A State Of Perl 6

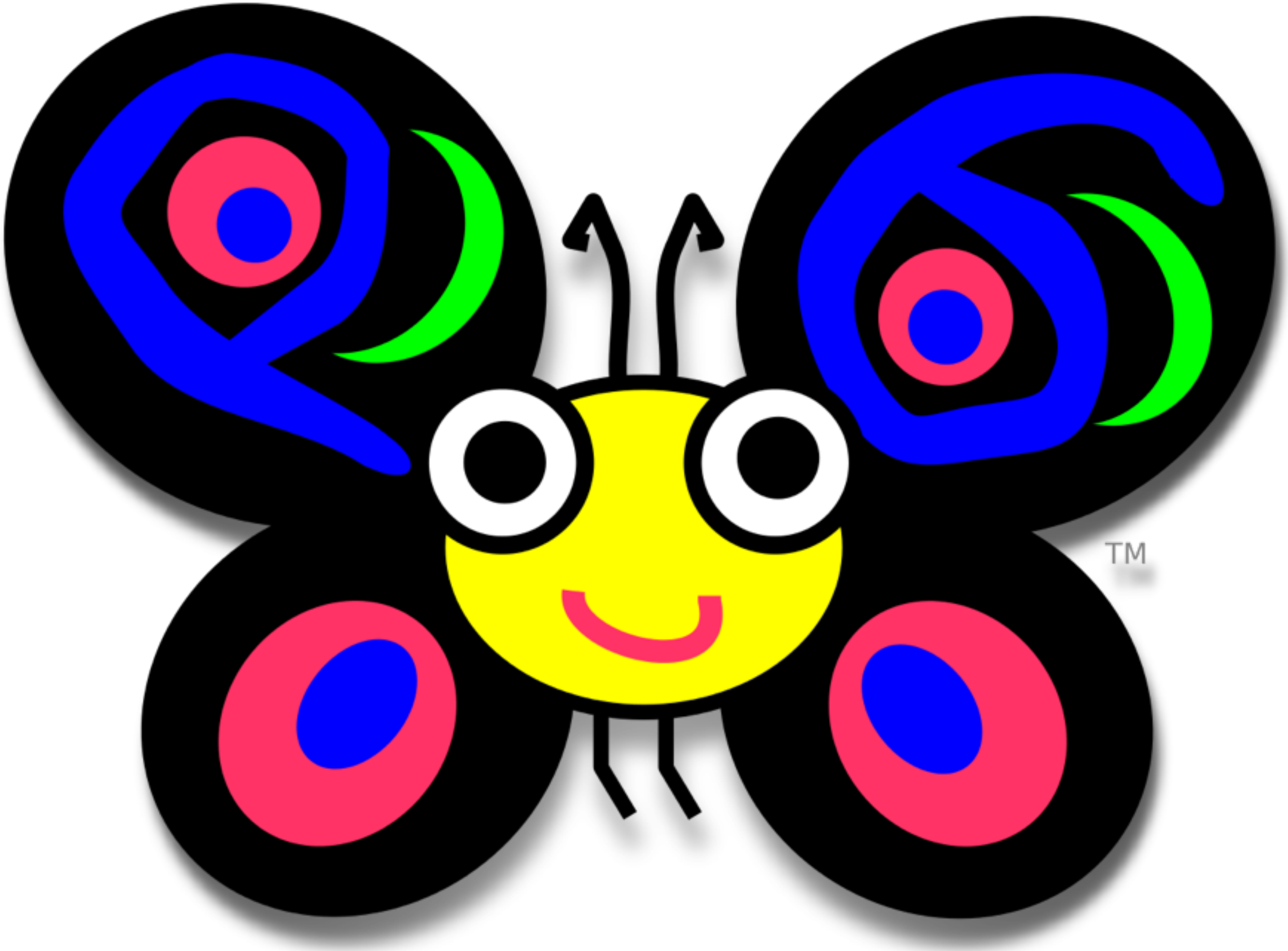
The growth of Camelia in the past year or so.

Elizabeth Mattijsen

24 August 2014

Wishing TimToady All The Best!

Hi, I’m Camelia!



And I’ve been Growing!

Nom Nom?

Rakudo and NQP Internals Workshop

- September 2013 in Frankfurt, Germany
- Developed / Presented by Jonathan Worthington
- 20 Attendees
- Course materials open-sourced
- http://6guts.wordpress.com/2013/09/17/material-from-the-rakudo-and-nqp-internals-course

12 Monthly Releases with new Features

Buf, squish, Promise, Thread, Channel, is default, is dynamic, will (phaser), once, start, .VAR, tr///, async sockets, EVAL, Nil, is DEPRECATED, uniname, uniprop, unival, Supply, winner, on, univals, minpairs, maxpairs, rotor, is cached, act, batch, stable, watch_path, bytes_supply, chars_supply, signal, first-index, last-index, grep-index, $*KERNEL, $*DISTRO, $*VM, $*PERL, later, earlier, use v5, LABEL:, next LABEL, last LABEL, :42nd, zip-latest, subbuf-rw, LoL, SEQ, HyperWhatever, —profile

(and all the other things I missed)

Not to mention the many bugfixes and many, many optimizations

3 Backends to Choose From

- **MoarVM**
  - Most functional, fastest, most development
- **JVM**
  - Only no signal support yet, very stable
- **Parrot**
  - Old faithful, no asynchronous support

Features that Perl 5 stole

(recently)

- **Subroutine signatures**
  - Fully functional, with multi-method dispatch
- **Lexical subs**
  - Almost everything is lexical, subs also
- **Smartmatch**
  - Still fully functional, **not** experimental!

MoarVM

(Metamodel On A Runtime)

- **The** Virtual Machine for Perl 6
- First **1.5 year** development in secret
- Now about **8** man years of development
- Using proven external libs when possible
- Running Rakudo since January
- http://www.moarvm.org/features.html

S17 - Concurrency

```
Design Philosophy
    Focus on composability
    Boundaries between synchronous and asynchronous should be explicit
    Implicit parallelism is OK
    Make the hard things possible
Schedulers
Promises
Channels
Supplies
System events exposed as Supplies
I/O features exposed as Supplies
The Event Loop
    Threads
    Atomic Compare and Swap
Low-level primitives
    Locks
    Semaphore
```

Rewritten from Scratch

http://perlcabal.org/syn/S17.html

Asynchronous and Reactive Programming

- Powerful **primitives** to handle complex situations
- **Promise**, to be kept or broken
- **Channel**, for queuing data to be processed
- **Supply**, for reactive programming
- **Thread**, only if you really, **really** must

Simple Explicit Parallelization

```
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
```

Simple Signal Handling

```
signal(SIGINT).tap( {
    say "Thank you for your attention”;
    exit;
} );
```

```
for @todo {
   state $quitting;
   state $tap = signal(SIGINT).tap( { $quitting = True } );
   LAST $tap.close;
   LEAVE exit(1) if $quitting;
   … # code to protect;
}
```

Spesh

(The MoarVM bytecode specializer)

Spesh seeks out hot code, sees what kinds of arguments it is given, and makes a specialized version.

Slightly longer:

Spesh takes code that is highly dynamic, with a lot of late binding and polymorphism, and – based on the actual types that show up at runtime – generates specialized versions of the code that do away with the costly late-binding. Or revert to the original code if it turns out to be more dynamic than initially thought.

Not something Perl 5 can ever have!

JIT for MoarVM

(GSOC Project)

- Uses the information gathered by **spesh**
- To create **JIT**ted **machinecode** at runtime
- Now available with **—jit-enable**
- This is **very** exciting!

And Perl 5 Migration?

- **use v5;**
- Parse Perl 5 source with Perl 6 grammar
- Create same intermediate AST’s
- So you can mix Perl 5 and Perl 6 code at will
- Now being redeveloped as a proper “**slang**”

So. Nom Nom?

Are you an Early Adopter?

- What are you waiting for?
- Get **rakudobrew**!
- justrakudoit.wordpress.com/2014/05/05/rakudobrew
- Gives you the most recent Rakudo easily!

Do you use Moose with Type Checking?

- Clearly, you’re not too worried about performance
- You value ease of development!
- Then Perl 6 is there for you **now!**
- https://github.com/rakudo/rakudo/
- Word has it, Rakudo Perl 6 on MoarVM is now faster than Perl 5 with Moose (even without Type Checking)

Want a quick intro?

- Learn **Perl 6** in **Y** minutes
- The best thing we have to an intro at the moment
- http://learnxinyminutes.com/docs/perl6/

Perl 6

Perl 6

More of what made Perl 5 great…

…and less of what made Perl 5 grate!

(Courtesy Damian Conway)

Questions?

A State Of Perl 6

The growth of Camelia in the past year or so.

Elizabeth Mattijsen
24 August 2014