Perl6

Jonas Linde <jonas@init.se>

Agenda

- Historik
- Typer
- Unicode
- Samtidighet
- Stil

Historik

Tidslinje

- 1999 - Frustration
- 2000 - “Fix the language”
- 2001 - Apocalypses
- 2004 - Synopses
- 2008 - Rakudo + test suite
- 2015 - Fokus
- 2015-12-25 - Perl 6.0.0

Typer

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

Unicode i koden

```
> say ¼
0.25

> say 2³
8

> say ٣ + ٧
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

Samtidighet

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.
```

Stil

Objektorienterad programmering

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

Funktionell programmering

- *Rena funktioner* returnerar alltid samma resultat med samma parametrar
- *Rena funktioner* har inga sideffekter
- Perl6 har allt som Haskel har
- utom makron

Metaprogrammering

```
> sub postfix:<!> { [*] 1..$^n }
> say 5!;
120

> augment class Trip {
>     method picnic {…}
> };

> my Trip $adastra =
>    .new(destination => 'where no one has gone before');
> $adastra does role {
>    method launch { say "fwoosh!!!" }
> }
```

Tack för ordet!