The Inspiration
coccoon?
Perl 5 Recap: 2000 - 2010

- 2000 - Perl 5.6
- 2002 - Perl 5.8
- 2007 - Perl 5.10
- 2010 - Perl 5.12 + yearly release
- The lean years have passed!
Perl 6 Recap: 2000 - 2010

- Camel Herders Meeting / Request for Comments
- Apocalypses, Exegeses, Synopses
- Parrot as a VM for everybody
- Pugs (on Haskell) / Perl 6 test-suite
- Rakudo (on Parrot) / Niecza (on mono/.NET)
- Nothing “Production Ready”
The 0’s - Cocooning Years

- Perl was busy with itself
- Redefining itself
- Re-inventing itself
- What is Perl?
- These years have passed!
Not your normal de-cocooning
Perl 5 and Perl 6 will co-exist for a long time to come!
Perl 5 in the 10’s

- A new release every year!
- Many optimisations, internal code cleanup
- Perl 6-like features: say, state, given/when, ~, //, ...
- Perl 6-like Modules: Moose / Moo / Mouse, Method::Signatures, Promises
- and a Monthly development release

5.20 is out! Go get it and use it!
Perl 6 in the 10’s

• Niecza more feature-complete, initially
• Not Quite Perl (NQP) developed and stand-alone
• 6model on NQP with multiple backends
• MoarVM - a Virtual Machine for Perl 6
• Rakudo runs on Parrot, JVM, MoarVM
• also a Monthly development release
Co-existence? Yes!
But Perl 6 will become larger and be more future proof!
Cool Perl 6 features in Perl 5

• say
• yada yada yada (…)
• state variables
• defined-or (//)
• lexical subs
• subroutine signatures
• OK as long as it doesn’t involve types
print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo

print "Foo
";
Foo

say "Foo";
Foo
print "Foo
";
Foo

say "Foo";
Foo

print "Foo\n";
Foo

say "Foo";
Foo

say "Foo";
Foo
sub a { ... }; a();
Unimplemented at -e line 1

my $a = ...;
syntax error at -e line 1, near "= ..."
sub a { ... }; a();
Stub code executed

my $a = ...;
say $a.WHAT; say $a;
(Failure)

===SORRY!===
Stub code executed
sub a { state $x = 10; ++$x }; say a for 0..9;

1
11
12
13
14
15
16
17
18
19
20
sub a { state $x = 10; ++$x }; say a for 0..9;
1
12
13
14
15
16
17
18
20

sub a { state $x = 10; ++$x } say a for 0..^10;
11
12
13
14
15
16
17
18
19
20
my $a; my $b = 42;
say $a // $b; say $a || $b;
42
42

my $a = 0; my $b = 42;
say $a // $b; say $a || $b;
0
42
my $a; my $b = 42;
say $a // $b; say $a || $b;
42
42
my $a = 0; my $b = 42;
say $a // $b; say $a || $b;
0
42
my $a; my $b = 42;
say $a // $b; say $a || $b;
42
42
my $a = 0; my $b = 42;
say $a // $b; say $a || $b;
0
42
my sub a { say "foo" }; a(); a();

foo

Undefined subroutine &main::a called at -e line 1.

say "foo"; a();

foo

Undefined subroutine &main::a called at -e line 1.
```perl
{ my sub a { say "foo" }; a(); };

say "foo"; a();

foo

Undefined subroutine &main::a called at -e line 1.

{ sub a { say "foo" }; a() };

===SORRY!=== Error while compiling -e

Undeclared routine: a used at line 1

say "foo"; a();

===SORRY!=== Error while compiling -e

Undeclared routine: a used at line 1
```
sub a ($f, %n) { say $f; say %n; }
a("bar", a => 1, b => 2);
bar
b2a1

sub a ($f, %n) { say $f; say %n; say @_; }
a("bar", a => 1, b => 2);
bar
b2a1
barb2a1

=begin
ERROR!
=end

sub a ($f, *%n) { say $f; say %n; }
a("bar", a => 1, :b<2>);
"a" => 1, "b" => 2).hash

sub a ($f, *%n) { say $f; say %n; say @_; }
a("bar", a => 1, b => 2);

---->
b a ($f, *%n) { say $f; say %n; say @_; }
⏏

; a("b
sub a ($f, %n) { say $f; say %n };  
a("bar", a => 1, b => 2);  
bar  
b2a1

sub a ($f, *%n) { say $f; say %n; say @_;  
a("bar", a => 1, b => 2);  
bar  
b2a1  
barb2a1

sub a ($f, *%n) { say $f; say %n; say @_;  
a("bar", a => 1, :b<2>);  
bar  
"a" => 1, "b" => 2).hash

sub a ($f, *%n) { say $f; say %n; say @_;  
a("bar", a => 1, b => 2);  
===SORRY!=== Error while compiling -e
Placeholder variable '@_' cannot override existing signature
at -e:1

--------> b a ($f, *%n) { say $f; say %n; say @_;  

a("b
Problematic Perl 6 features in Perl 5

• Standard async still dependent on ithreads
• Promises specific event loop dependent
• Smart match to be experimental / deprecated
• Subroutine signatures limited and only syntactic sugar
• **Bolting on** stuff later stays **difficult**
Perl 5 - Standard async

- Since 5.8 we have **ithreads**
- Which suffers from a flawed implementation
- But for which there is no real alternative
- Using fork() emulation intended for Windows
- Officially **discouraged** since 5.20
- Maybe **forks.pm** qualifies as an alternative
- Still, programming threads reliably is **superhuman**

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

--Brian Kernighan
Perl 5 - Promises

- Asynchronicity for non-superhumans
- Only available as a CPAN module
- Needs an event loop
- Cannot usually have more than one event loop
- Probably not really asynchronous
Perl 5 - Smart match

- You need typing for smart match to make sense
- Perl 5 will most likely **never** have typing
- Only of limited usefulness and source of confusion
- Hence marked **experimental** again
- And potentially **deprecated** in the future
Perl 5 - Subroutine signatures

• Just arrived with perl 5.20!

• Alas, syntactic sugar only

• And no real named parameters

• In Perl 6, part of multi-method dispatch

• **Nice** to have nonetheless!
sub a ({$name = "You"}) { say "Hey, $name!" };
a; a :name<Orlando>;
Hey, You!
Hey, Orlando!

say (name=>"Orlando").WHAT;
(Pair)
say (name=>"Orlando").perl;
"name" => "Orlando"
say :name<Orlando>.WHAT;
(Pair)
say :name<Orlando>.perl;
"name" => "Orlando"
multi a (Num $n) { say "Number $n" }; multi a (Str $a) { say "String $a" }; a(42); a("foo");
Number 42
String foo

my $a = "foo"; say $a;
foo
my Num $a = "foo"; say $a;
Type check failed in assignment to '$a'; expected 'Num' but got 'Str' in block at -e:1
my Str $a = "foo"; say $a;
foo
Perl 5 features in Perl 6

- “use v5”
- It stays hard to integrate / mimic the indescribable
use v5

• By TimToady, maintained by FROGGS (Tobias Leich)
• Re-implement Perl 5 just like Rakudo Perl 6
• Grammar / Action based, as a “slang”
• No XS (at least not as we know it)
• Call Perl 5 code from Perl 6 and vice-versa
• Now passes $\sim 10\%$ of Perl 5 test-suite
• Part of next Rakudo Star distribution!
Needed for Perl 6 adoption

• A good introduction (e)book
• More modules, CPAN support
• Better performance
A good introduction (e)book

• Rumour has it a certain someone is working on that
• Don’t let that stop you from writing your own!
• Or just blog about your experiences
• And let us know that you did!
More modules - CPAN support

• Can already upload Perl distributions to CPAN

• Can find distributions on CPAN

• Install from CPAN really soon with panda

• Effort to start porting CPAN module later this year
Better performance

• MoarVM is now standalone
• Performance on MoarVM creeping towards Perl 5
• Startup Perl 5 + Moose about same as Perl 6
• Code introspection for optimization built-in
• GSoC project to develop JIT for MoarVM underway
• First JITted code execution already seen!
Why use Perl 6 in production?

- Saner implicit/explicit multi-core programming
  - Channels, Promises, Supplies…
- No versioning issues with modules
- Cool features = happier programmers
use Test; ok 42,"foo";
ok 1 - foo

{ use Test }; ok 42,"foo";

===SORRY!=== Error while compiling -e
Undeclared routine:
  ok used at line 1. Did you mean 'on'?
for ^10 { rand.sleep; .print }; 0123456789
real 0m5.486s
user 0m0.259s
sys 0m0.051s

await do for ^10 { start { rand.sleep; .print } }; 6783524091
real 0m1.217s
user 0m0.260s
sys 0m0.051s
my $a = 1|2|3; say $a;
any(1,2,3)
my $a = 1|2|3; say $a == 1;
any(True, False, False)
my $a = any(1..3); say so $a == 1;
True
say (0..^100).pick;
71
my $a = any(^100); say so $a == (^100).pick;
True
my $a = any(^100); say so $a == (^1000).pick;
True
my $a = any(^100); say so $a == (^1000).pick;
False
How to try Perl 6
(as a user)

• Rakudo with plenty of modules to try
• Next version will have v5 most likely
• http://rakudo.org/downloads/star
How to try Perl 6
(as a tester)

• The Rakudo equivalent of perlbrew:

• https://github.com/tadzik/rakudobrew
How to try Perl 6
(as a contributor)

• mkdir foo && cd foo

• git clone https://github.com/rakudo/rakudo.git

• cd rakudo

• perl Configure.pl —gen-moar

• make install

• install/bin/perl6 -v
Examples?

- Rosetta Code
- [http://perl6.org/community/rosettacode](http://perl6.org/community/rosettacode)
- Perl 6 Advent Calendar
- [https://perl6advent.wordpress.com](https://perl6advent.wordpress.com)
- Jonathan Worthington’s presentations with examples
- [http://jnthn.net/articles.shtml](http://jnthn.net/articles.shtml)
Support?

- fine folks of the #perl6 channel on irc.freenode.org
- blogs: http://planeteria.org/perl6/
- from Perl 5: http://perlgeek.de/en/article/5-to-6
- the nitty gritty: http://perlcabal.org/syn/
Questions?
How the Camel is de-cocooning

Elizabeth Mattijsen
YAPC::NA, 23 June 2014
Thank You!
for the White Camel