - Unicode
  - Samtidighet
  - Stil
  - Reguljära uttryck

Perl

Historik

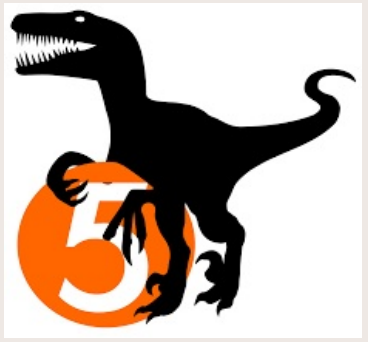
- 1987 - Practical Exctraction And Report Language
- 1988 - Perl 2 - regexes
- 1989 - Perl 3 - binärdata
- 1991 - Perl 4 - *Programming Perl*
- 1994 - Perl 5 - moduler, objekt, lexikaler med mera
- 1994 - CGI.pm
- 1995 - CPAN - *Comprehensive Perl Archive Network*
- 2004 - Perl 5.6 - Unicode, 64 bitar, filer > 2GiB
- 2007 - Perl 5.10 - switch
- 2025 - Perl 5.42

Logotyper

- *Programming Perl* © 1991 O'Reilly

- *The Perl and Raku Foundation* © ~1999

Logotyper

- *Perl 5 raptor* © 2012 kraih 
- *Perl Logo* © 2024 TPRF 

Bakåtkompatibilitet

- Perl 5.42 är bakåtkompatibelt med Perl 5.0
- Perl 6 var avsett att vara bakåtkompatibelt men är ju nu sitt eget språk
  - fast det går att köra Perl 5-kod i Raku
- Perl 7 avbröts eftersom den inte skulle bli helt bakåtkompatibel
- Nyare features har inte perfekt bakåtkompatibilitet
  - `use experimental '...';`

Objektorientering

- Objekt introducerades i Perl 5.000
  - ett objekt är en referens till en datatyp som vet vad den heter
  - upplevs som ett hack
- *Moose* - modul med full objektorientering
- *Mouse* - mindre är Moose
- *Moo* - ännu mindre
- Perl 5.38 - `use experimental 'class';`

Webbramverk

- CGI.pm - *Common Gateway Interface*
  - först och nu omodernt
- CGI::Fast
  - snabbare men fortfarande omodernt
- Dancer
  - enklast
- Mojlicious
  - bättre men ansträngd bakåtkompatibilitet
- Catalyst
  - overkill för de flesta

Perl 44

- Perl 6 är ju redan taget
- Utvecklingen av Perl 7 avbröts
- Senaste Perl-version är 5.42
- Utveckling: Perl 5.43
- Nästa version: Perl 44 - kanske

jonas.linde@b3.se

Raku

Tidslinje

- 1999 - Frustration
- 2000 - 361 RFCs - önskemål
- 2001 - 12 Apocalypses - övergripande design
- 2001 - Exegeses - förklaringar
- 2004 - Synopses - renodlingar
- 2005 - Pugs - kompilator - pensionerad
- 2008 - Rakudo + Roast - kompilator + definierande tester

Tidslinje

- 2009 - Parrot - bytekodsmaskin - pensionerad
- 2010 - Yapsi - kompilator - pensionerad
- 2010 - Niecza - kompilator - pensionerad
- 2013 - MoarVM - bytekodsmaskin
- 2015 - Fokus
- 2015-12-25 - Perl 6.c - *Christmas*
- 2019 - Perl 6 => Raku
- 2020-10-24 - Raku 6.d - *Diwali*
- 2026 - Rakudo 2026.02

Sigill

Sigillen $, @, % och & anger variabeltyp - samma i Perl och Raku

- $ skalär
- @ lista
- % symbolisk lista
- & funktion
- * något av ovanstående beroende på kontext
- \ inget av ovanstående

Twigill

Utöver sigillen $, @, % och & finns sekundära sigill:

- * Dynamic
- ? Compile-time variable
- ! Attribute (class member)
- . Method (not really a variable)
- < Index into match object (not really a variable)
- ^ Self-declared formal positional parameter
- : Self-declared formal named parameter
- = Pod variables
- ~ The sublanguage seen by the parser at this lexical spot

Twigill-exempel

- `@!var[2]`
- `$<hostname>`
- `%!privat`

jonas.linde@b3.se

Typning

- Java - statisk typning:

```
private static float foo(boolean a, String b) { ... }
```

- Perl 5 - dynamisk typning:

```
sub foo { my ($a, $b) = @_; ... }
```

- Perl 6 - gradvis typning:

```
sub foo($a, Str $b) returns Rat { ... }
```

233 typer

- `Str`
- `Buf`
- `Bool`
- `Int`
- `int`
- `Rat`
- `Num`
- `Complex`
- …

Rat

```
> print 1/10
0

> print 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1
0.9999999999999999

> s:=0.; for i:=0; i<10; i++ {s+=.1}; fmt.Println(s);
0.9999999999999999

> console.log(1/10+1/10+1/10+1/10+1/10+1/10+1/10+1/10+1/10+1/10);
0.9999999999999999

> IO.puts 1/10+1/10+1/10+1/10+1/10+1/10+1/10+1/10+1/10+1/10;
0.9999999999999999

> my $a = 0; for (1..10) {$a = $a + 1/10}; say $a;
1
```

Unicode i koden

```
> say ¼
0.25

> say 2³
8

> say ৺ + ৴
10

> say τ
6.28318530717959

> my \π = 4;
> say π;
4
```

Unicode i data

- strängar konverteras till “Normalization Form Grapheme”

```
> say "a\c[COMBINING RING ABOVE]" eq "å"
True

> say "\r\n".chars
1

> say "\r\n" eq "\n"
False
```

- vill man hantera bytes kan man använda typen Buf
- utf-8 är default vid konvertering

Promise

```
> my @promises;
> for 1..5 -> $t {
>     push @promises, start {
>         sleep $t;
>         my $r = rand;
>         die if $r < 0.2;
>     };
> }
> await Promise.allof(@promises);
> say @promises>>.status;

[Kept Kept Kept Kept Broken]
```

Supply

```
> my $supply = supply {
>     for 1 .. 10 {
>         emit($_);
>     }
> }
> $supply.tap(->$v { print "$v " }); say '';
> $supply.tap(->$v { print "$v " }); say '';

1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
```

Channel

```
> my $channel = Channel.new;
> start {
>     my $closed = $channel.closed;
>     loop {
>         last if $closed;
>         print $channel.receive, ' ';
>     }
>     say '';
> }
>
> for ^10 -> $t {
>     sleep $t;
>     $channel.send($t);
> }
> $channel.close;

0 1 2 3 4 5 6 7 8 9
```

Proc::Async

```
> my $proc = Proc::Async.new(:w, 'grep', 'foo');
>
> $proc.stdout.tap(-> $v { print "Output: $v" });
> $proc.stderr.tap(-> $v { print "Error:  $v" });
>
> my $promise = $proc.start;
> $proc.say("this line has foo");
> $proc.say("this one doesn't");
> $proc.close-stdin;
> await $promise;
>
> say "Done.";

Output: this line has foo
Done.
```

Procedurell stil

```
> sub postfix:<!> { [*] 1..$^n }
> say 5!;

120
```

Objektorienterad stil

```
> class Trip is Journey does Transport {
>     has $.origin;
>     has $.destination;
>     has @!travellers;
>     has $.notes is rw;
>
>     method go(Rat $speed) { … }
>     method !homesick { … }
> }
```

Funktionell stil

- *Rena funktioner* returnerar alltid samma resultat med samma parametrar
- *Rena funktioner* har inga sideffekter
- Raku har allt som Haskel har
- inklusive makron

Metaprogrammering

- Abstract Syntax Tree:

```
> my $ast = "42 + 666".AST.statements.head.expression;
> say $ast;

RakuAST::ApplyInfix.new(
  left  => RakuAST::IntLiteral.new(42),
  infix => RakuAST::Infix.new("+"),
  right => RakuAST::IntLiteral.new(666)
)
```

Regexes

- Perl:

```
> $url ~= m|http://([^:/]+)(:(\d+))?|i;
> system "openssl s_client", "-host", $1,  "-port", $3||443;
```

- Raku:

```
> $url ~~ /^ "http://" $<host> = (<-[:/]>+) [":" $<port> = (\d+)]? /;

> $url ~~ m :ignorecase /
>     ^ "http://"
>     $<host> = ( <-[:/]> + )
>     [ ":" $<port> = ( \d + ) ]?
> /;

> my $sclient = run <openssl s_client -host>, $<host>,
>                   "-port", $<port>//443, :out, :in, :err;
```

Grammars

```
> grammar Calculator {
>     token TOP { [ <add> | <sub> ] }
>     rule  add { <num> '+' <num> }
>     rule  sub { <num> '-' <num> }
>     token num { \d+ }
> }
>
> class Calculations {
>     method TOP ($/) { make $<add> ?? $<add>.made !! $<sub>.made; }
>     method add ($/) { make [+] $<num>; }
>     method sub ($/) { make [-] $<num>; }
> }
>
> say Calculator.parse('2 + 3', actions => Calculations).made;

5
```

Towards a Raku Foundation

- *The Perl and Raku Foundation* aka. *Yet Another Society* är av historiska skäl inriktat på Perl
- Under 2025 inleddes därför arbetet med att separera Perl och Raku i varsin stiftelse
  - Initial finansiering är säkrad
  - Stadgar finns som pull request
  - Ser ut att bli registrerad i Holland

Tack för ordet!