

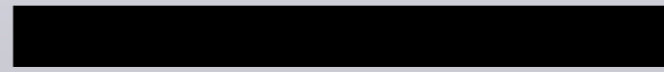
Pugs

an implementation of

Perl 6

Audrey Tang

def



Pugs...

Pugs...

def **Perl 6 Compiler**

Pugs...

def **Perl 6 Compiler**

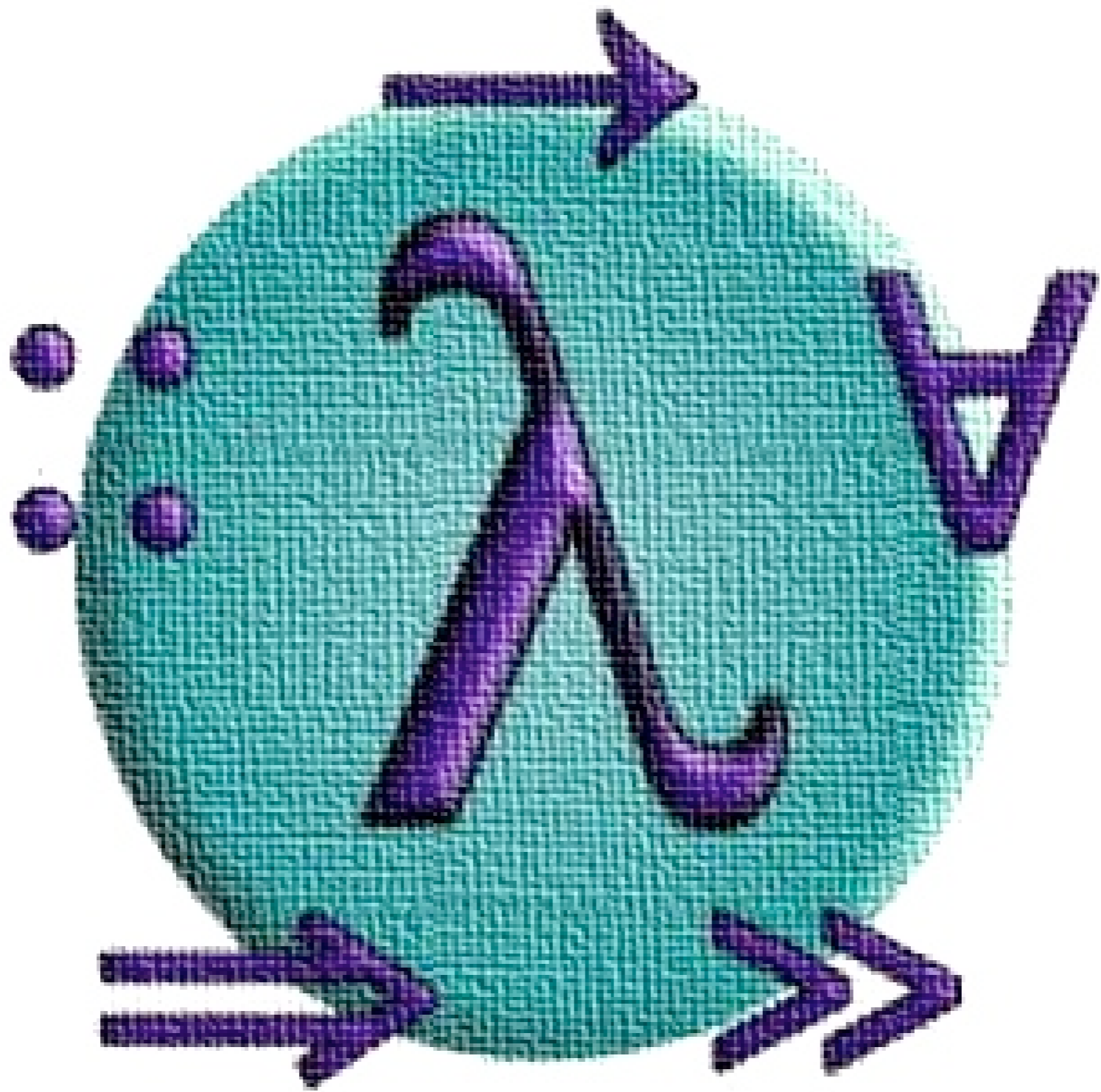
def **Perl 6 Runtime**

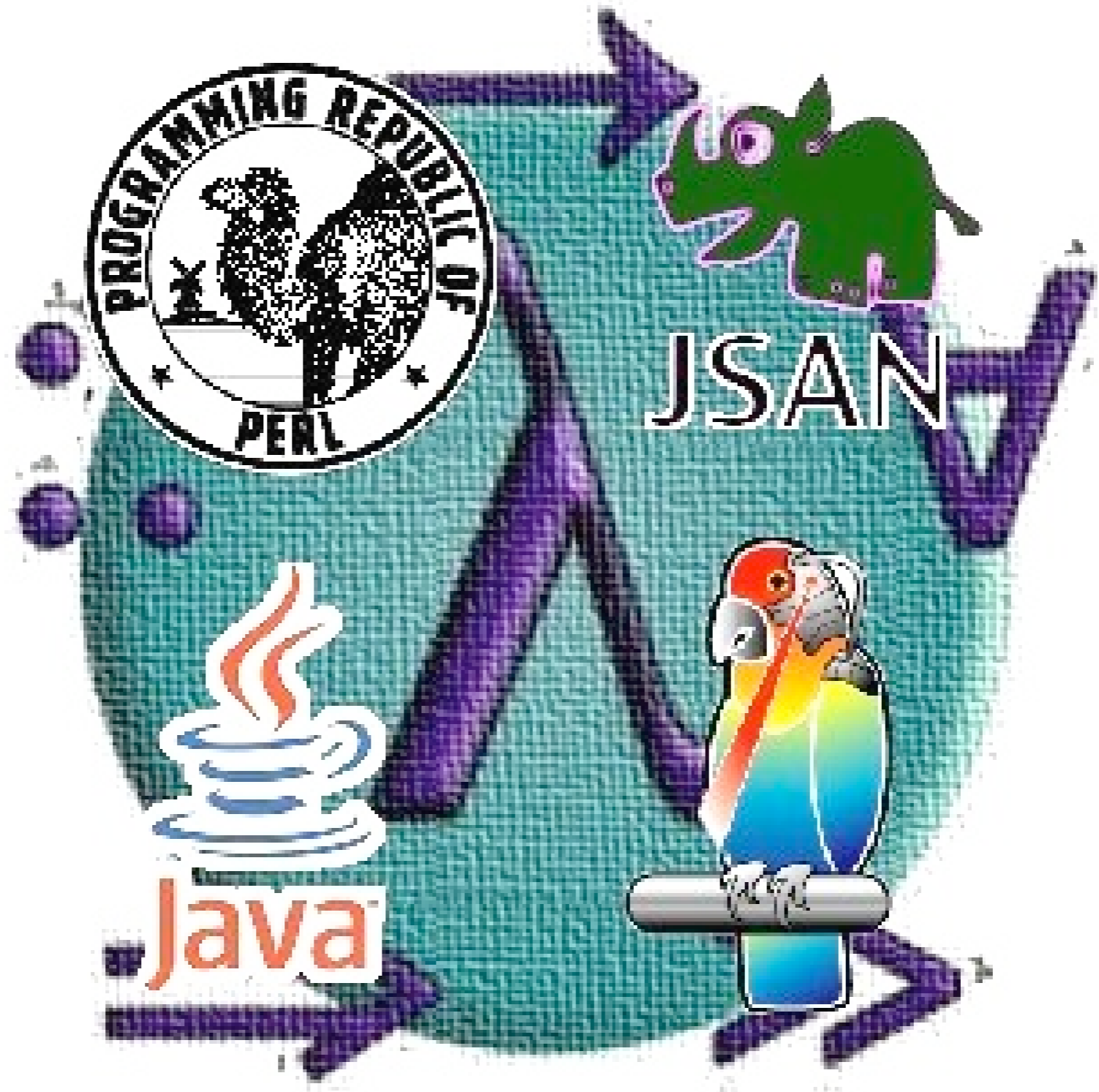
Pugs...

def **Perl 6 Compiler**

def **Perl 6 Runtime**

def **Perl 6 Test Suite**











Perl 6 Code

Perl 6 Code

✓ **120+ Modules**

Perl 6 Code

✓ **120+ Modules**

✓ **160+ Examples**

Perl 6 Code

- ✓ **120+ Modules**
- ✓ **160+ Examples**
- ✓ **18,000+ Unit Tests**

“Official Perl 6”

“Official Perl 6”

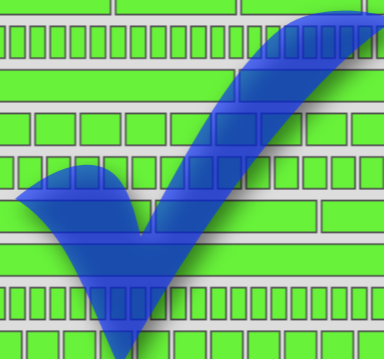
- ✓ Anything that passes the official test suite

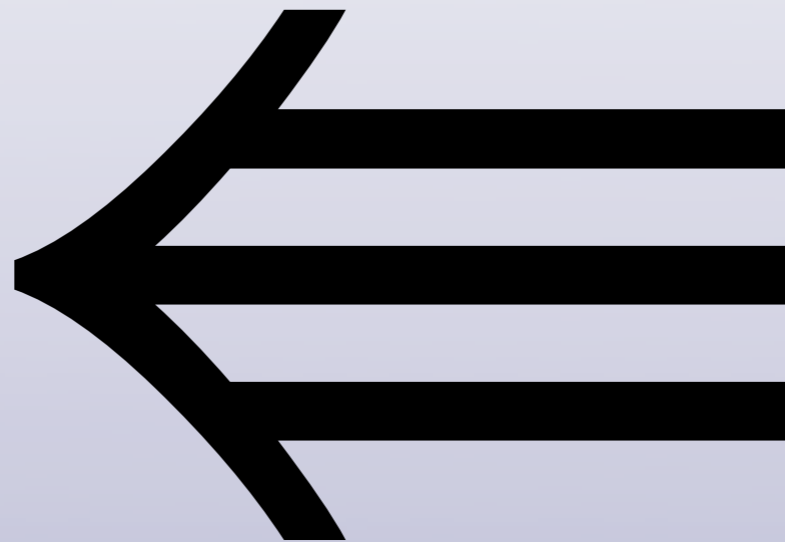
“Official Perl 6”

- ✓ Anything that passes the official test suite
- ✓ Defined by semantics, not by accidents of history

Test \Leftrightarrow Spec

t/builtins/lists/sort.t	OK		100.00%
t/builtins/lists/sum.t	OK		100.00%
t/builtins/lists/topic_in_map.t	OK		100.00%
t/builtins/lists/uniq.t	OK		100.00%
t/builtins/lists/zip.t	OK		100.00%
t/builtins/math/NaN.t	OK		100.00%
t/builtins/math/abs.t	OK		100.00%
t/builtins/math/exp.t	OK		100.00%
t/builtins/math/infinity.t	OK		100.00%
t/builtins/math/int.t	OK		100.00%
t/builtins/math/log.t	OK		100.00%
t/builtins/math/rand.t	OK		100.00%
t/builtins/math/rounders.t	OK		100.00%
t/builtins/math/sign.t	OK		100.00%
t/builtins/math/sqrt.t	OK		100.00%
t/builtins/math/trig.t	OK		100.00%
t/builtins/my.t	OK		100.00%
t/builtins/numify.t	OK		100.00%
t/builtins/perl.t	FAILED		90.91%
t/builtins/perl2.t	FAILED		94.74%
t/builtins/ref.t	OK		100.00%
t/builtins/sprintf_and_as.t	OK		100.00%
t/builtins/strings/append.t	OK		100.00%
t/builtins/strings/capitalize.t	OK		100.00%





Perl, circa 1995

Perl, circa 1995

- `use 5.000;`

Perl, circa 1995

- `use 5.000;`
- `require 'fastcwd.pl';`

Perl, circa 1995

- `use 5.000;`
- `require 'fastcwd.pl';`
- `require 'newgetopt.pl';`

Perl, circa 1995

- `use 5.000;`
- `require 'fastcwd.pl';`
- `require 'newgetopt.pl';`
- `require 'exceptions.pl';`

Perl, circa 2005

Perl, circa 2005

```
use v6-alpha;
```

Perl, circa 2005

```
use v6-alpha;  
use perl5:DBI;
```

Perl, circa 2005

use v6-alpha;

use perl5:DBI;

use perl5:Encode <encode decode>;

Perl, circa 2005

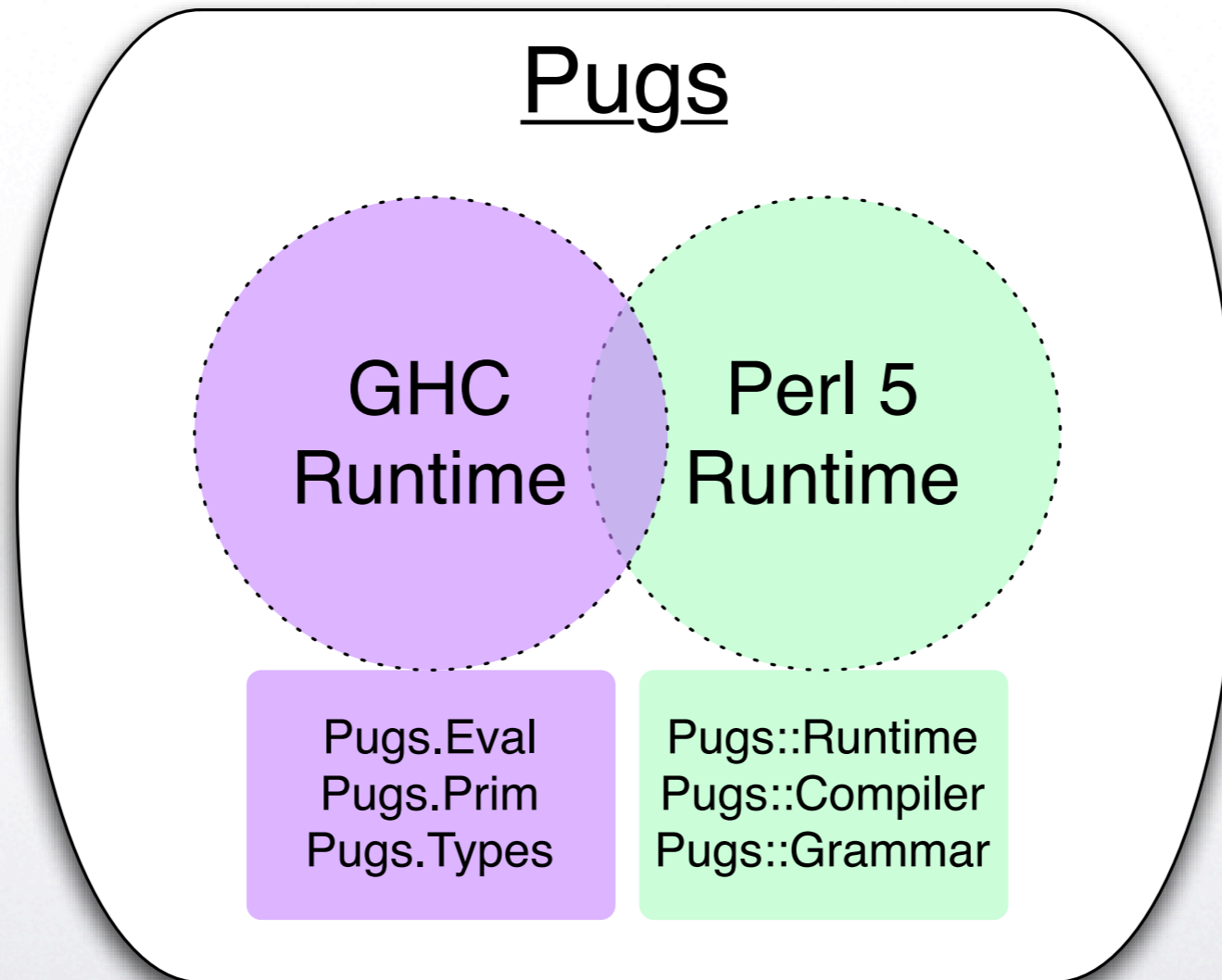
- `use v6-alpha;`
- `use perl5:DBI;`
- `use perl5:Encode <encode decode>;`
- `use perl5:Template;`

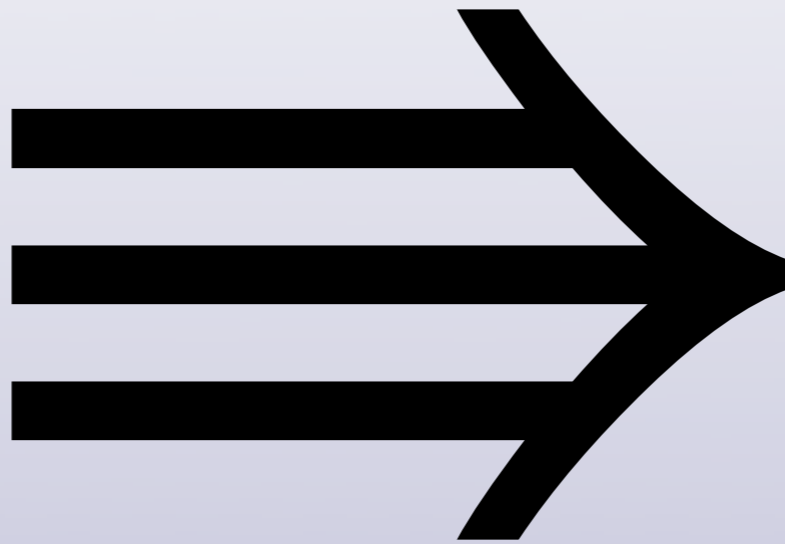
Perl, circa 2005

- `use v6-alpha;`
- `use perl5:DBI;`
- `use perl5:Encode <encode decode>;`
- `use perl5:Template;`

- `# Implementation of "fork"`
`eval "fork()" :lang<perl5>;`

Dual Core





Pugs

Intermediate

Language

Backends

Backends

⇒ Perl 5

Backends

⇒ **Perl 5**

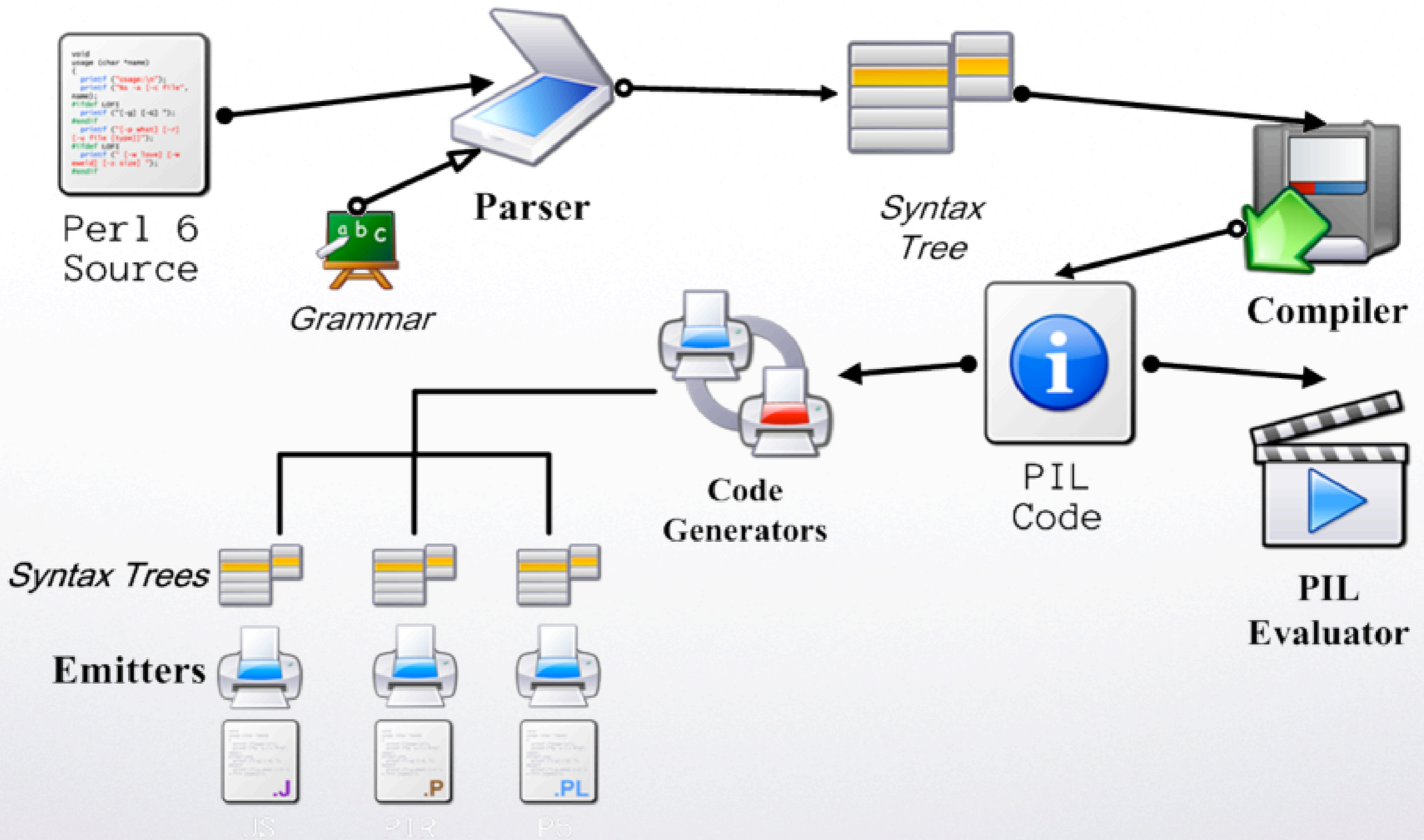
⇒ **Parrot**

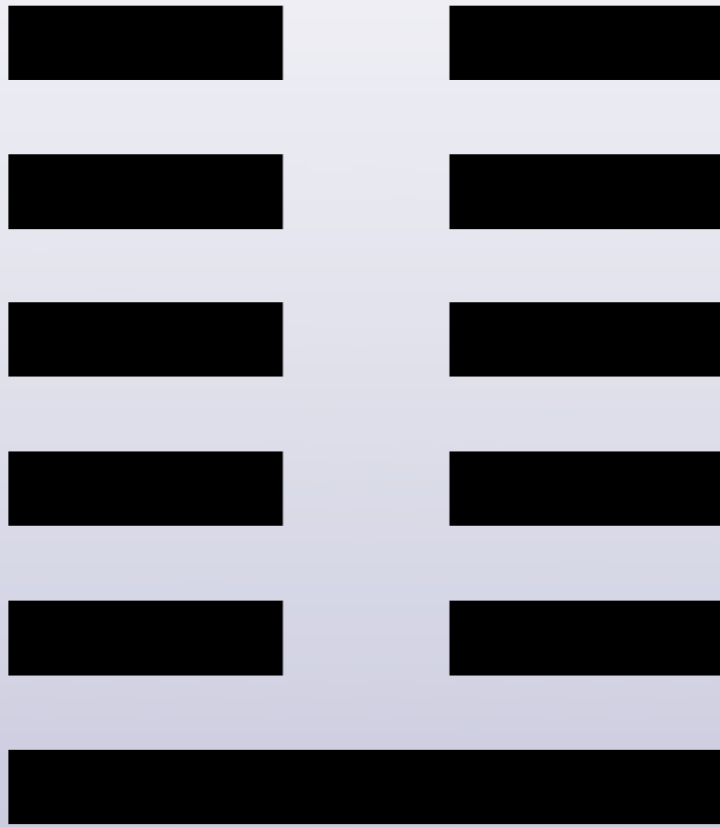
Backends

⇒ **Perl 5**

⇒ **Parrot**

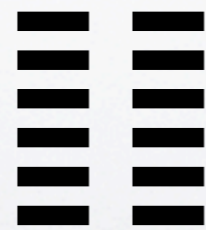
⇒ **JavaScript**





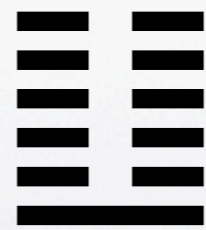
6.0

Primitives



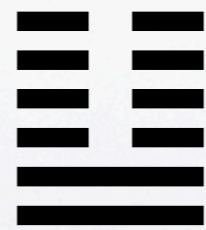
6.2

Functions



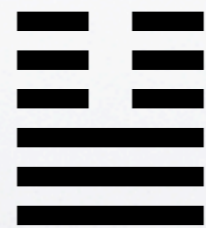
6.28

Objects



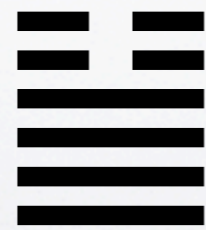
6.283

Grammars



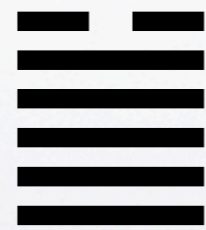
6.2831

Types



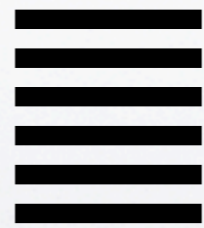
6.28318

Macros



6.283185

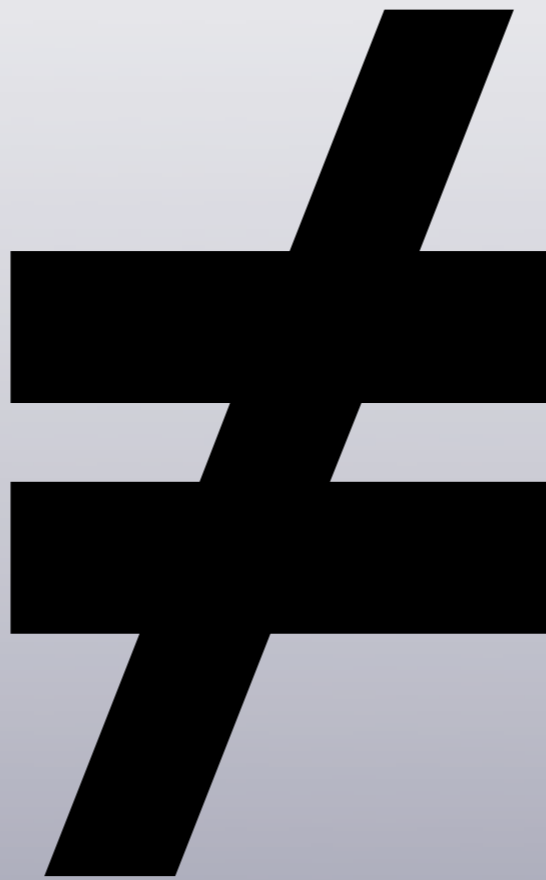
Self Hosting



2π

Perl 6!





**“Frivolous Toy
interpreter”**

(as seen on Slashdot)

**“Frivolous
Toy interpreter”**

~~“Frivolous~~

Toy interpreter”

“Toy interpreter”

“Toy ~~interpreter~~”

“Toy”

— **O** *fun*

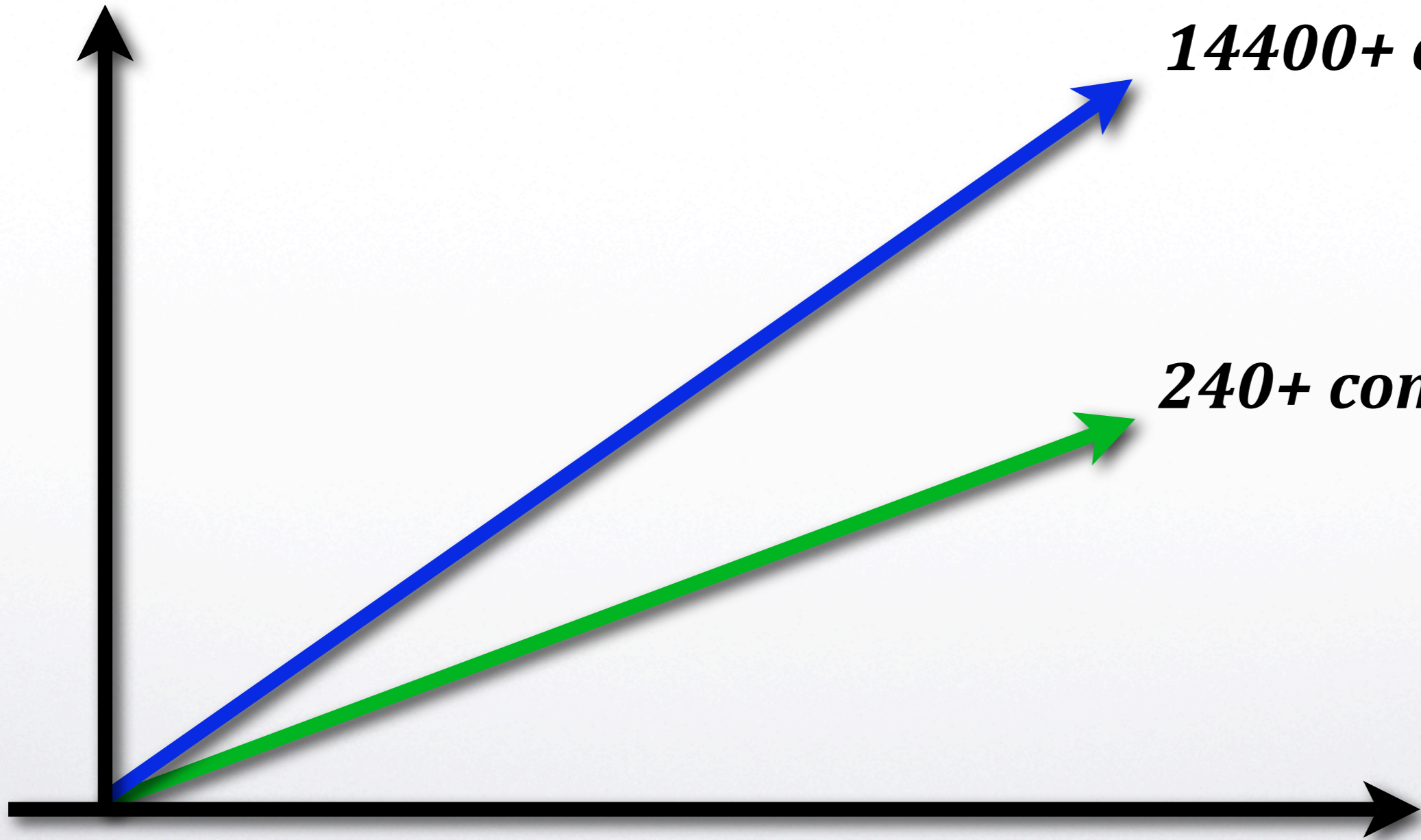


Arrow length

14400+ commits

240+ committers

Time



Test Driven

Test Driven

☺ Bug report »»» Test

Test Driven

☺ Bug report »»» Test

☺ Smoke Server

Test Driven

☺ Bug report »»» Test

☺ Smoke Server

☺ : todo<unspecced>

Anarchistic

Anarchistic

😊 10+ languages

Anarchistic

😊 **10+ languages**

😊 **30+ sub-projects**

Anarchistic

😊 **10+ languages**

😊 **30+ sub-projects**

😊 **Fast feedback loop**

Patches Welcome?

Commits Welcome!

```
16:19 @audreyt A commit bit is in your inbox now...
16:19 @audreyt Don't forget to add your name to the AUTHORS file;
16:19 @audreyt Welcome aboard!
16:19 @audreyt Welcome aboard!
```




irc.freenode.net


#perl6

λ Camels

λ Camels

 **200+ People**

λ Camels

 **200+ People**

 **20+ Regulars**

λ Camels

- 📞 200+ People
- 📞 20+ Regulars
- 📞 TimToady++

svnbot6

```
13:26 < svnbot6 > r10754 | audreyt++ | * reduce-metaop.t - unTODO one surprisingly
13:26 < svnbot6 > r10754 | audreyt++ |   succeeding test involving [=].
13:26 < svnbot6 > r10754 | audreyt++ |   That's all for tonight...
14:35 < svnbot6 > r10755 | audreyt++ | * INSTALL: Bump our parrot requirement to
14:35 < svnbot6 > r10755 | audreyt++ |   0.4.5 for the shiny regex/token/rule
14:35 < svnbot6 > r10755 | audreyt++ |   :ratchet/:sigspace support.
15:50 < svnbot6 > r10756 | fglock++ | Pugs::Grammar::Perl6 - added @{exp}, exp[], s///,
15:50 < svnbot6 > r10756 | fglock++ |   - added modules Pugs::Compiler::Perl6,
15:50 < svnbot6 > r10756 | fglock++ |   Pugs::Emitter::Perl6::Perl5,
15:50 < svnbot6 > r10756 | fglock++ |   Pugs::Runtime::Perl6
15:50 < svnbot6 > r10756 | fglock++ |   - added stub module: v6-pugs
15:50 < svnbot6 > r10756 | fglock++ | Pugs::Compiler::Precedence
15:50 < svnbot6 > r10756 | fglock++ |   - fixed postcircumfix to allow an empty list
15:53 < svnbot6 > r10757 | fglock++ | renamed Pugs-Grammar-Perl6 to Pugs-Compiler-Perl6
15:56 < svnbot6 > r10758 | fglock++ | Pugs-Compiler-Perl6 - fixed test.pl
```

```
12:20 < svnbot6 > r10728 | fglock++ | Pugs-Compiler-Perl6 - fixed test.pl
12:23 < svnbot6 > r10729 | fglock++ | Pugs-Compiler-Perl6 - fixed test.pl
12:28 < svnbot6 > r10730 | fglock++ | Pugs-Compiler-Perl6 - fixed test.pl
```

evalbot6

```
16:25 < audreyt> ?eval [+] 1..100  
16:26 < evalbot_10746> 5050  
16:26 < audreyt> ?eval { $_ ?? $_ * &?BLOCK($_-1) !! 1 }.(10)  
16:26 < evalbot_10746> 3628800
```

```
16:26 < evalbot_10746> 3628800
```

lambdabot

```
16:30 < audreyt> @p1 f h = hGetContents h >>= \x -> return (lines x)
16:30 < lambdabot> f = (lines `fmap`) . hGetContents
16:32 < audreyt> @djinn (a -> b) -> (c -> b) -> Either a c -> b
16:32 < lambdabot> f a b c =
16:32 < lambdabot>   case c of
16:32 < lambdabot>   Left d -> a d
16:30 < lambdabot>   Right e -> b e
```

```
16:30 < lambdabot>   Right e -> b e
16:35 < lambdabot>   Right e -> b e
```

IRC.pugscode.org

perl6 2006-10-18,Wed

[Logs](#) [Channels](#) [Help](#) [Search](#) [←Prev date](#) (Last day)

Who

Alias_	Well, it would be nice from a PR point of view to have it formalised and wrapped up in a candy-shell And might actually finally mean I don't have to listen to the Duke Nukem jokes any more :)
avar	ascii art of the duke in the installer?:)
Alias_	That would be hilarious :)
[particle]	i think duke nukem is under misc/ somewhere in the repo... ;)
Alias_	A camel with a shotgun and strippers
avar	heh
TimToady	I think when people start playing with 6.2.13 they'll realize it's really getting rather solid already.
	Alias_ nods
Alias_	I was just thinking in more symbolic terms
TimToady	and I really don't mind the anonymonks proving themselves to be idiots in front of everyone. well, o
---	buetow joined perl6
TreyHarris	TimToady: which anonymonks are you referring to? not names, just what comprehension results in
TimToady	it seems a little odd to use the word "comprehension" on that particular subset
---	dakrone joined perl6 hexmode joined perl6 weinig joined perl6
TimToady	Parents' Back To School Night &

TimToady Parents' Back To School Night &

weinig joined perl6
hexmode joined perl6

TimToady it seems a little odd to use the word "comprehension" on that particular subset

dakrone joined perl6
hexmode joined perl6
weinig joined perl6

Blog.pugscode.org

Pugs

Implementing Perl 6... and other related technologies.

* October 2006

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

sphere BLOG SEARCH

* Recent Posts

- » [Off to Portland...](#)
- » [Pugs 6.2.13 released!](#)
- » [Run Pugs: a Web terminal for Pugs](#)

2006.10.18

| Off to Portland...

My plane to Portland is taking off in a few hours. So, short recaps:

- fglock++ is hacking relentlessly on the next generation of [Perl6-to-Perl5 emitter](#), incorporating type constraints and autoboxing into the mix.
- lanny++ fixed Perl6::Doc's [Makefile.PL](#) to work with nmake 1.5. This reminds me that we still need to upload it to CPAN separately as a replacement to the horribly outdated [Perl6::Bible](#).
- andara++ continues to tune [runpugs](#) to avoid exhausting the dreaded resource limits. Help welcome, especially for some Ajaxy sugar to make it as sexy as [tryruby](#)...

Some short term post-release plans:

- Drop support for GHC 6.4.1, as soon as ghc-6.6 (with extralibs) makes it to Debian and FreeBSD. The build system is haunted by workarounds for old semi-broken Haskell Cabal

Run.pugscodcode.org

Run Perl 6 now -- in your browser!

This live web terminal runs the latest development snapshot of the [pugs](#) interpreter for [Perl 6](#). For more details, read the [info page](#).

Interactive Pugs Session

```
(P)erl6
(U)ser's
(G)olfing
(S)ystem

Version: 6.2.13 (r14415)
Copyright 2005-2006, The Pugs Contributors

-----
Web: http://pugscodcode.org/      Email: perl6-compiler@perl.org

Welcome to Pugs -- Perl6 User's Golfing System
Type :h for help.

Loading Prelude... done.
pugs>
```


Smoke.pugscodde.org

repository snapshot / linux

Pugs 6.2.13 r14410	18 Oct 2006 03:51 Wed	233.02 min	100.00 % ok	17954: 17954,	0, 4638,	747,	0 » » SYN
Pugs 6.2.13 r14403	17 Oct 2006 15:26 Tue	30.08 min	100.00 % ok	17954: 17954,	0, 4638,	747,	0 » » SYN
Pugs 6.2.13 r14402	17 Oct 2006 14:44 Tue	34.87 min	99.69 % ok	17956: 17900,	56, 1182,	6956,	0 » » SYN
Pugs 6.2.13 r14383	17 Oct 2006 05:11 Tue	29.93 min	100.00 % ok	17954: 17954,	0, 4638,	747,	0 » » SYN
Pugs 6.2.13 r14375	17 Oct 2006 00:38 Tue	28.28 min	100.00 % ok	17954: 17954,	0, 4638,	747,	0 » » SYN
Pugs 6.2.12 r14350	15 Oct 2006 06:36 Sun	29.82 min	100.00 % ok	17954: 17954,	0, 4638,	747,	0 » » SYN
Pugs 6.2.12 r14342	15 Oct 2006 03:27 Sun	31.53 min	100.00 % ok	17954: 17954,	0, 4639,	747,	0 » » SYN
Pugs 6.2.12 r14339	14 Oct 2006 20:00 Sat	32.58 min	100.00 % ok	17954: 17954,	0, 4639,	747,	0 » » SYN
Pugs 6.2.12 r14334	14 Oct 2006 10:00 Sat	51.47 min	99.99 % ok	17954: 17952,	2, 4644,	747,	6 » » SYN
Pugs 6.2.12 r14331	13 Oct 2006 19:56 Fri	29.97 min	99.99 % ok	17954: 17953,	1, 4644,	747,	6 » » SYN

repository snapshot / MSWin32

Pugs 6.2.13 r14413	18 Oct 2006 13:36 Wed	58.77 min	99.99 % ok	17918: 17917,	1, 4637,	782,	0 » » SYN
Pugs 6.2.13 r14413	18 Oct 2006 07:57 Wed	31.05 min	99.99 % ok	17954: 17953,	1, 4638,	782,	0 » » SYN
Pugs 6.2.13 r14413	18 Oct 2006 04:42 Wed	17.83 min	99.99 % ok	17954: 17953,	1, 4638,	782,	0 » » SYN
Pugs 6.2.13 r14410	17 Oct 2006 21:40 Tue	17.78 min	99.99 % ok	17954: 17952,	2, 4638,	782,	0 » » SYN

Spec.pugscodex.org

Whitespace and Comments

- Single-line comments work as in Perl 5, starting with a # character and ending at the subsequent newline. They count as whitespace equivalent to newline for purposes of separation. Unlike in Perl 5, # may not be used as the delimiter in quoting constructs.

- Show the snippet from t/syntax/comments.t (line 117 ~ line 128 — 4 ✓, 0 ✕) -

- Multiline comments are provided by extending the syntax of POD to nest `=begin comment/=end comment` correctly without the need for `=cut`. The format name does not have to be `comment` -- any unrecognized format name will do to make it a comment. (However, bare `=begin` and `=end` probably aren't good enough, because all comments in them will show up in the formatted output.)

- Show the snippet from t/syntax/comments.t (line 129 ~ line 162 — 2 ✓, 0 ✕) -

We have single paragraph comments with `=for comment` as well. That lets `=for` keep its meaning as the equivalent of a `=begin` and `=end` combined. As with `=begin` and `=end`, a comment started in code reverts to code afterwards.

- Show the snippet from t/syntax/comments.t (line 163 ~ line 185 — 2 ✓, 0 ✕) -

Since there is a newline before the first `=`, the POD form of comment counts as whitespace equivalent to a newline.

equivalent to a newline

since there is a newline before the first = the POD form of comment counts as whitespace

from the snippet t/syntax/comments.t (line 163 ~ line 185 — 2 ✓, 0 ✕)

Mailing Lists

Mailing Lists

 **perl6-users**

Mailing Lists

 **perl6-users**

 **perl6-language**

Mailing Lists

 **perl6-users**

 **perl6-language**

 **perl6-compiler**

Repositories

Repositories



<http://svn.openfoundry.org/pugs/>

Repositories



<http://svn.openfoundry.org/pugs/>



<http://perlcabal.org/~audreyt/darcs/pugs/>

Repositories



<http://svn.openfoundry.org/pugs/>

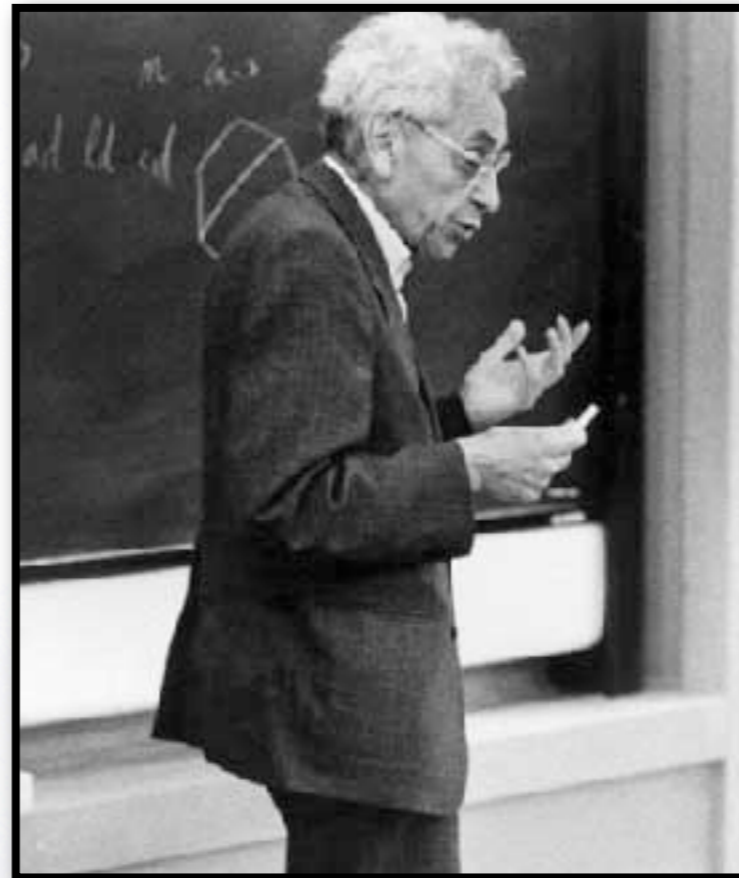


<http://perlcabal.org/~audreyt/darcs/pugs/>

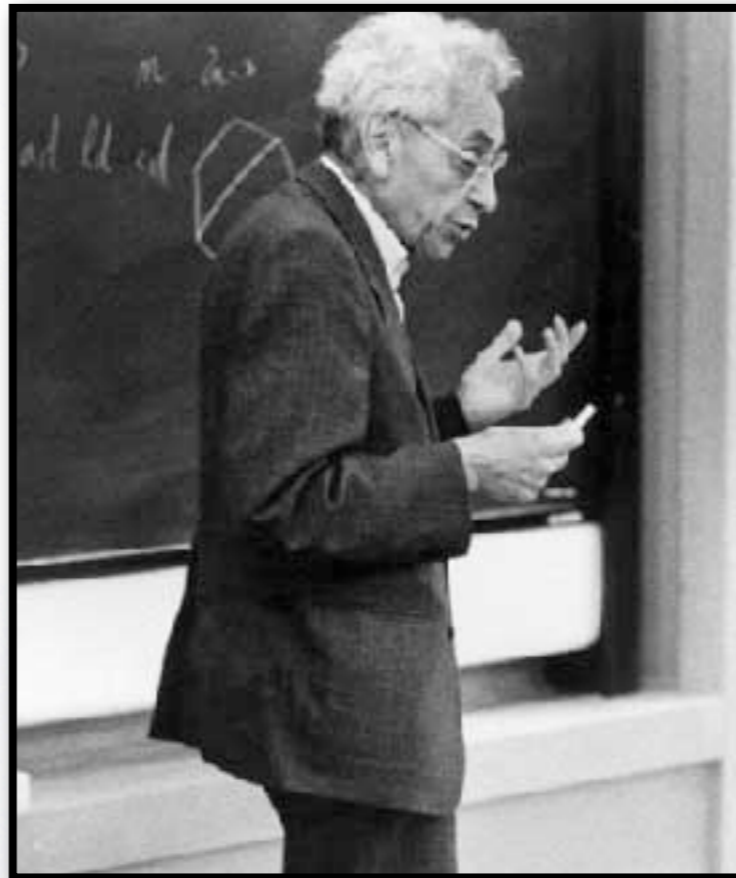




Erdösing



Erdösing



2006..*

Hackathons

Hackathons

 Taipei

Hackathons

 **Taipei**

 **Vienna**

Hackathons



Taipei



Vienna



Toronto

Hackathons



Taipei



Vienna



Toronto



Amsterdam

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel

Hackathons



Taipei



Vienna²



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel

Hackathons



Taipei



Vienna²



Vienna



Tokyo



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel



Vienna²



Tokyo



Redmond

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel



Vienna²



Tokyo



Redmond



Chicago

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel



Vienna²



Tokyo



Redmond



Chicago



Boston

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel



Vienna²



Tokyo



Redmond



Chicago



Boston



Portland

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel



Vienna²



Tokyo



Redmond



Chicago



Boston



Portland



Sao Paulo

Hackathons



Taipei



Vienna



Toronto



Amsterdam



Echt



Lismore



Mt. Arbel



Vienna²



Tokyo



Redmond



Chicago



Boston



Portland



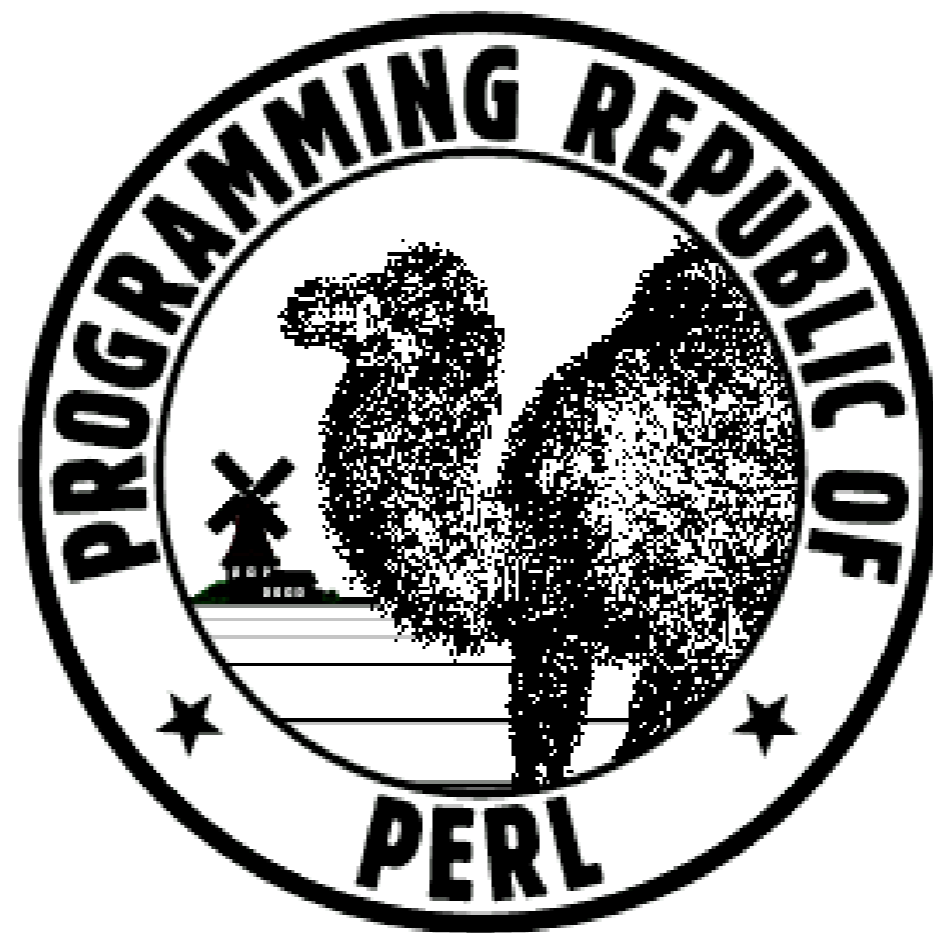
Sao Paulo



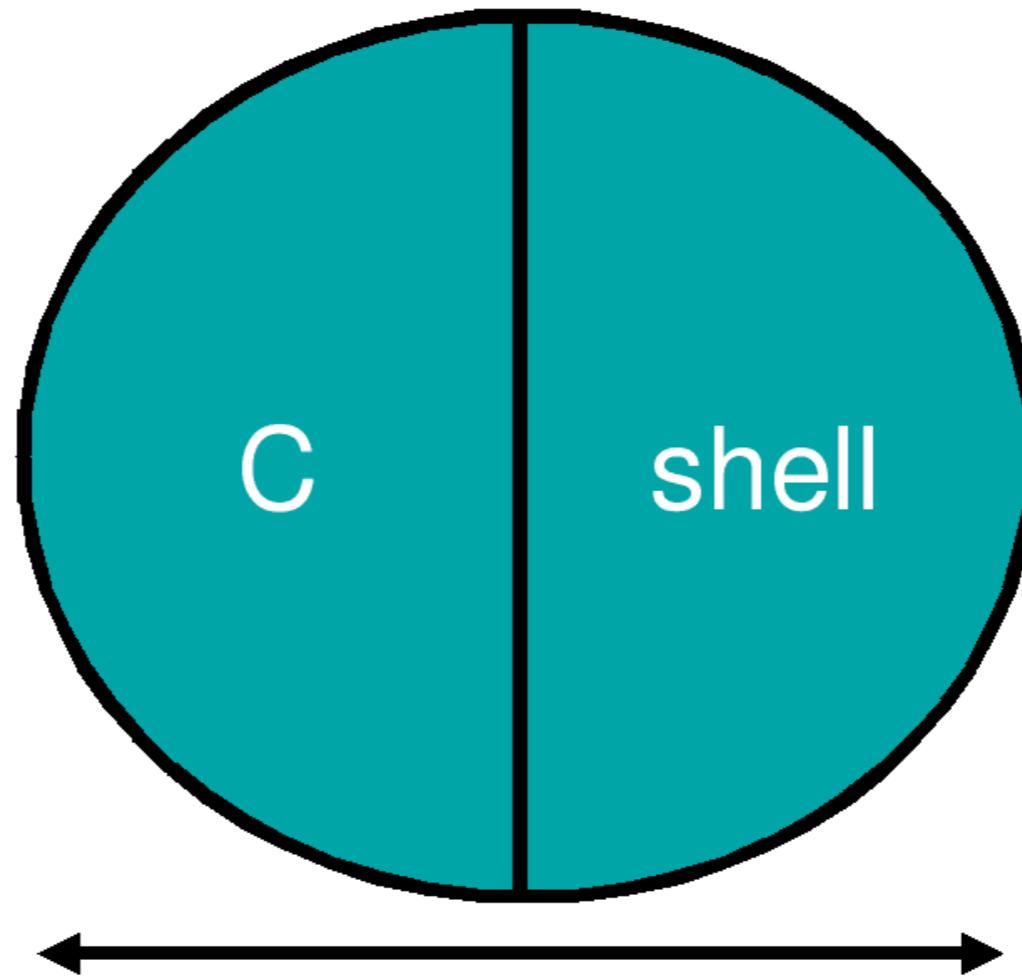
...and more!

P

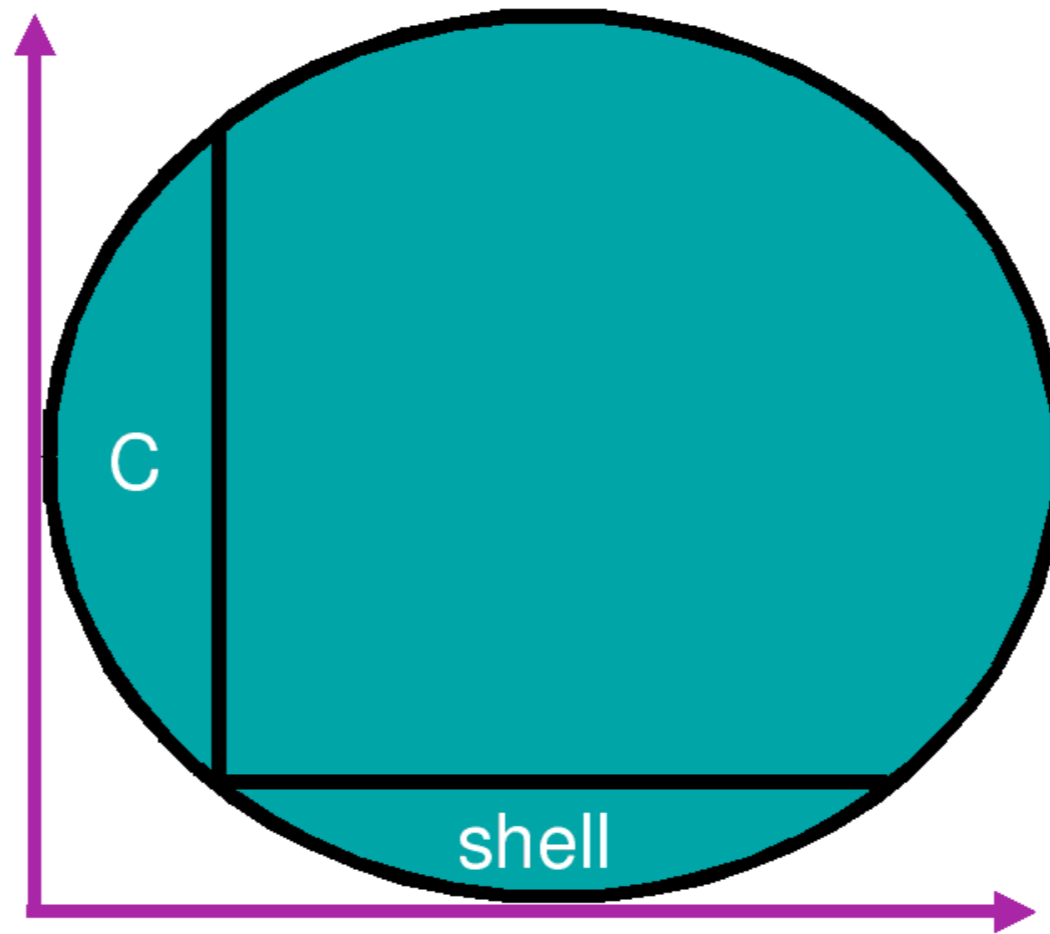
Practical



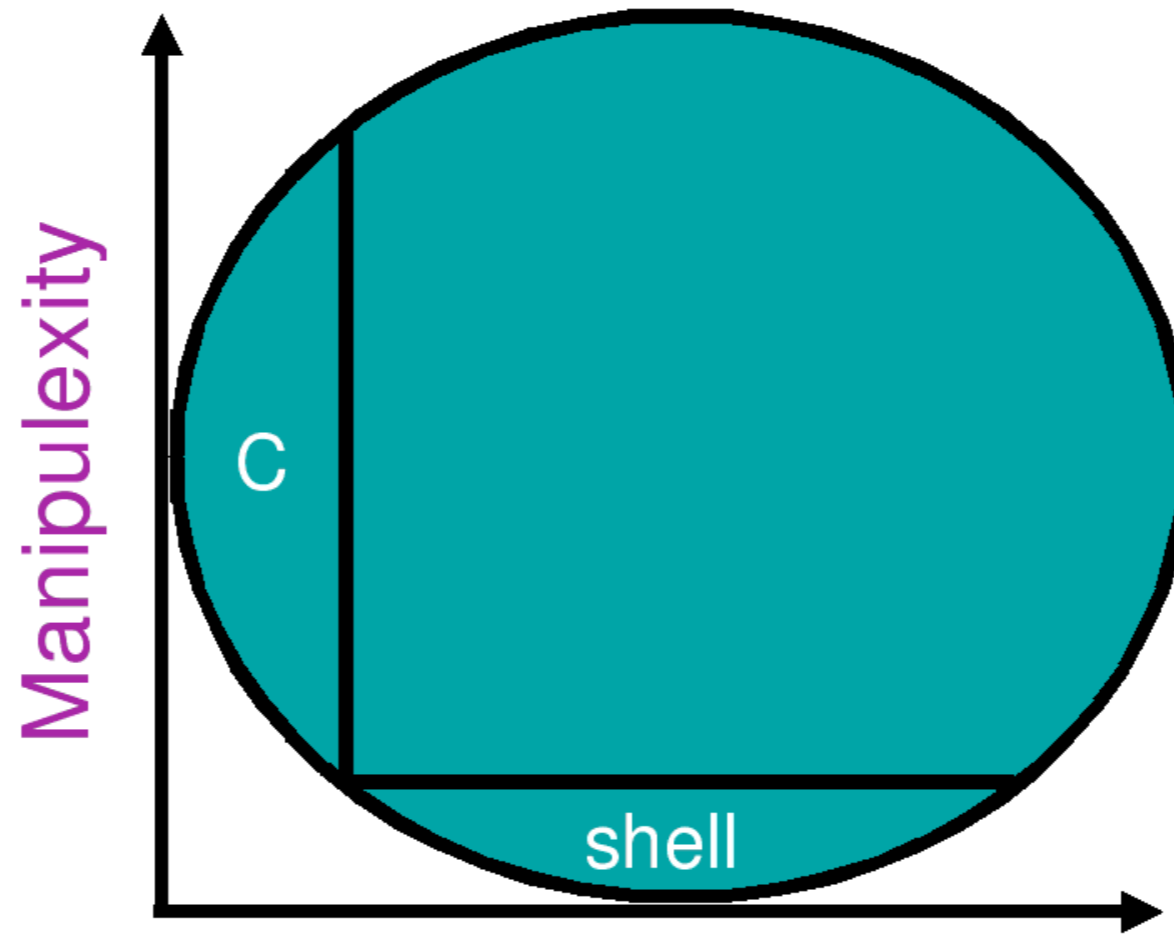
The World of Unix (1987)



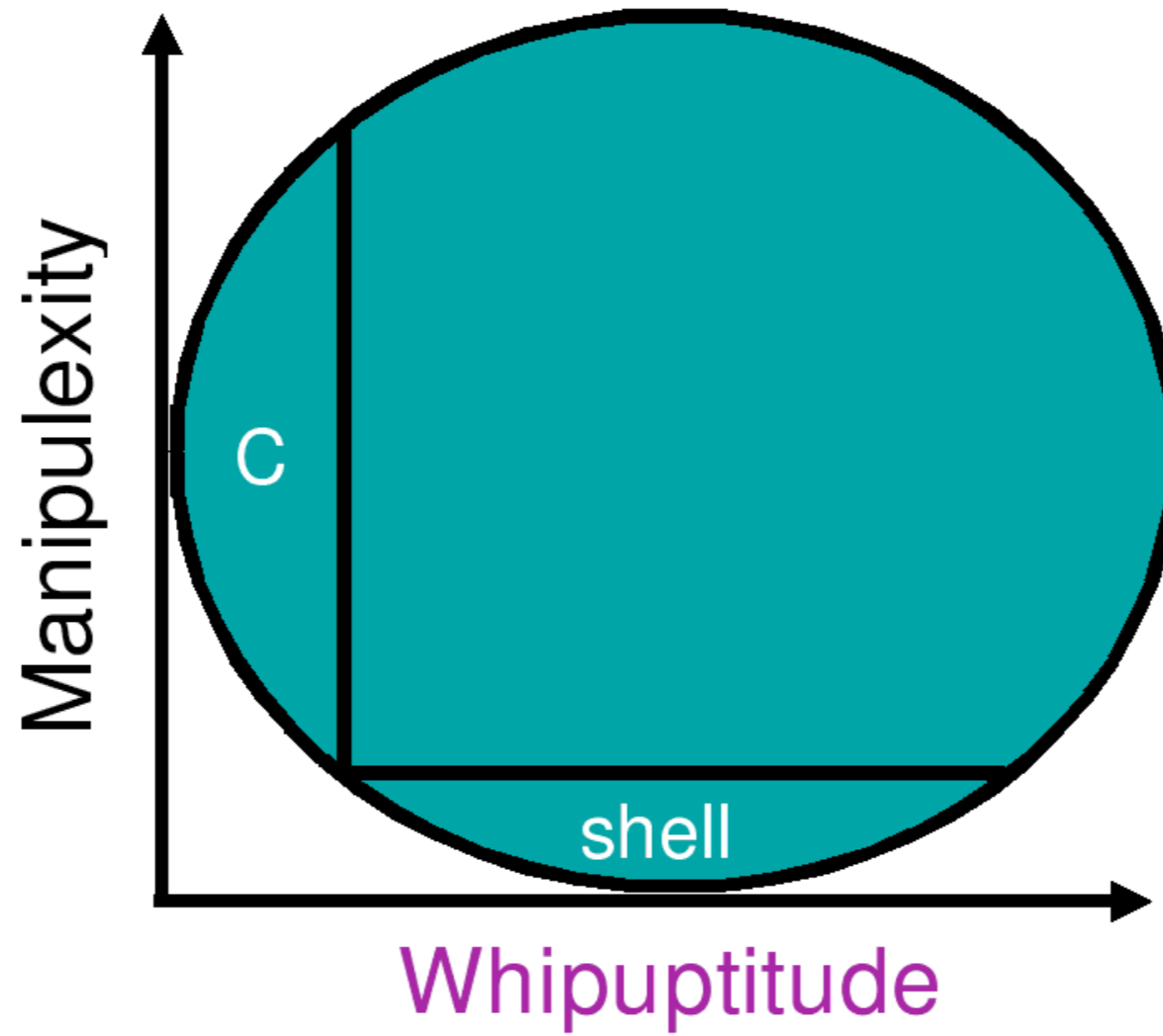
A new dimension



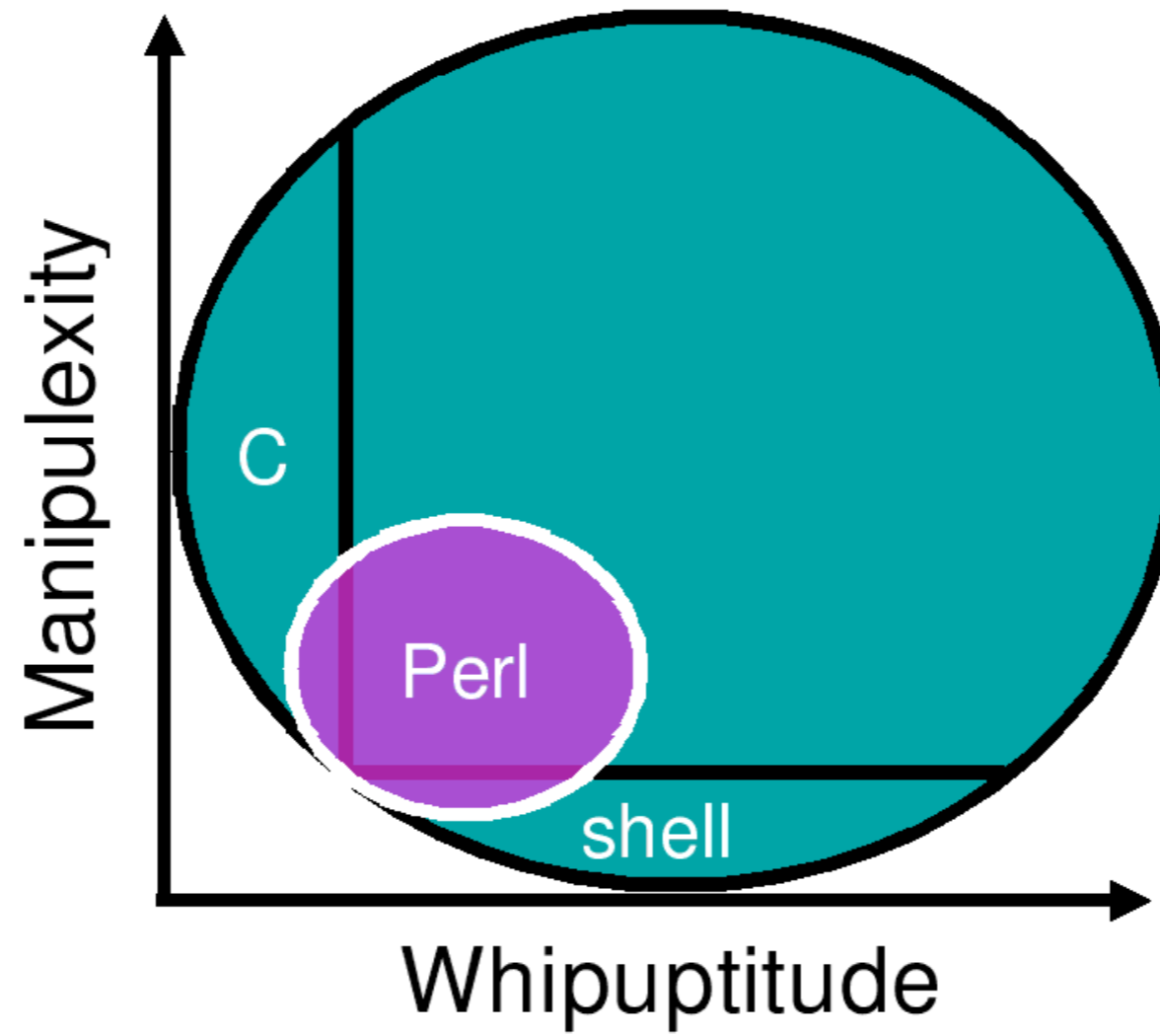
C is good at...



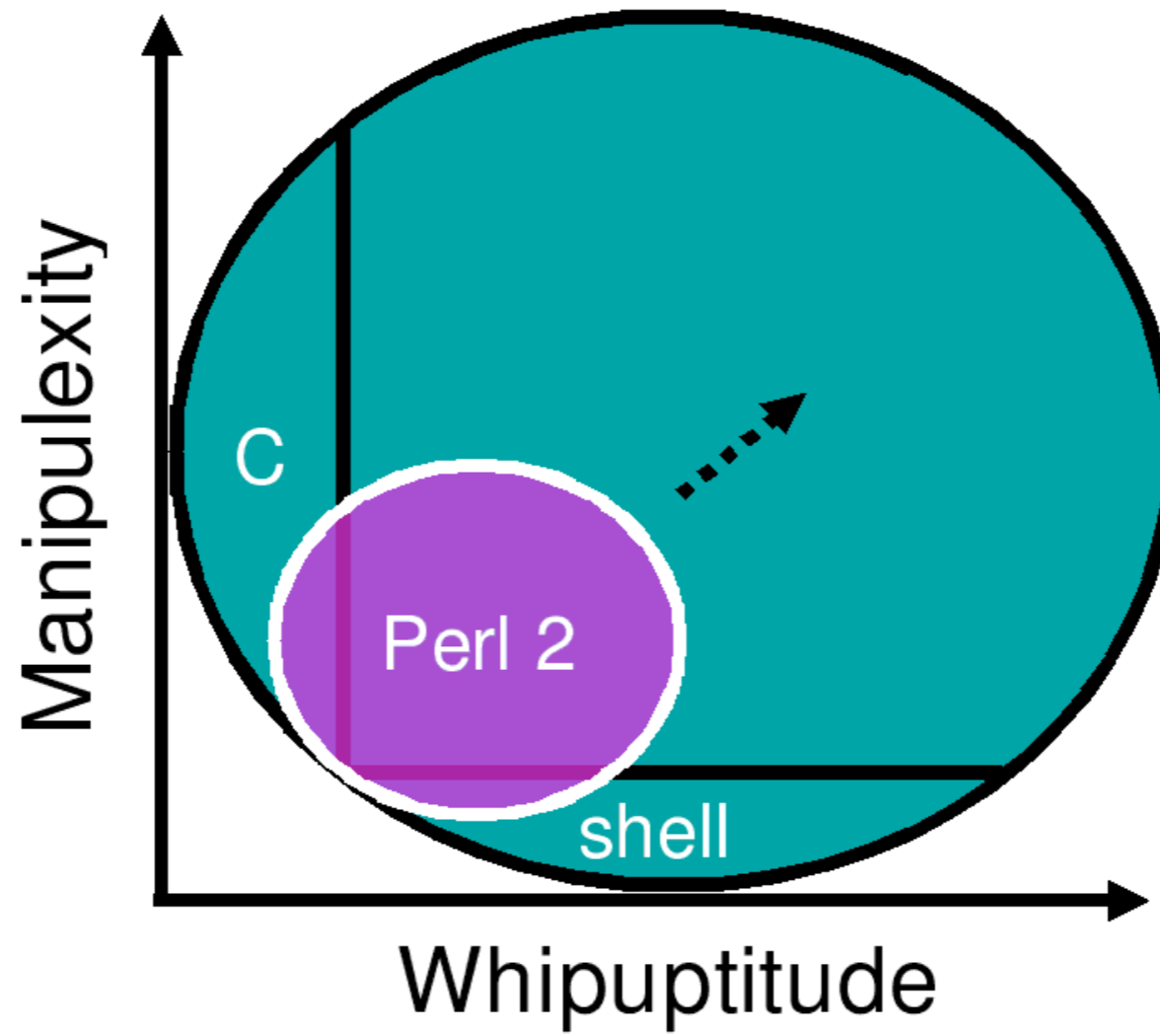
While shell is good at...



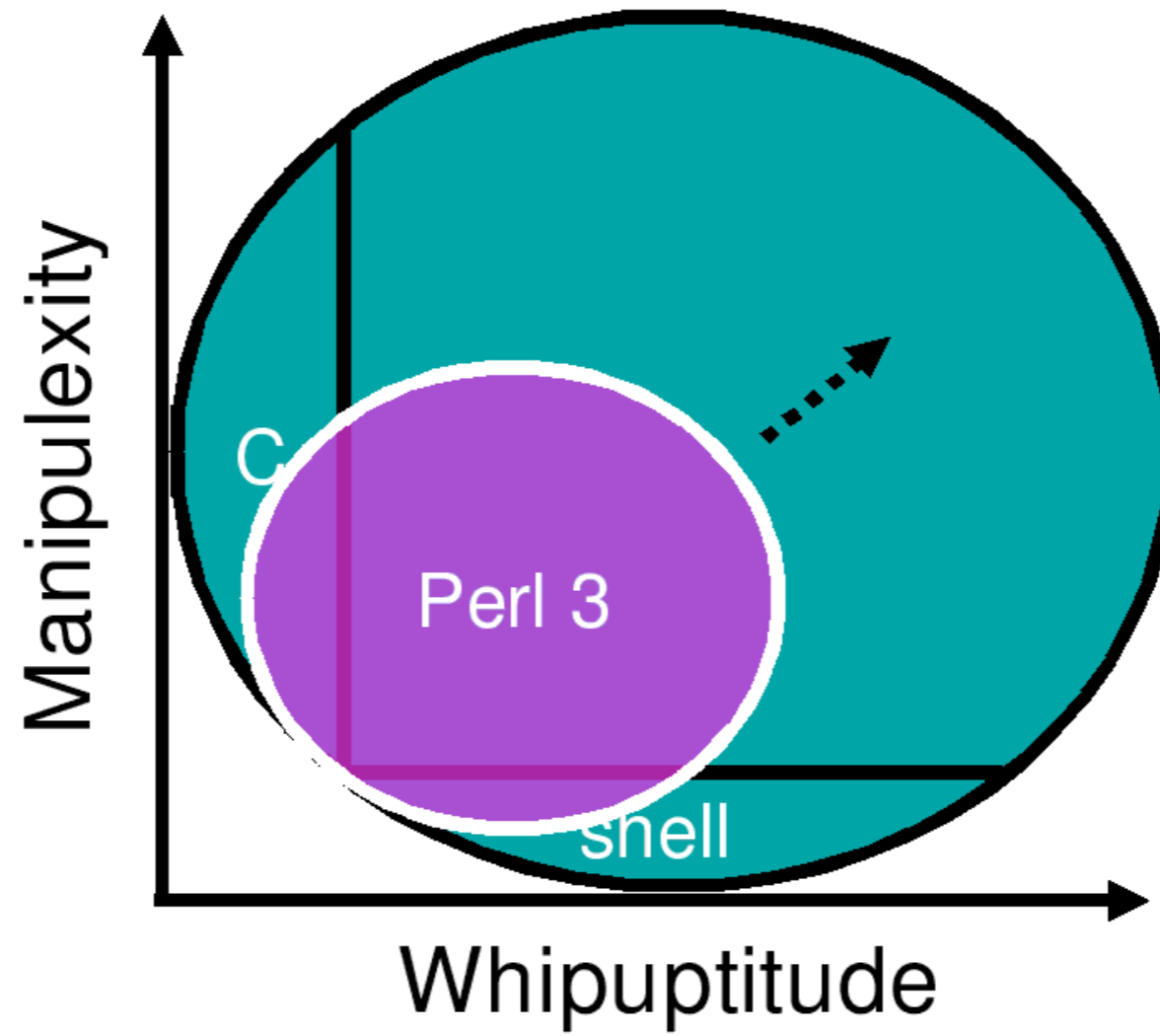
The hatching of Perl



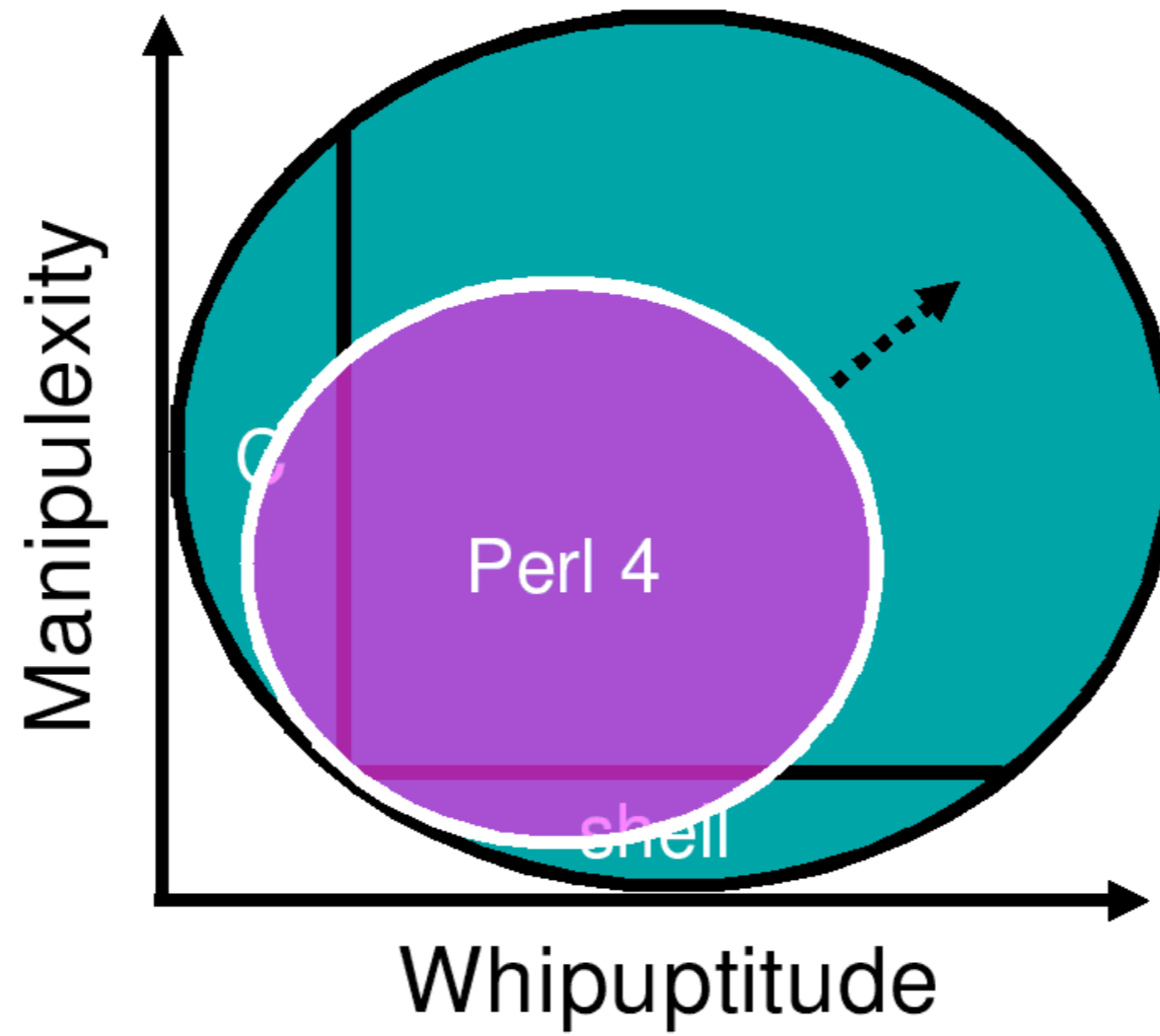
Designed to evolve...



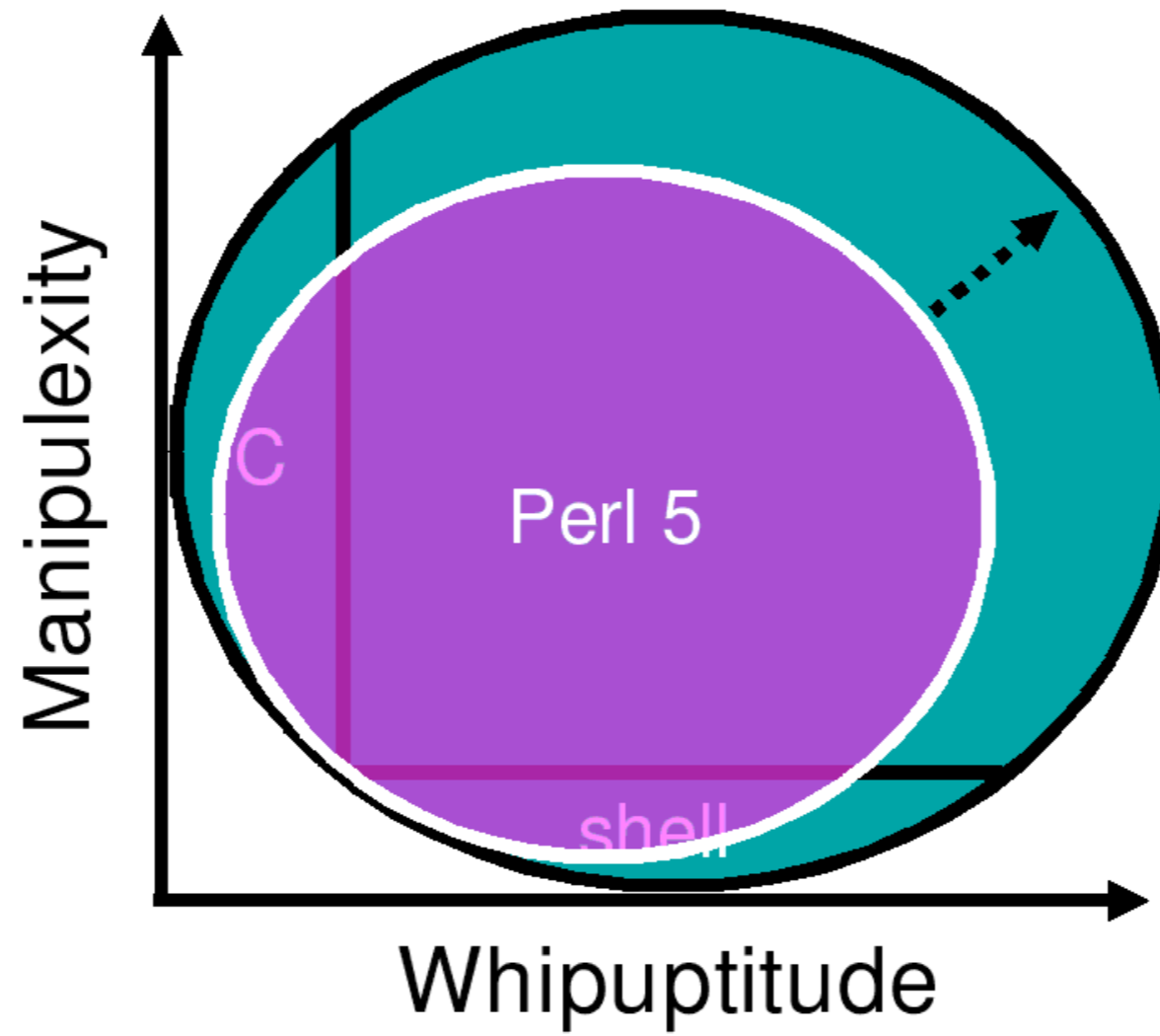
And evolve...



And evolve...and evolve...



All your programs are belong to us...





Abstractions

↳ *Sexy*

Closures

```
sub make_counter {  
  my $start = shift;  
  return sub { ++$start };  
}
```

```
my $from_ten = make_counter(10);  
my $from_three = make_counter(3);
```

```
print $from_three->(); # 4  
print $from_three->(); # 5  
print $from_ten->();   # 11
```

Tie

```
use Tie::Google;
tie my %search => 'Tie::Google';

for (@{ $search{'Perl Pugs'} }) {
    print "* $_->{title} - $_->{URL}\n";
}
```

Abstractions⁺⁺

Abstractions⁺⁺

∇ bless()

Abstractions++

▽ `bless()`

▽ IO Layers

Abstractions++

▽ `bless()`

▽ `IO Layers`

▽ `BEGIN {...}`



Shorthands

↳ Natural

DeCSS

```
s '$/= \2048;while(<>){G=29;R=142;if((@a=unqT="C*",_)[20]&48){D=89;_=unqb24,qT,@
b=map{ord qb8,unqb8,qT,_^$a[--D]}@INC;s/...$/1$&/;Q=unqV,qb25,_;H=73;O=$b[4]<<9
|256|$b[3];Q=Q>>8^(P=(E=255)&(Q>>12^Q>>4^Q/8^Q))<<17,O=O>>8^(E&(F=(S=O>>14&7^0)
^S*8^S<<6))<<9,_=(map{U=%16orE^=R^=110&(S=(unqT,"\xb\ntd\xbz\x14d")[_/16%8]);E
^=(72,@z=(64,72,G^=12*(U-2?0:S&17)),H^=%64?12:0,@z)[_%8]}(16..271))[_]^((D>>=8
)+=P+(~F&E))for@a[128..$#a]}print+qT,@a}';s/[D-HO-U_]/\$$&/g;s/q/pack+/g;eval
```

Shorthands⁺⁺

Shorthands⁺⁺

 **Regex**

Shorthands⁺⁺

📄 **Regex**

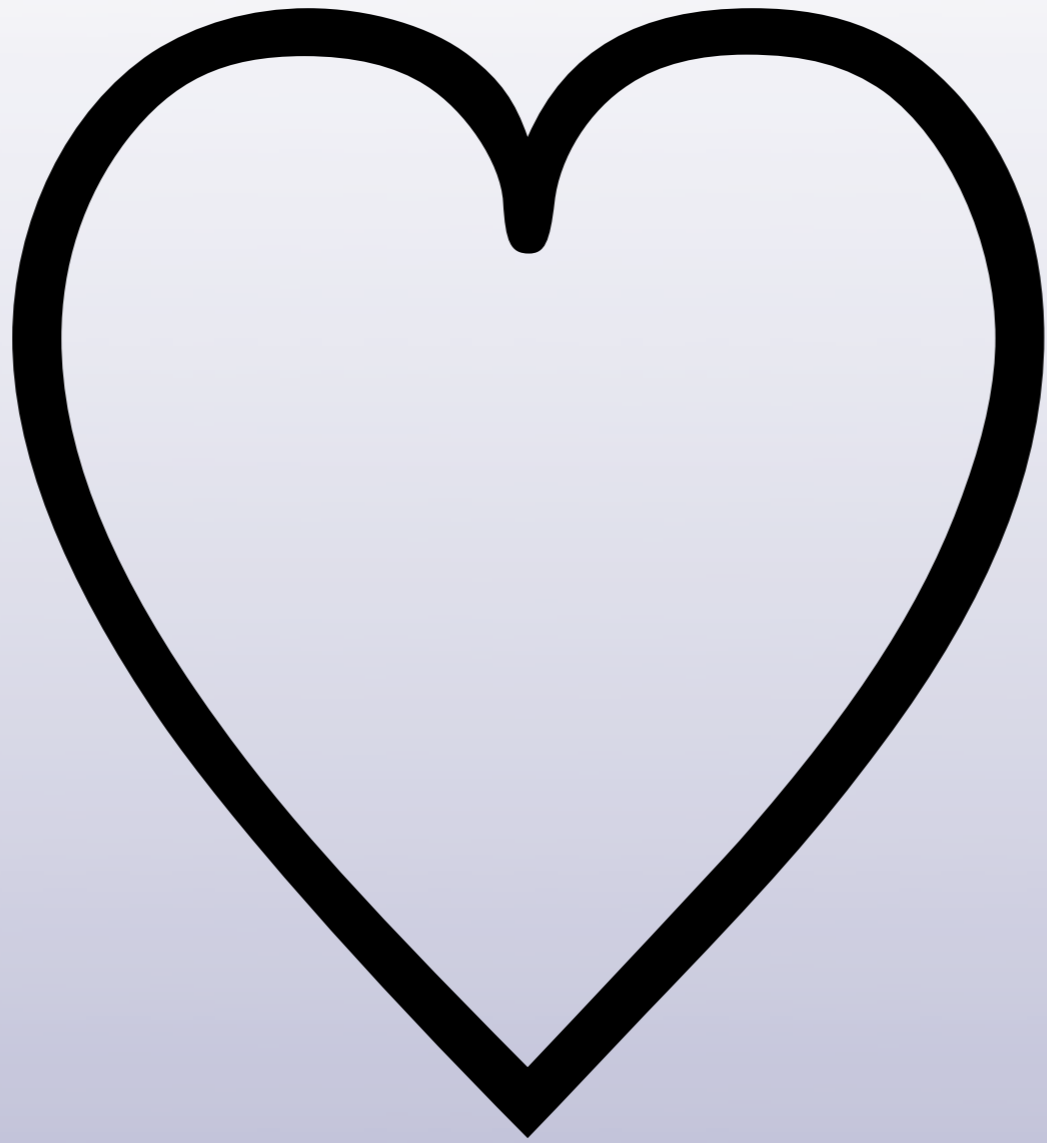
📄 **Context**

Shorthands⁺⁺

☞ **Regex**

☞ **Context**

☞ **Topical \$ _**

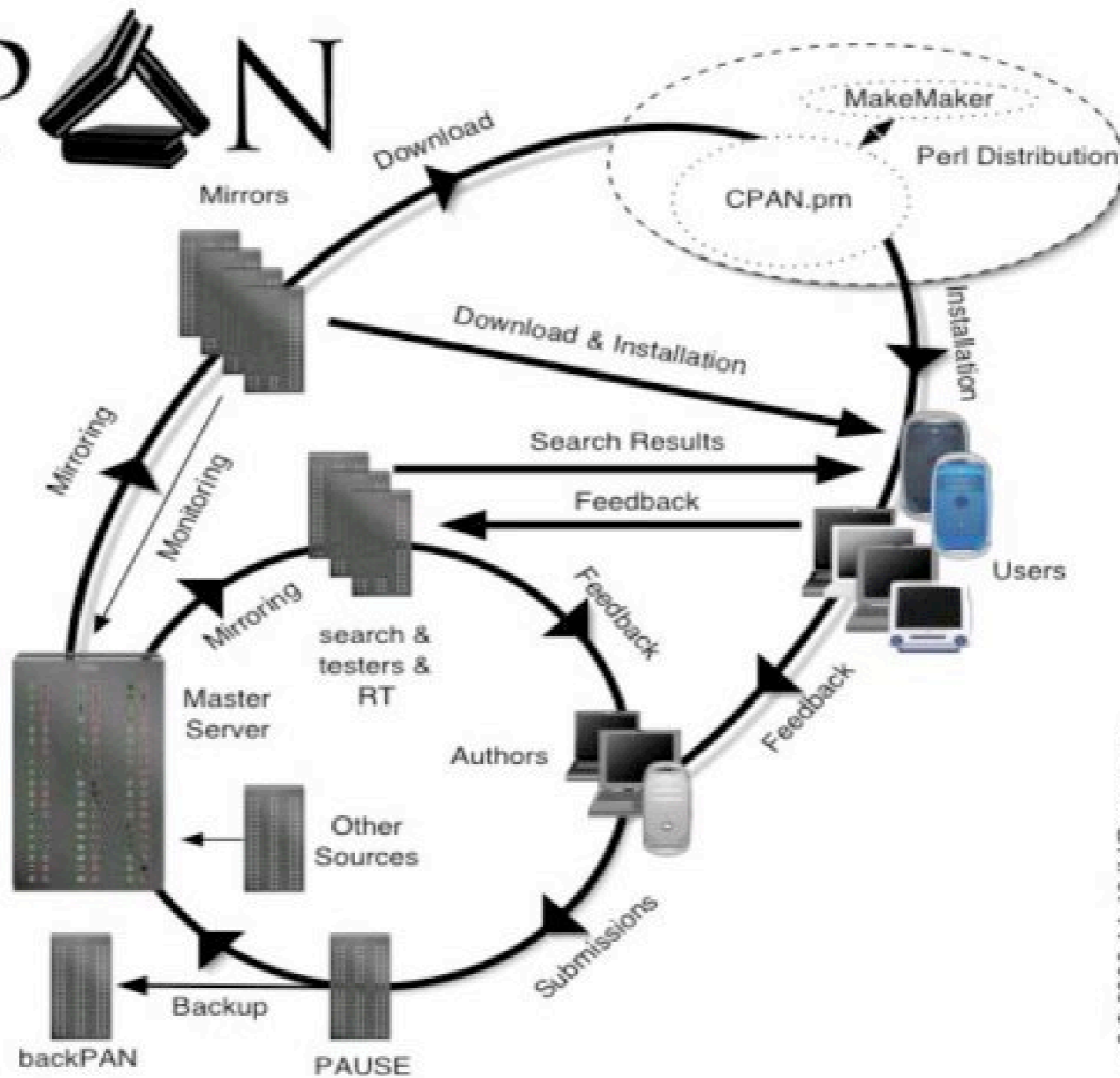


Best coding

↪ No coding

CP  N

CPAN



v0.2 2002-11-18 jhi@cpan.org

CPAN

CPAN

♥ **11 years (this Thursday)!**

CPAN

♡ **11 years (this Thursday)!**

♡ **5000+ authors**

CPAN

- ♡ **11 years (this Thursday)!**
- ♡ **5000+ authors**
- ♡ **10000+ modules**

Services

Services

 **Package Management**

Services

 **Package Management**

 **Rating & Discussion**

Services

- ♡ **Package Management**
- ♡ **Rating & Discussion**
- ♡ **Smoke Testing**

Services

- ♡ **Package Management**
- ♡ **Rating & Discussion**
- ♡ **Smoke Testing**
- ♡ **Issue Tracking**

Vocabulary

≇ Syntax

**“The Best thing
happened to Perl”**

But...

Perl 5

is not the best thing

for **CPAN**



Syntax

Redundancy



use v5;

```
use v5;  
sub render {
```

```
use v5;  
sub render {  
    my $self = shift;
```

```
use v5;  
sub render {  
  my $self = shift;  
  my %opts = (x => 1, y => 1, z => 0, %{$_[0]});
```

```
use v5;  
sub render {  
  my $self = shift;  
  my %opts = (x => 1, y => 1, z => 0, %{$_[0]});  
  for my $item ( $self->filter(@{ $self->{_items} }) ) {
```

```
use v5;
sub render {
  my $self = shift;
  my %opts = (x => 1, y => 1, z => 0, %{$_[0]});
  for my $item ( $self->filter(@{ $self->{_items} }) ) {
    print $item->draw({
      x => $opts{x},
      y => $opts{y},
      z => $opts{z},
    }), "\n";
  }
}
```



```

use v5;
sub render {
  my $self = shift;
  my %opts = (x => 1, y => 1, z => 0, %{$_[0]});
  for my $item ( $self->filter(@{ $self->{_items} }) ) {
    print $item->draw({
      x => $opts{x},
      y => $opts{y},
      z => $opts{z},
    }), "\n";
  }
}

```



```
use v6-alpha;
```

```
use v6-alpha;  
method render ($x = 1, $y = 1, $z = 0) {
```

```
use v6-alpha;  
method render ($x = 1, $y = 1, $z = 0) {  
    for @.filter(@.items) {
```

```
use v6-alpha;
method render ($x = 1, $y = 1, $z = 0) {
    for @.filter(@.items) {
        say .draw(:$x, :$y, :$z);
    }
}
```

Jenga Internals





Bug-for-bug compatibility



Best Practice
takes discipline

Standards and Styles for Developing Maintainable Code



Perl Best Practices

O'REILLY®

Damian Conway

Ο.ΡΕΙΛΛΥ.

ΔΑΜΙΑΝ ΚΟΝΒΕΥ.

Best Practice
should be **Natural!**



2000

RFCs

2001

Parrot

2002

Apocalypse

2003 Ponie

(late, as in the late Arthur Dent)

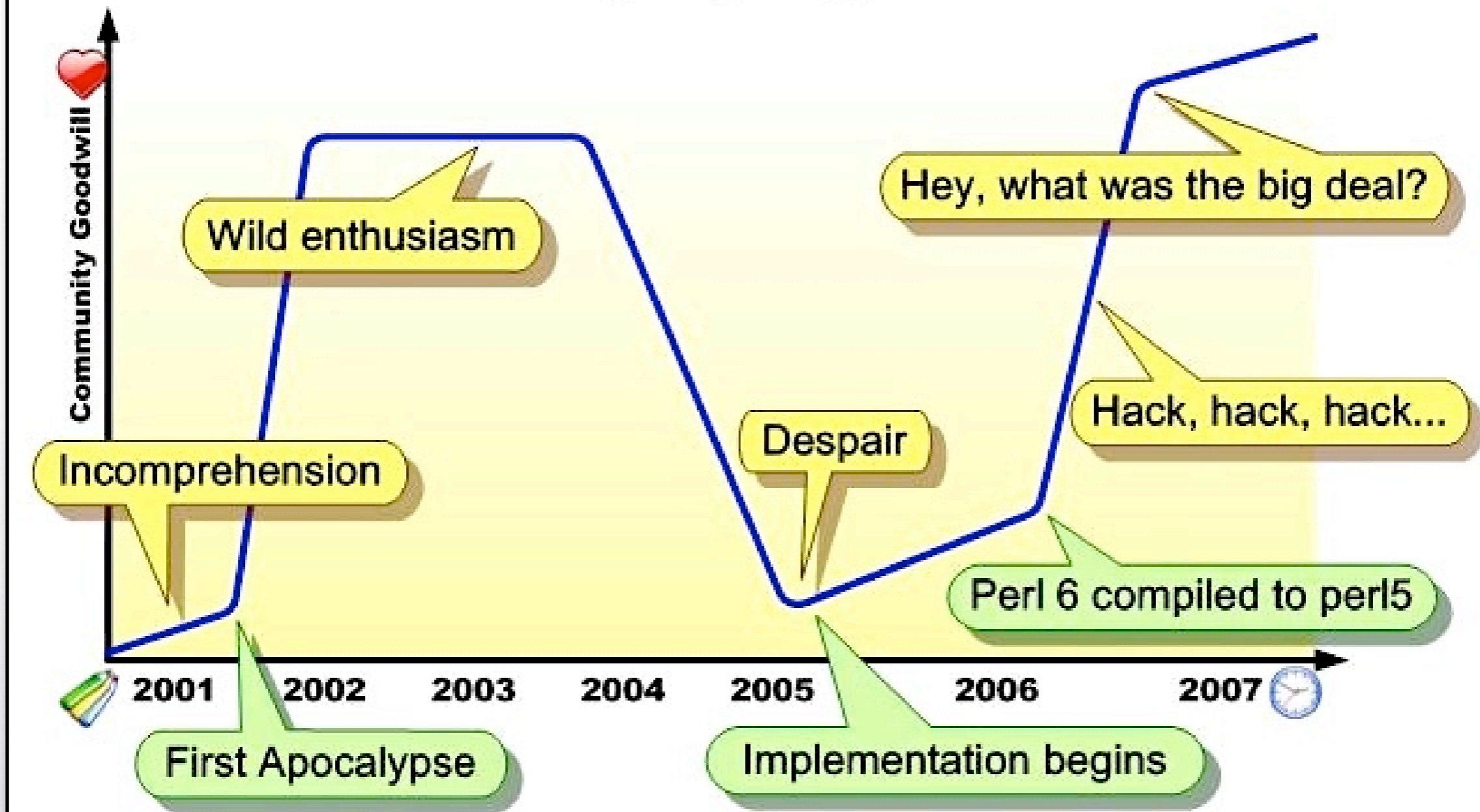
2004

Synopses

2005 Pugs

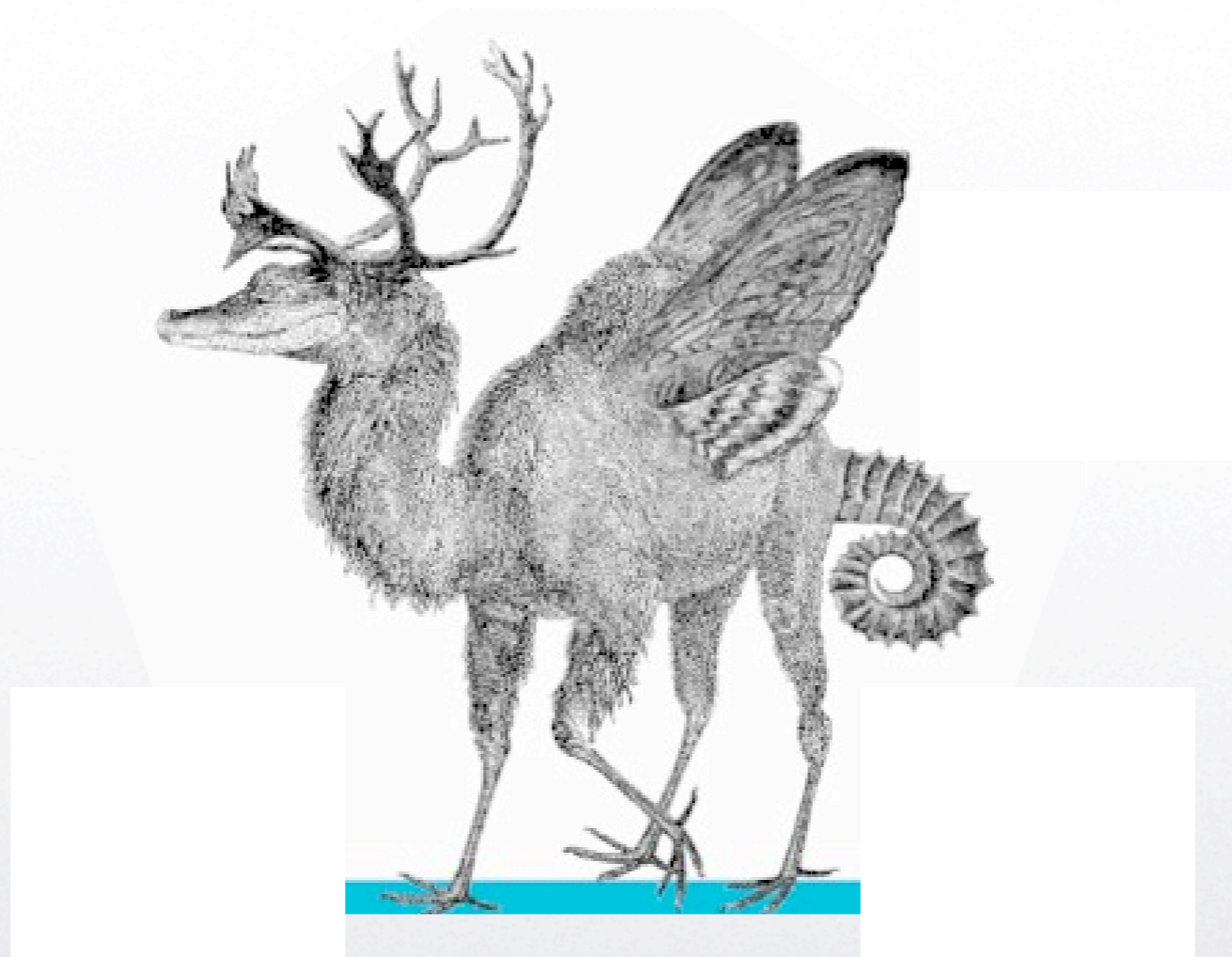
2006
v6-alpha

Perl 6 - (Imaginary) Timeline





Reconcile the Irreconcilable



Static

vs

Dynamic

Types



Typing



```
use v5;  
sub f {  
    sqrt($_[0] ** 2 + $_[1] ** 2)  
}  
my $five = f( 3, 4 );
```

```
use v6-alpha;  
sub f {  
    sqrt(@_[0] ** 2 + @_[1] ** 2)  
}  
my $five = f( 3, 4 );
```

Gradual Typing with Annotations

```
sub f (Num $x, Num $y) {  
    sqrt($x ** 2 + $y ** 2);  
}  
my Num $five = f( 3, 4 );
```


subset Positive of Num where { $\$__ > 0$ }


```
subset Positive of Num where { $ _ > 0 }
```

```
sub f (  
    Positive $x, Positive $y  
    --> Positive where { $ _ >= ($x & $y) }  
) {
```

```
subset Positive of Num where { $_ > 0 }
```

```
sub f (  
    Positive $x, Positive $y  
    --> Positive where { $_ >= ($x & $y) }  
) {  
    sqrt($x ** 2 + $y ** 2);  
}
```

```
subset Positive of Num where { $ _ > 0 }
```

```
sub f (  
  Positive $x, Positive $y  
  --> Positive where { $ _ >= ($x & $y) }  
) {  
  sqrt($x ** 2 + $y ** 2);  
}
```

```
my $five := f(3, 4); # inferred as Positive
```

Compiler

vs

Runtime

BEGIN {...}

BEGIN {...}

- **Compiler *is* a REPL**

BEGIN {...}

- **Compiler *is* a REPL**
- **Expose the entire compiler**

BEGIN {...}

- **Compiler *is* a REPL**
- **Expose the entire compiler**
- **All parts are swappable**

BEGIN {...}

- **Compiler *is* a REPL**
- **Expose the entire compiler**
- **All parts are swappable**
- **Even the lexer**

```
macro circumfix:</* */> ($x)
  is parsed /.*/
  { '' }
```

```
/* This is a C-style comment */
```

```
sub postfix:<!> ($x) {  
    [*] 1..$x  
}
```

```
say 10!; # 3628800
```

```
macro GREETING () {  
  # A Late-bound macro  
  q:code(:COMPILING){ "Hello, $s" };  
}
```

```
my $s = "world";  
say GREETING; # Hello, world
```

Lazy

vs

Eager


```
# "cat"  
for =<> { .say }
```

```
# "cat"  
for =<> { .say }
```

```
# "cat" with line numbers  
for each(0..*; =<>) {  
    say "Line $^num: $^text";  
}
```



```
# Lists are lazy streams!  
my @fib = (
```

```
# Lists are lazy streams!  
my @fib = (  
    0, 1,
```

```
# Lists are Lazy streams!  
my @fib = (  
    0, 1,  
    each(@fib; @fib[1..*]).map(&infix:<+>  
)
```

```
# Lists are Lazy streams!  
my @fib = (  
    0, 1,  
    each(@fib; @fib[1..*]).map(&infix:<+>  
)  
);  
  
say "The first ten numbers are: @fib[^10]";
```


Items are eager values. However...

```
# Items are eager values. However...  
my $ignored = lazy { 9 ** 9 ** 9 };
```



```
# Items are eager values. However...  
my $ignored = lazy { 9 ** 9 ** 9 };  
my $unused = lazy { say [1..$ignored] };
```

```
# Items are eager values. However...  
my $ignored = lazy { 9 ** 9 ** 9 };  
my $unused = lazy { say [1..$ignored] };  
  
say "Hello, world!";
```

Classes

vs

Prototypes


```
class Dog is Mammal does Pet {
```

```
class Dog is Mammal does Pet {  
  my $.count where 0..100;
```

```
class Dog is Mammal does Pet {  
  my $.count where 0..100;  
  
  has $!brain;
```

```
class Dog is Mammal does Pet {  
  my $.count where 0..100;  
  
  has $!brain;  
  
  has &.vocalize = &say;  
  has $.name is rw = "fido";  
}
```



```
class Dog is Mammal does Pet {  
  my $.count where 0..100;  
  
  has $!brain;  
  
  has &.vocalize = &say;  
  has $.name is rw = "fido";  
  
  has $.fur handles Groomable;  
  has $.tail handles <wag hang>;  
}
```

```
class Dog is Mammal does Pet {  
  my $.count where 0..100;  
  
  has $!brain;  
  
  has &.vocalize = &say;  
  has $.name is rw = "fido";  
  
  has $.fur handles Groomable;  
  has $.tail handles <wag hang>;  
  
  method owner () handles s/^owner_// { ... }  
  
}
```



```
my Dog $fido .= new;
```

```
my Dog $fido .= new;
```

```
$fido.HOW; # the meta object for Dog
```

```
my Dog $fido .= new;
```

```
$fido.HOW; # the meta object for Dog
```

```
$fido.WHAT; # the Dog prototype object
```

```
my Dog $fido .= new;
```

```
$fido.HOW;    # the meta object for Dog  
$fido.WHAT;   # the Dog prototype object  
$fido.WHICH;  # $fido's Object ID
```

```
my Dog $fido .= new;
```

```
$fido.HOW;    # the meta object for Dog  
$fido.WHAT;   # the Dog prototype object  
$fido.WHICH;  # $fido's Object ID
```

```
Dog.isa(Dog); $fido.isa(Dog);
```



```
$fido.HOW.add_method(  
    'bark',  
    method () { $.vocalize('Woof!') }  
);
```

```
$fido.HOW.add_method(  
    'bark',  
    method () { $.vocalize('Woof!') }  
);
```

```
Dog.can('bark'); $fido.can('bark');
```

Parallelism

vs

Sanity


```
# Hyper Operator (SSE/GPU friendly)  
[1, 1, 2, 3, 5] »+« [1, 2, 3, 5, 8];
```

```
# Hyper Operator (SSE/GPU friendly)
[1, 1, 2, 3, 5] »+« [1, 2, 3, 5, 8];
# === [2, 3, 5, 8, 13]
```


Recursive Visits

- « **[[1, 2], 3]**;

```
# Recursive Visits  
-« [[1, 2], 3];  
# === [[-1, -2], -3]
```


Hyper Methods

[1, 4, 9, 16]»**.sqrt;**

```
# Hyper Methods  
[1, 4, 9, 16]».sqrt;  
# === [1, 2, 3, 4]
```



```
% time env GHCRTS=-N1 pugs -e '(1..100000)>>.sqrt'  
real 9.387s  
user 9.219s
```

```
% time env GHCRTS=-N2 pugs -e '(1..100000)>>.sqrt'  
real 5.807s  
user 6.959s
```


Junctions

```
sub is_prime (Int $n --> Bool) {  
    $n % all(2 .. $n.sqrt+1);  
}
```

Junctions

```
sub is_prime (Int $n --> Bool) {  
    $n % all(2 .. $n.sqrt+1);  
}
```

```
sub has_twin_prime (Int $n --> Bool) {  
    is_prime($n & ($n ± 2));  
}
```

Junctions

```
sub is_prime (Int $n --> Bool) {  
    $n % all(2 .. $n.sqrt+1);  
}
```

```
sub has_twin_prime (Int $n --> Bool) {  
    is_prime($n & ($n ± 2));  
}
```

```
sub infix:<±> ($x, $y) {  
    ($x + $y) | ($x - $y);  
}
```

Junctions

```
sub is_prime (Int $n --> Bool) {  
    $n % all(2 .. $n.sqrt+1);  
}
```

```
sub has_twin_prime (Int $n --> Bool) {  
    is_prime($n & ($n ± 2));  
}
```

```
sub infix:<±> ($x, $y) {  
    ($x + $y) | ($x - $y);  
}
```

Concurrency



Locking




```
async {  
    $x.withdraw(3);  
  
    $y.deposit(3);  
}
```



```
async {  
    $x.withdraw(3);  
    ☠ Race Condition ☠  
    $y.deposit(3);  
}
```



```
async {  
  $x.lock;  
  $y.lock;
```

```
async {  
    $x.lock;  
    $y.lock;  
    $x.withdraw(3);  
    $y.deposit(3);  
}
```

```
async {  
    $x.lock;  
    $y.lock;  
    $x.withdraw(3);  
    $y.deposit(3);  
}
```

```
async {  
    $y.lock;  
    $x.lock;  
}
```

☠️ Deadlock ☠️

```
async {  
  $x.lock;  
  $y.lock;  
  $x.withdraw(3);  
  $y.deposit(3);  
}
```

```
async {  
  $y.lock;  
  $x.lock;  
}
```

Software Transactional Memory


```
# No Locks, no races!  
contend {  
    $x.withdraw(3);  
    $y.deposit(3);  
}
```


Retry with "defer"

```
# Retry with "defer"  
method withdraw ($n) {
```

```
# Retry with "defer"  
method withdraw ($n) {  
    defer if $.balance < n;  
}
```

```
# Retry with "defer"  
method withdraw ($n) {  
  defer if $.balance < n;  
  $.balance -= $n;  
}
```

```
# Retry with "defer"  
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

```
# Retry with "defer"  
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```



```
# Retry with "defer"  
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

```
# Choice with "maybe"
```

```
# Retry with "defer"  
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

```
# Choice with "maybe"  
sub transfer ($x1, $x2, $y) {
```

Retry with "defer"

```
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
    maybe { $x1.withdraw(3) }
```

Retry with "defer"

```
method withdraw ($n) {  
  defer if $.balance < n;  
  $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
  maybe { $x1.withdraw(3) }  
  maybe { $x2.withdraw(3) }  
}
```

Retry with "defer"

```
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
    maybe { $x1.withdraw(3) }  
    maybe { $x2.withdraw(3) }  
    $y.deposit(3);  
}
```

Retry with "defer"

```
method withdraw ($n) {  
  defer if $.balance < n;  
  $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
  maybe { $x1.withdraw(3) }  
  maybe { $x2.withdraw(3) }  
  $y.deposit(3);  
}
```

Retry with "defer"

```
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
    maybe { $x1.withdraw(3) }  
    maybe { $x2.withdraw(3) }  
    $y.deposit(3);  
}
```

Retry with "defer"

```
method withdraw ($n) {  
  defer if $.balance < n;  
  $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
  maybe { $x1.withdraw(3) }  
  maybe { $x2.withdraw(3) }  
  $y.deposit(3);  
}
```

Composable with nested "maybe"

Retry with "defer"

```
method withdraw ($n) {  
  defer if $.balance < n;  
  $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
  maybe { $x1.withdraw(3) }  
  maybe { $x2.withdraw(3) }  
  $y.deposit(3);  
}
```

Composable with nested "maybe"

```
contend {
```

Retry with "defer"

```
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
    maybe { $x1.withdraw(3) }  
    maybe { $x2.withdraw(3) }  
    $y.deposit(3);  
}
```

Composable with nested "maybe"

```
contend {  
    maybe { transfer($x1, $x2, $y) }  
}
```

Retry with "defer"

```
method withdraw ($n) {  
  defer if $.balance < n;  
  $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
  maybe { $x1.withdraw(3) }  
  maybe { $x2.withdraw(3) }  
  $y.deposit(3);  
}
```

Composable with nested "maybe"

```
contend {  
  maybe { transfer($x1, $x2, $y) }  
  maybe { transfer($x3, $x4, $y) }
```

Retry with "defer"

```
method withdraw ($n) {  
    defer if $.balance < n;  
    $.balance -= $n;  
}
```

Choice with "maybe"

```
sub transfer ($x1, $x2, $y) {  
    maybe { $x1.withdraw(3) }  
    maybe { $x2.withdraw(3) }  
    $y.deposit(3);  
}
```

Composable with nested "maybe"

```
contend {  
    maybe { transfer($x1, $x2, $y) }  
    maybe { transfer($x3, $x4, $y) }  
}
```

My Language

vs

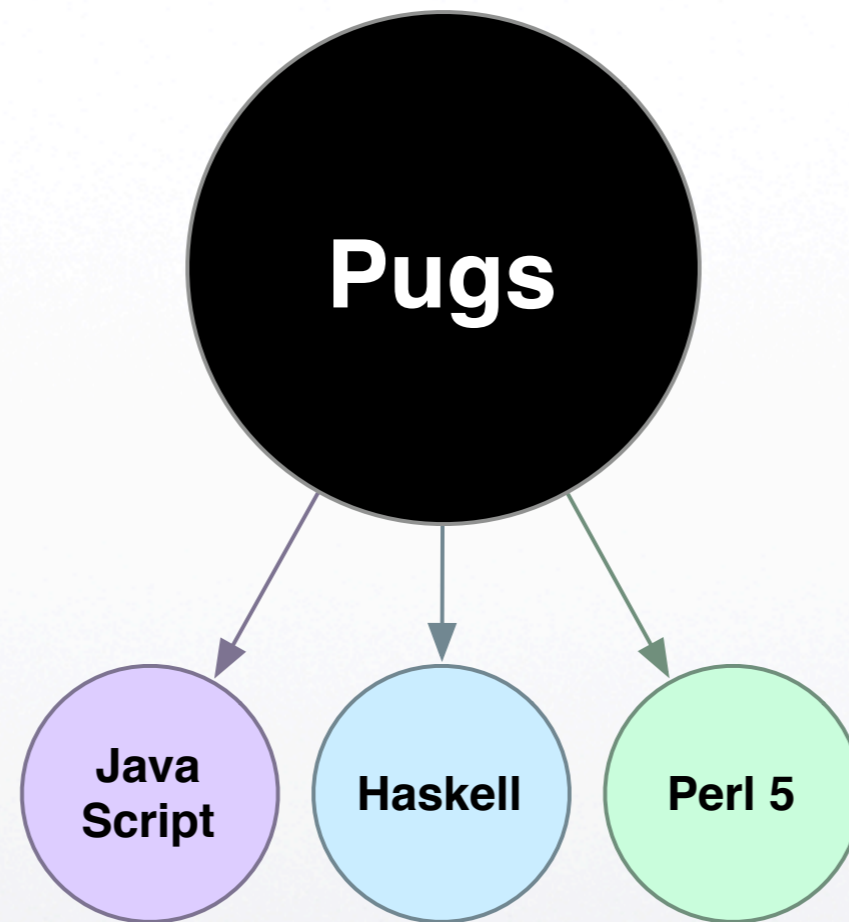
Your Language

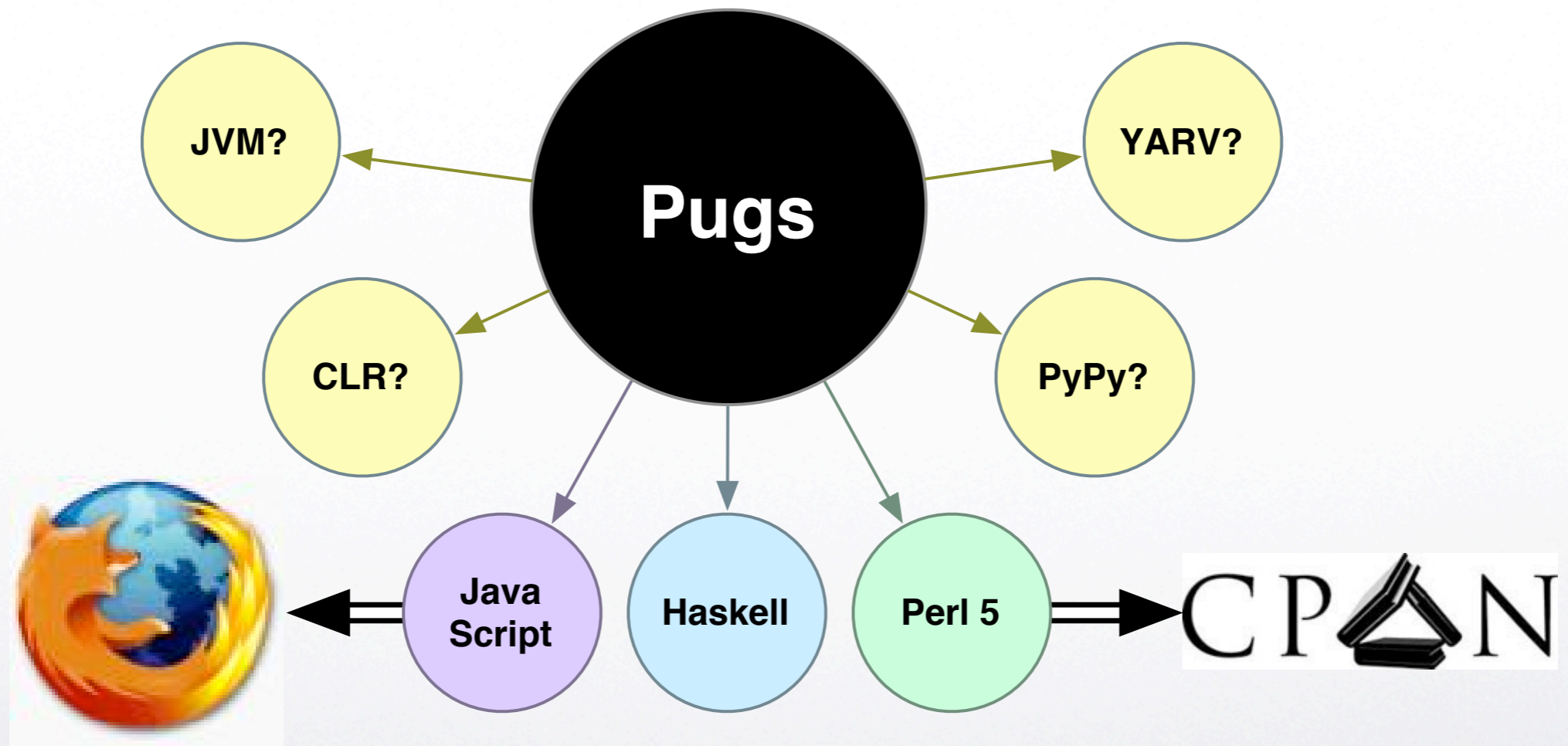

```
use jsan:DOM;
```

```
use jsan:DOM;  
use perl5:DBI;
```



```
use jsan:DOM;  
use perl5:DBI;  
use haskell:Numeric;
```







Popular Target Language

Generating JS

Generating JS

ژ **Java: Google Web Toolkit**

Generating JS

ژ **Java: Google Web Toolkit**

ژ **Perl: Jifty**

Generating JS

ژ **Java: Google Web Toolkit**

ژ **Perl: Jifty**

ژ **Ruby: Rails/JS**

Generating JS

ژ **Java: Google Web Toolkit**

ژ **Perl: Jifty**

ژ **Ruby: Rails/JS**

ژ **Python: Pyjamas**

Generating JS

ژ Java: Google Web Toolkit

ژ Perl: Jifty

ژ Ruby: Rails/JS

ژ Python: Pyjamas

ژ C#: Script#

**Same language
for both sides**



Client-side

just a tiny subset

x

Compiling to JS

Compiling to JS

🔗 HOP/Scheme2JS

Compiling to JS

۳ HOP/Scheme2JS

۳ Links

Compiling to JS

پ HOP/Scheme2JS

پ Links

پ HaXe

Compiling to JS

👉 HOP/Scheme2JS

👉 Links

👉 HaXe

👉 Pugs!

PIL2JS

pugs -C JS

pugs -C JS

ژ Written in Perl 5

pugs -C JS

❑ **Written in Perl 5**

❑ **Passes 90% of tests**

pugs -C JS

ش **Written in Perl 5**

ش **Passes 90% of tests**

ش **~30k Runtime**

PIL2JS Runtime

PIL2JS Runtime

ث Primitives & Autoboxing

PIL2JS Runtime

‣ Primitives & Autoboxing

‣ Meta-object protocol

PIL2JS Runtime

- ❑ **Primitives & Autoboxing**
- ❑ **Meta-object protocol**
- ❑ **Supports JSAN libraries**

JSAN

JSAN

ژ `"CPAN".replace(/CP/, "JS")`

JSAN

ژ `"CPAN".replace(/CP/, "JS")`

ژ **Module system with Prototype.js**

JSAN

ژ `"CPAN".replace(/CP/, "JS")`

ژ **Module system with Prototype.js**

ژ **Test.Simple, Jemplate, *etc.***

Shortcomings

Shortcomings

- ث Calling convention too complex

Shortcomings

- ❑ Calling convention too complex
- ❑ CPS runloop is slow

Shortcomings

- ❑ Calling convention too complex
- ❑ CPS runloop is slow
- ❑ No tail recursion nor *goto*

Shortcomings

- ❑ Calling convention too complex
- ❑ CPS runloop is slow
- ❑ No tail recursion nor *goto*
- ❑ But there's hope!

JS 2.0

JS 2.0

ث Self hosting

JS 2.0

- ث Self hosting
- ث Backtranslate to JS₁

JS 2.0

ژ Self hosting

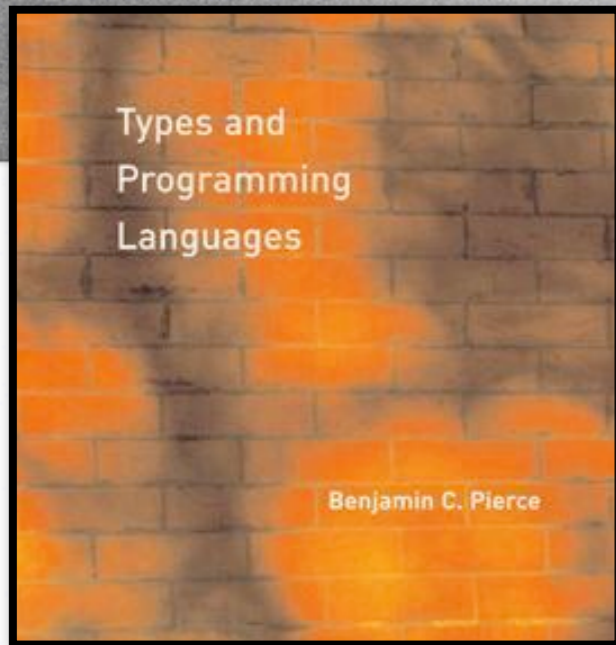
ژ Backtranslate to JS₁

ژ Types, Modules, Continuations

JS 2.0

- Self hosting
- Backtranslate to JS₁
- Types, Modules, Continuations
- Part of Firefox 3.0 (next year)





Feb 1

TaPL arrived
as an exercise...

Feb 6

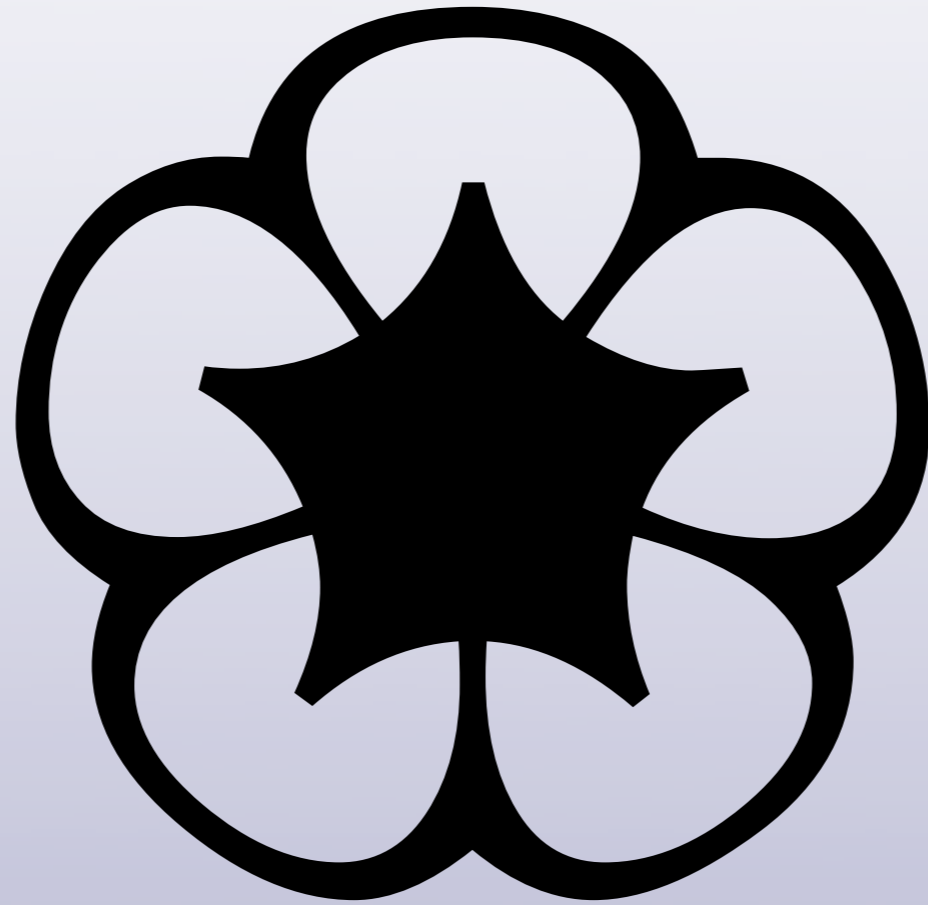
Junctions

$(1 | 2) + (3 | 4) \rightsquigarrow (4 | 5 | 6)$

Feb 16

Input/Output

say "Hello, world"



Mar 19

PCRE Regex

s : P5 : g / 5 / 6 / ;

May 8

svnbot.p6

r2851 | ib1ech++

May 25

Prelude.pm

```
sub sprintf ($fmt, *@args)
```

May 29

Embedded Perl 5

```
use perl5 :DBI;
```



Jun 2

evalbot.p6

[#per16] ?eval 1+1

Jun 24

Perl6 → *PIL* → *Parrot*

make smoke-pir

Jul 14

PIL* → *Perl5

make smoke-per15

Jul 17

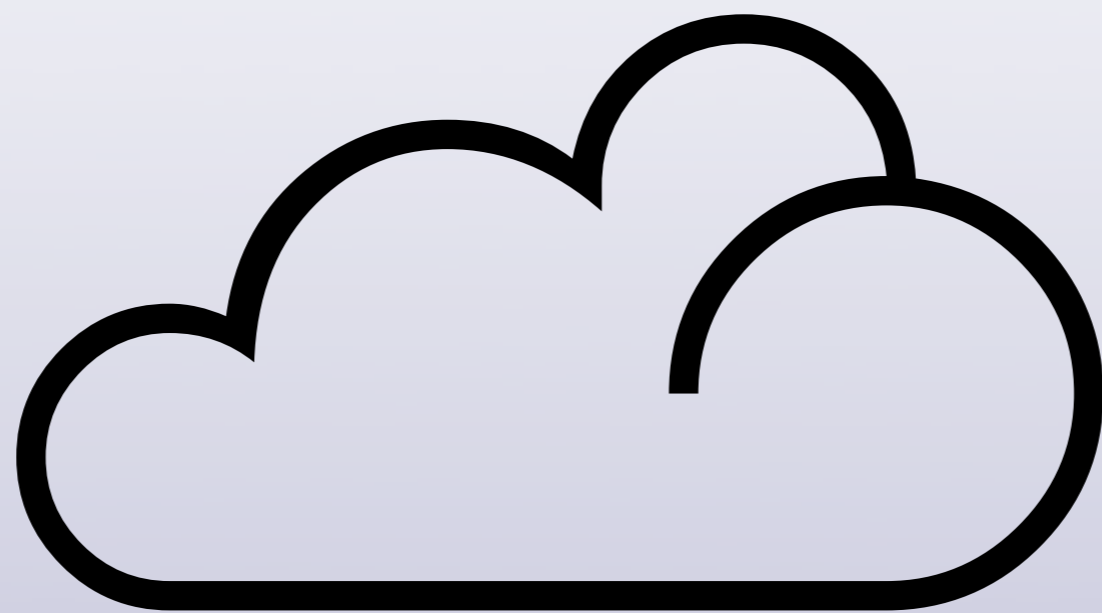
PIL* → *JavaScript

make smoke-js

Aug 3

Kontent Wiki

use per15:Template;



Nov 2nd

Runtime API

Per16::ObjectSpace

Nov 3

Concurrency API

```
sub f is throttled(:limit(3)) { ... }
```

Nov 4

Packaging API

`perl5-Foo-1.0-cpan+KANE.jib`

Nov 7

Coroutines

```
coro { yield 1; }
```

Nov 23

1st commit from Larry
(still waiting for Guido 😊)



Jan 6

YAML Serialization

say \$x.yaml;

Feb 3

Self-parsing Grammar

grammar Grammar;

Feb 22

Larry joins #perl6

<fglock> TimToady: welcome

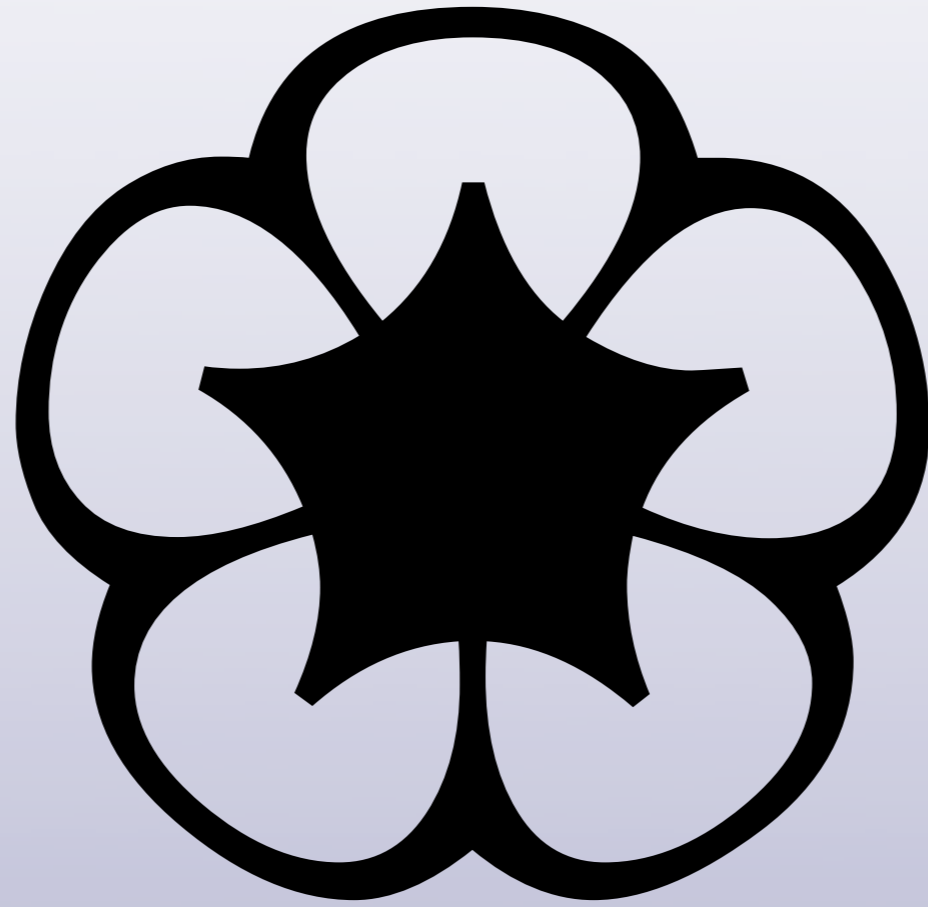
<Juerd> Just try to not get addicted :)

<TimToady> Juerd: too late..

Feb 25

Code DOM

```
$AST = q:code/ say "hi" /;
```



Mar 11

Evaluator in Perl 5

Pugs :: Runtime

Mar 16

Bootstrapped on Perl 5

1rep.p6 1rep.p6

Apr 1

Calling Convention API

```
$tree = \($obj: attr => 1, $child);
```

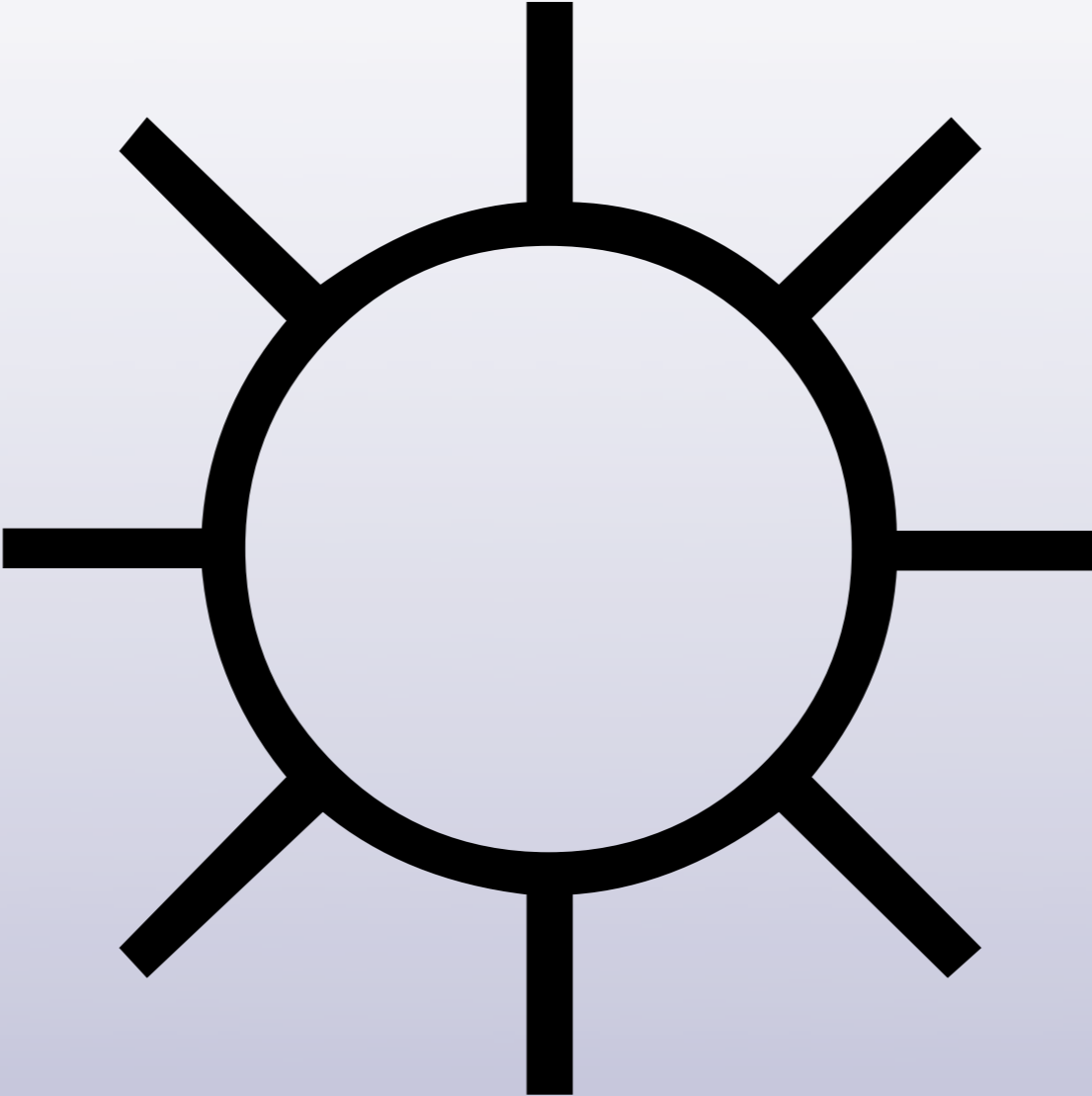
Apr 21

MIT License

May 8

Predictive Parsing

<TimToady> "do, or do not.
there is no *try*..."

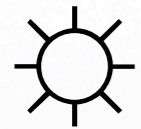


June 1

Summer of Code

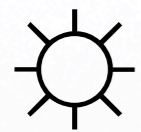
SoC: Perl.org

SoC: Perl.org

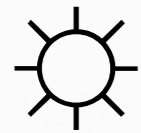


Perl 6 DBI Module

SoC: Perl.org

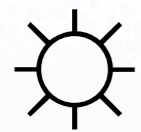


Perl 6 DBI Module

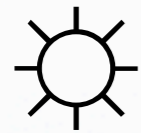


Perl 6 to Perl 5 Translator

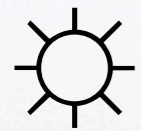
SoC: Perl.org



Perl 6 DBI Module

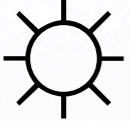


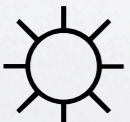


Perl 6 to Perl 5 Translator



Pugs Bootstrap From Perl 5 and Rules

SoC: Perl.org

-  **Perl 6 DBI Module**
-  **Perl 6 to Perl 5 Translator**
-  **Pugs Bootstrap From Perl 5 and Rules**
-  **Software Transactional Memory for Parrot**

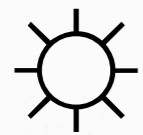
SoC: Haskell.org

SoC: Haskell.org

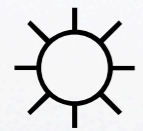


Fast Mutable Collection Types

SoC: Haskell.org



Fast Mutable Collection Types



Unicode ByteString and Data.Rope

June 4

Software Transactional Memory

async { contend { ... } }

June 26

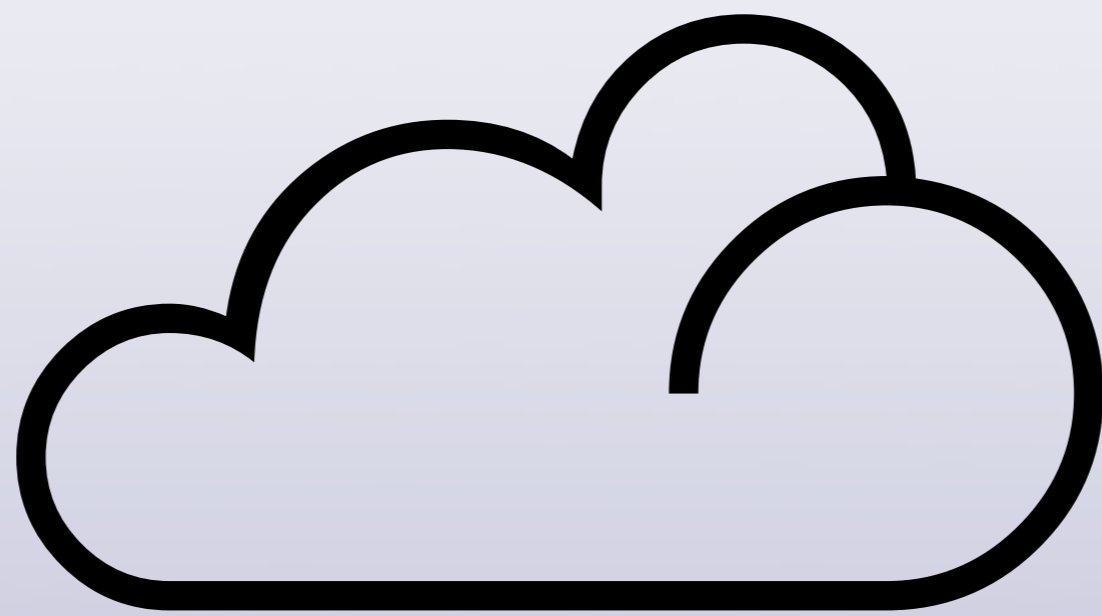
v6.pm

use v6-alpha;

August 17

Smartlinks.pl

L <S02/"bidirectional mirrorings">
is q «123», 123, "angle brackets";



September 16

*Native Grammar Engine
via Embedded Perl5*

`s : g/PGE/PCR/ ;`

October 9

Fully reentrant continuations

```
sub callcc (Code &c)  
{ &c(&?CALLER_CONTINUATION) }
```

October 11

GHC 6.6

<TimToady> I upgraded and my \$job
program ran 60 times faster...

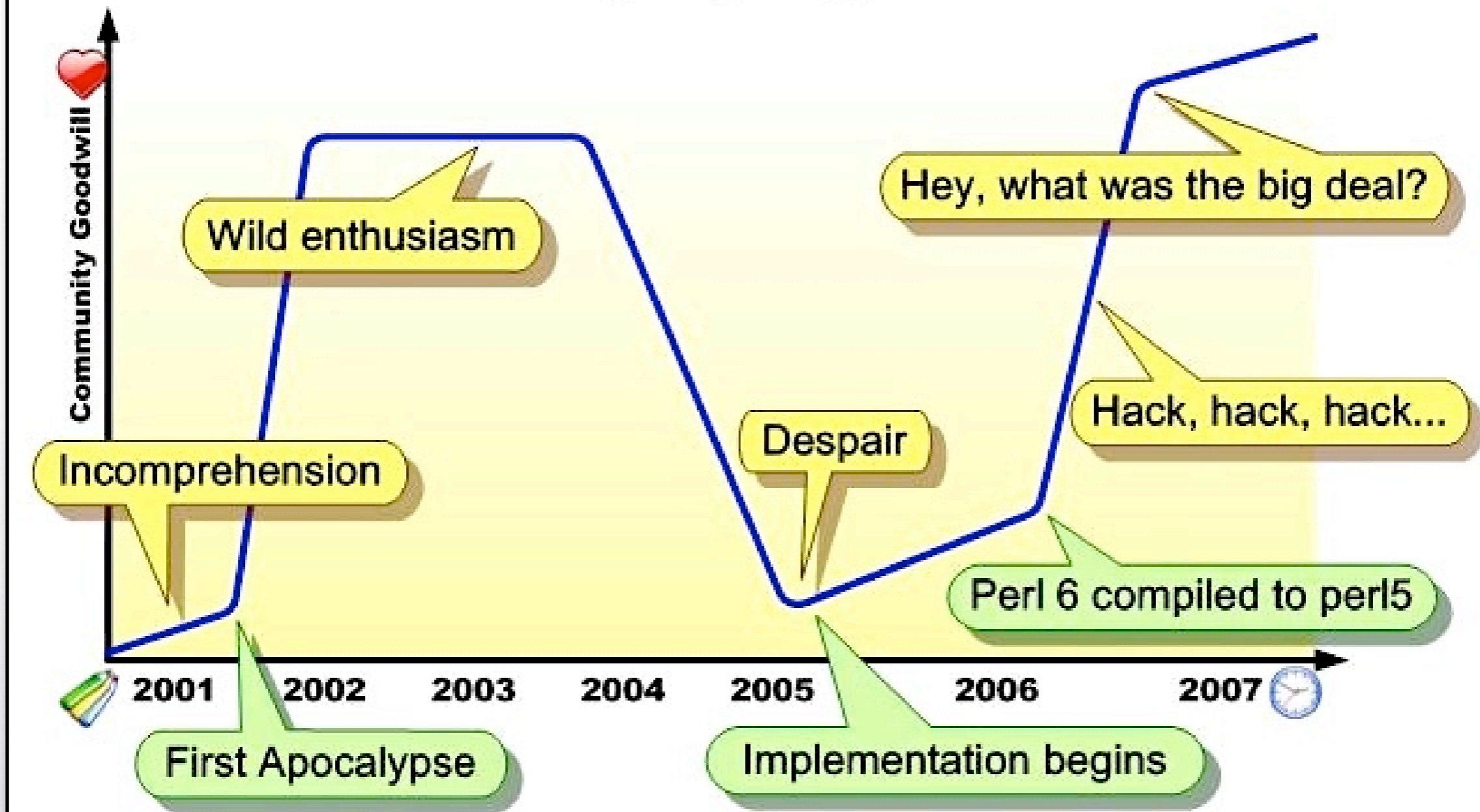
October 20

SMP Data Parallelism

```
[(1|2), (3|4)].>>sqrt
```



Perl 6 - (Imaginary) Timeline



**“CPAN is the language
Perl is just *syntax*”**

Production

Production

🏠 Existing Perl 5 code base

Production

⤴ Existing Perl 5 code base

⤴ GHC may be unavailable

Production

- ⤴ Existing Perl 5 code base
- ⤴ GHC may be unavailable
- ⤴ Can't rewrite from scratch

The Perl 5 VM

The Perl 5 VM

🏠 **Actively developed**

The Perl 5 VM

🏠 **Actively developed**

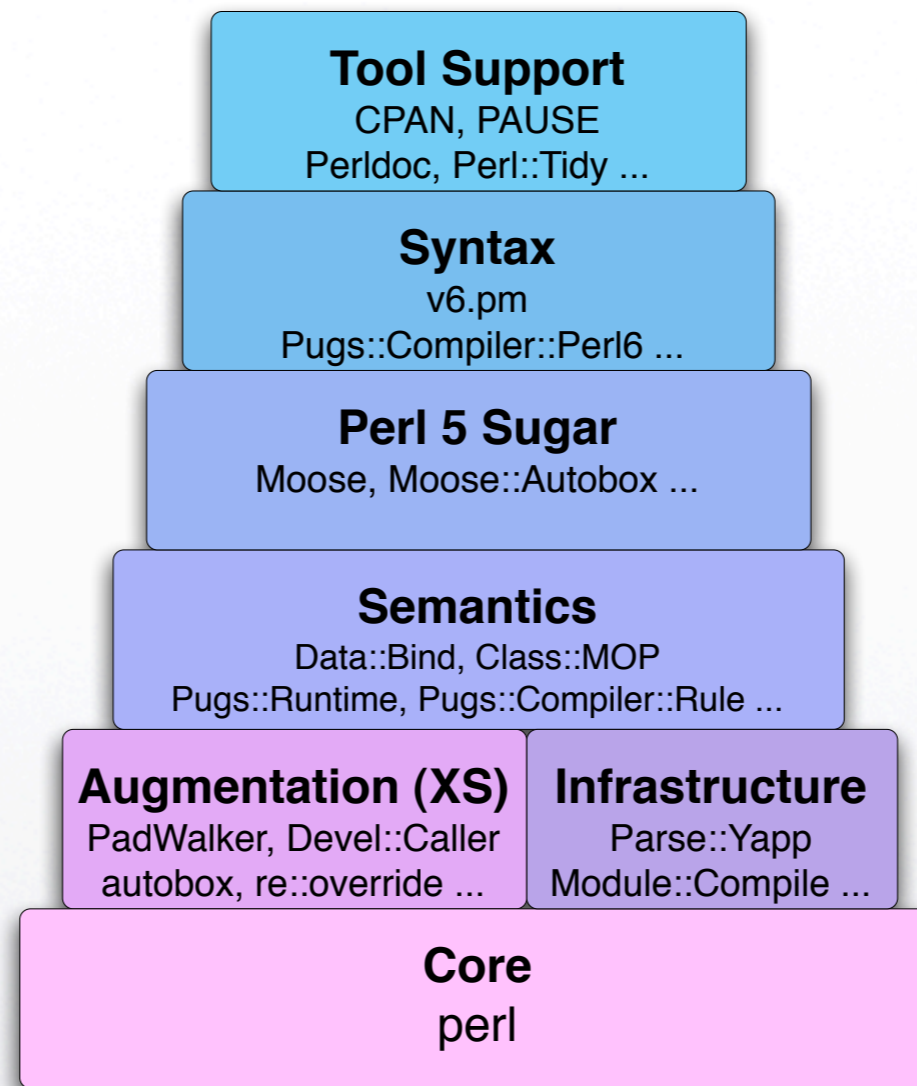
🏠 **5.10 is much more v6ish**

The Perl 5 VM

- 🏠 **Actively developed**
- 🏠 **5.10 is much more v6ish**
- 🏠 **Just needs a Perl 6 runtime**

Experimental

Perl 6's CPAN stack



Stable

Moose



Objects with Class

```
use v6-alpha;  
class Point;
```

```
has $.x is rw; # instance attributes  
has $.y;       # default "is readonly"
```

```
method clear () {
```

```
    $.x = 0; # accessible within the class
```

```
    $.y = 0;
```

```
}
```

```
use v5;
package Point;
use Moose;

has x => (is => 'rw');
has y => (is => 'ro');

sub clear {
    my $self = shift;

    $self->{x} = 0;
    $self->y(0);
}
```

Subclassing

```
use v6-alpha;
class Point3D;

is Point;

has $.z;

method clear () {
    call;
    $.z = 0;
};
```

```
use v5;  
package Point3D;  
use Moose;  
  
extends 'Point';  
  
has z => (isa => 'Int');  
  
override clear => sub {  
    my $self = shift;  
    super;  
    $self->{z} = 0;  
};
```

```
use v5;  
package Point3D;  
use Moose;  
  
extends 'Point';  
  
has z => (isa => 'Int');  
  
after clear => sub {  
    my $self = shift;  
  
    $self->{z} = 0;  
};
```

Subset Types


```

use v6-alpha;
class Address;
use perl5:Locale::US;
use perl5:Regex::Common <zip $RE>;

my $STATES = Locale::US.new;
subset US_State of Str where {
    $STATES{any(<code2state state2code>)}{.uc};
};

has US_State $.state is rw;
has Str $.zip_code is rw where {
    $_ ~~ $RE<zip><<US>{'-extended' => 'allow'}
};

```

```

use v5;
package Address;
use Moose;
use Moose::Util::TypeConstraints;
use Locale::US;
use Regexp::Common 'zip';

my $STATES = Locale::US->new;
subtype USState => as Str => where {
    $STATES->{code2state}{uc($_)}
    or $STATES->{state2code}{uc($_)};
}

has state => (is => 'rw', isa => 'USState');
has zip_code => (
    is => 'rw',
    isa => subtype Str => where {
        /$RE{zip}{US}{-extended => 'allow'}/
    },
);

```

More features

More features

 **Roles (Dynamic Traits)**

More features

 **Roles (Dynamic Traits)**

 **Coercion**

More features

 **Roles (Dynamic Traits)**

 **Coercion**

 **Meta Objects**

Module::Compile



Source Filter


```
use v5;
use Filter::Simple sub {
    s{(^ sub \s+ \w+ \s+ \{ )}
        {$1\nmy $self = shift;\n}mgx;
}
```

Filter::Simple Bad

Filter::Simple Bad

✱ Extra dependency

Filter::Simple Bad

- ✖ Extra dependency
- ✖ Slows down startup

Filter::Simple Bad

- ❖ **Extra dependency**
- ❖ **Slows down startup**
- ❖ **Breaks the debugger**

Filter::Simple Bad

- ❖ **Extra dependency**
- ❖ **Slows down startup**
- ❖ **Breaks the debugger**
- ❖ **Wrecks other Source Filters**

We can **fix it!**

```
use v5;  
use Filter::Simple sub {  
    s{(^ sub \s+ \w+ \s+ \{ )}  
    {$1\nmy $self = shift;\n}mgx;  
}
```



```
use v5;  
use Filter::Simple::Compile sub {  
    s{(^ sub \s+ \w+ \s+ \{ )}  
    {$1\nmy $self = shift;\n}mgx;  
}
```

How?

Little-known fact:

“use Foo”

looks for Foo.pmc

before Foo.pm

```
% echo 'print "Hello\n"' > Foo.pmc
```

```
% perl -MFoo -e1
```

```
Hello
```

**Save filtered
results to .pmc...**

**...no filtering
needed next time!**

Module::Compile Good

Module::Compile Good

- ✱ Free of dependencies on user's site

Module::Compile Good

- ✱ Free of dependencies on user's site
- ✱ Fast startup time

Module::Compile Good

- ✱ Free of dependencies on user's site
- ✱ Fast startup time
- ✱ Debuggable source is all in .pmc

Module::Compile Good

- ✦ Free of dependencies on user's site
- ✦ Fast startup time
- ✦ Debuggable source is all in .pmc
- ✦ Composable precompilers

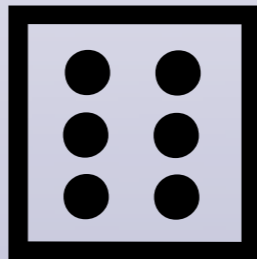
Filter :: Simple :: Compile

```
# Drop-in replacement to Filter::Simple  
package Acme::Y2K;  
use Filter::Simple::Compile sub {  
    tr/y/k/;  
}
```

```
# It's Lexical!
{
    use Acme::Y2K;
    package Foo;
    mydir "tmp";
}
my $normal_code_here;
```

Deploying Perl 6

v6.pptm



Source:
Rule.pm

```
use v6-alpha;
```

```
grammar Pugs::Grammar::Rule;
```

```
rule ws :P5 {
```

```
    ^((?:\s|\#(?:-s:.)*)+)
```

```
}
```

```
# ...more rules...
```

Target:
Rule.pmc

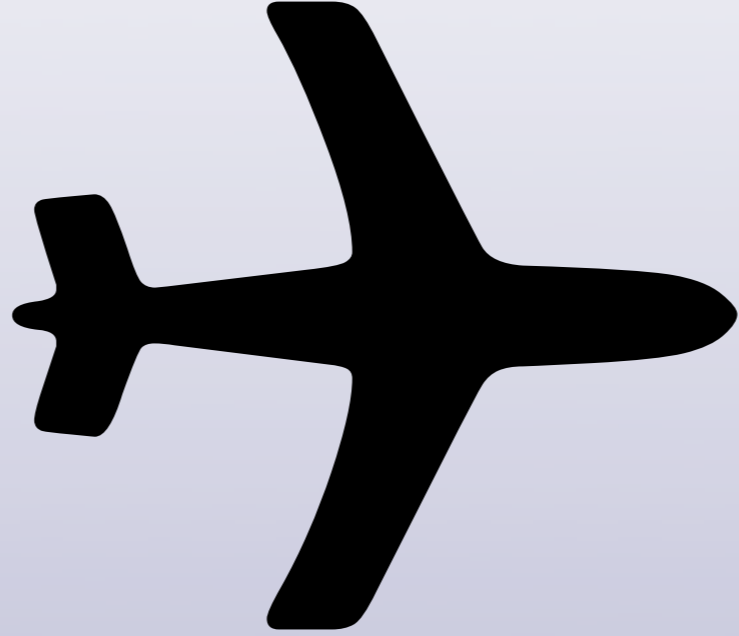
```

# Generated file - do not edit!
#####((( 32-bit Checksum Validator )))#####
BEGIN { use 5.006; local (*F, $/); ($F = __FILE__) =~ s!c$!!; open(F)
or die "Cannot open $F: $!"; binmode(F, ':crlf'); unpack('%32N*', <F>)
== 0x1D6399E1 or die "Checksum failed for outdated .pmc file: ${F}c"}
#####
package Pugs::Grammar::Rule;
use base 'Pugs::Grammar::Base';
*{'Pugs::Grammar::Rule::ws'} = sub {
    my $grammar = shift;
    #warn "rule argument is undefined" unless defined $_[0];
    $_[0] = "" unless defined $_[0];
    my $bool = $_[0] =~ /^((?:\s|\#(?:-s:.)*)+)(.*)$/sx;
    return {
        bool => $bool,
        match => $1,
        tail => $2,
        #capture => $1,
    }
};
# ...more rules...

```

Write Perl 6
compile to Perl 5





**When will
Perl 6 be released?**

By Christmas!

When Perl 6 arrives,
every day will be like
Christmas!



Fin.