Ten Things You Need To Know about Perl 6

Jeffrey Goff, Evozon Systems LLC
DevDays Vilnius 2017
https://github.com/drforr
http://theperlfisher.blogspot.ro

Where Do I Go For Help?

- http://www.perl6.org
- http://docs.perl6.org
- https://modules.perl6.org
- irc://irc.freenode.net #perl6

Where Do I Find New Toys?

- http://modules.perl6.org
- https://github.com/tadzik/panda
- https://github.com/ugexe/zef

Highlights

- Clean, single-pass parseable grammar
- User-Friendly error messages
- Unicode Friendly
- Sigils that make sense
- Math that works
- Function signatures
- Bootstrapped with regular expressions
- OO with a Metaprogramming model
- Custom operators
- Built-in Concurrency

Single-pass grammars

```
my @doctor = { :first('Christopher'), :last('Eccleston'), years => 1 },
             { :first('David'),       :last('Tennant'),   years => 4 },
             { :first('Matt'),        :last('Smith'),     years => 4 },
             { :first('Peter'),       :last('Capaldi'),   years => 3 };

say 'First New Who Doctor: ', @doctor[0]{'first'}, ' ', @doctor[0]<last>;

say "Average run: " ~ sum( map { .<years> }, @doctor ) /
@doctor.elems;
```

Error messages

```
sub mean( @a ) {
  my $sum = sum( @a )
  ( $sum / @a.elems )
}
say mean 1, 2, 4;

$ perl6 test.pl6

Two terms in a row across lines (missing semicolon or comma?)
at /home/jgoff/test.pl6:2
------>   my $sum = sum( @a )⏏<EOL>
    expecting any of:
        infix
        infix stopper
        ...
```

Unicode Friendly

```
my $α = 2 + ⅒  ;

say $α ÷ 2;

# 1.05
```

Or, **really** old-school

```
use Slang::Roman;
say 0rↃ  + 0rDCLXVI;

# 1666

my $a = 0rIV / 0rM;
say $a;

# 0.004
```

Sigils that make sense

```
my @powers = 1, 2, 4, 8;
my %foo = :name('Jeff'), :division('Programming');

say @powers[ 1, 2 ];
say %foo<name>;

# (2 4)
# Jeff
```

Fundamental Arithmetic

- perl -E 'say 0.1 + 0.2 – 0.3'
  - +5.55111512312578e-17
- python -c 'print 0.1 + 0.2 – 0.3'
  - +5.55111512312578e-17
- ruby -e 'print 0.1 + 0.2 – 0.3'
  - +5.55111512312578e-17
- perl6 -e 'say 0.1 + 0.2 – 0.3'
  - 0

Function Signatures

```
sub plus( Int $a, Int $b ) {
        $a + $b
}
say plus 1, 2;
# 3

say plus 1, 2, 3;
===SORRY!=== Error while compiling -e
Calling a(Int, Int, Int) will never work with declared signature
(Int $a, Int $b)
at -e:1
------> ub foo( Int $a, Int $b ) { $a + $b }; say ⏏foo 1, 2, 3;
```

Not so regular expressions

```
my $id = '[ { "first": "Christopher", "last": "Eccleston", "id": 1 } ]';

my regex Str        {                  "" (<-[ " ]>+) ""                 };
my regex Integer    {                          \d+                       };
my regex Value      {                  <Integer> | <Str>                 };
my regex Pair       {                <Str> ':' \s+ <Value>               };
my regex Pairs      {     '{' \s+   <Pair>+    %% (',' \s+) \s+ '}'     };
my regex List       { '[' \s+      <Pairs>+   %% (',' \s+)     \s+ ']' };

$id ~~ m{ <List> };

say $/;
```

Object Orientation

```
enum Chirality <left right>;

class Hand {
    has Chirality $.chirality;
    has $.content; }

role Humanoid {
    has Hand $.left-hand  .= new( :chirality( left ) );
    has Hand $.right-hand .= new( :chirality( right ) ); }

class Character does Humanoid {
    has Str $.name is required;
    has Int @.Attr where 0 < * <= 18;

    method attack( Character $enemy, Effort $effort ) { ... } }

class Sword { has Int $.Dmg }
```

Look Ma, Proper Lists!

```
my $fh = open 'sample.tsv', :r;
my @line;
for $fh.lines -> $line {
        my ( $id, $last-name, $first-name ) =
                $line.chomp.split( "\t" );
        @line[$id - 1] = $last-name, $first-name;
}
close $fh;
say @line;

# [("Tyler", "Rose"), ("Smith", "Mickey"),
   ("Jones", "Martha"), ("Noble", "Donna")]
```

Custom operators

```
multi sub prefix:<∑>( *@values ) {
    [+] @values
}

my $total = ∑ 0 ... ∞;

multi sub postfix:<⁺>( $x ) { $x.charge( +1 ) }
multi sub postfix:<⁻>( $x ) { $x.charge( -1 ) }
multi sub prefix:<√>( $x ) { sqrt( $x ) }

W⁺ = (-W¹ + iW²)/√2 ;
W⁻ = (-W¹ + iW²)/√2 ;
```

Perl 6 style

```
for 'sample.txt'.IO.lines.kv -> $index, $line {
	my ( $method, $id, $name, timestamp ) = $line.chomp.split( "\t" );

	given $method {
		when 'create'   { %object{$id} = $name, $timestamp }
		when 'delete'   { %object{$id}:delete }
	}
	if $method eq 'read' | 'update' | 'delete' {
		say "missing object on line $index" unless %object{$id}
	}
}
```

Junctions

```
my $needs-object = 'read' | 'update' | 'delete';
for 'sample.txt'.IO.lines.kv -> $index, $line {
	my ( $method, $id, $name, timestamp ) = $line.chomp.split( "\t" );

	given $method {
		when 'create'   { %object{$id} = $name, $timestamp }
		when 'delete'   { %object{$id}:delete }
	}
	if $method eq $needs-object {
		say "missing object on line $index" unless %object{$id}
	}
}
```