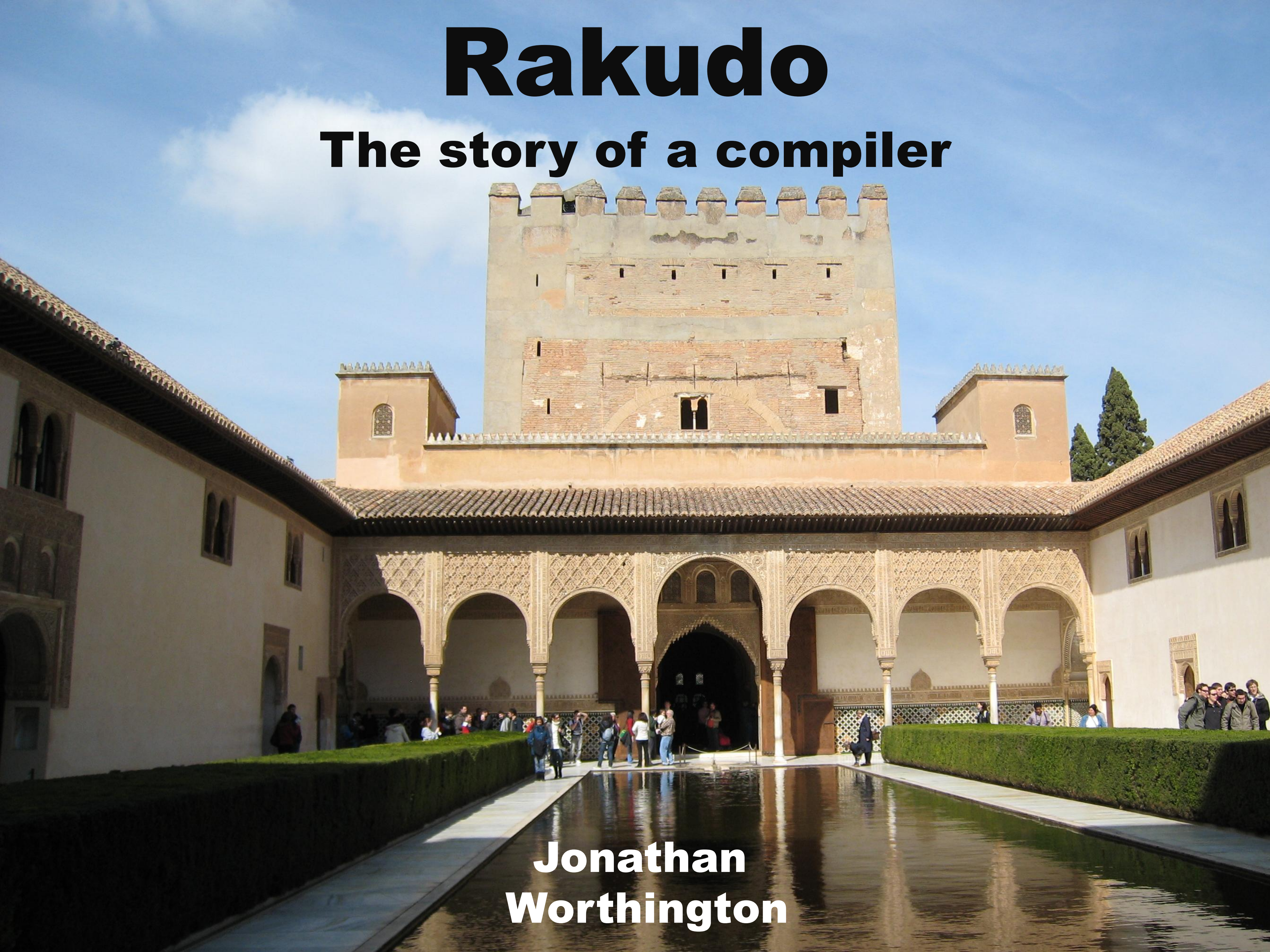
Rakudo

The story of a compiler



Jonathan Worthington

Rakudo Development History

Started by Patrick Michaud around 2005

Not called Rakudo until around January 2008; was just “Perl 6 on Parrot”

Key insight:

Perl 6 should be parsed by Perl 6

Perl 6 should be largely implemented in Perl 6
(or a subset of it)

First Steps

First Steps

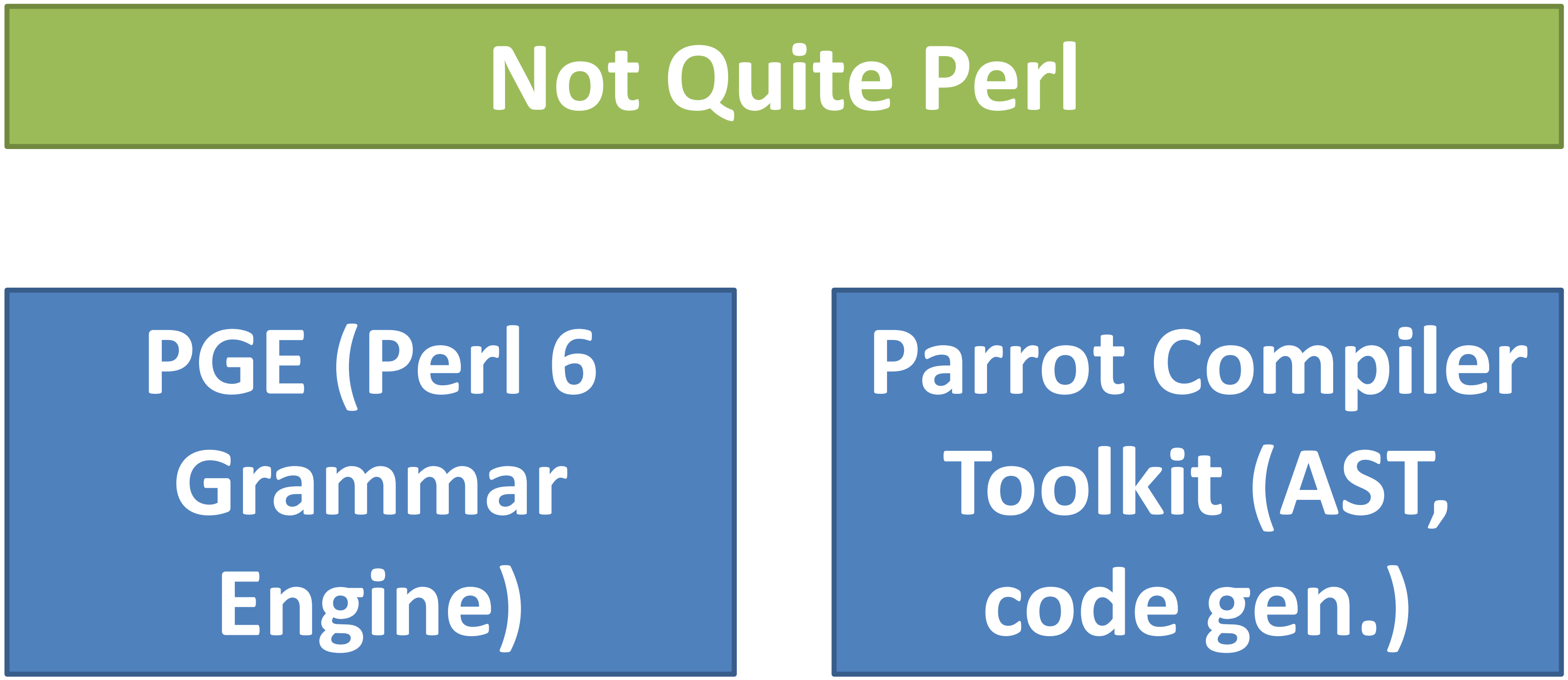
PGE (Perl 6 Grammar Engine)

First Steps

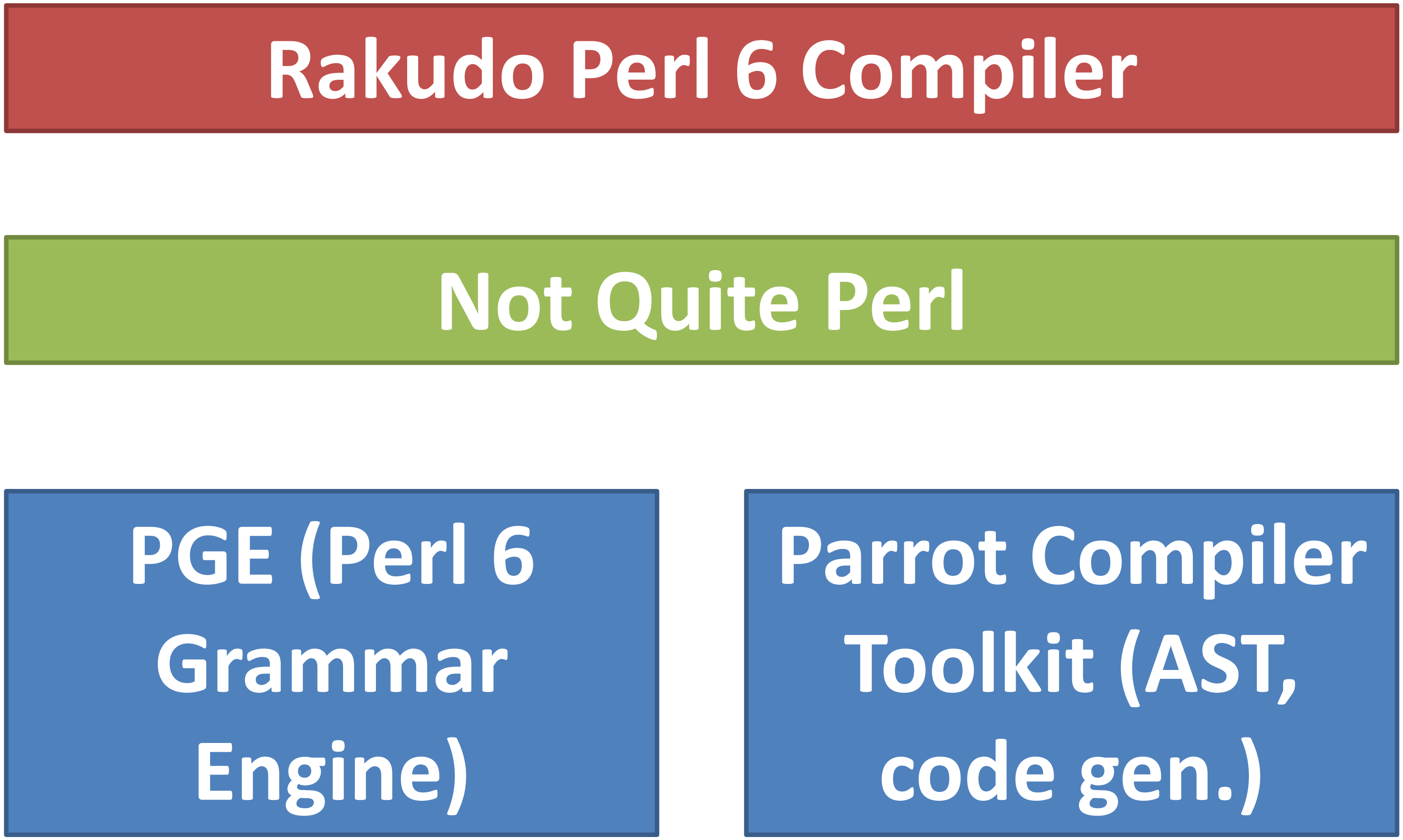
PGE (Perl 6 Grammar Engine)

Parrot Compiler Toolkit (AST, code gen.)

First Steps


Not Quite Perl
PGE (Perl 6 Grammar Engine)
Parrot Compiler Toolkit (AST, code gen.)

First Steps


Rakudo Perl 6 Compiler
Not Quite Perl
PGE (Perl 6 Grammar Engine)
Parrot Compiler Toolkit (AST, code gen.)

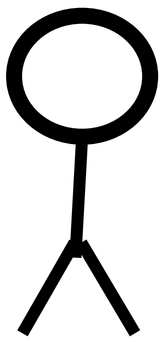
In summer 2008...

I went to OSCON

Met Patrick Michaud for the first time

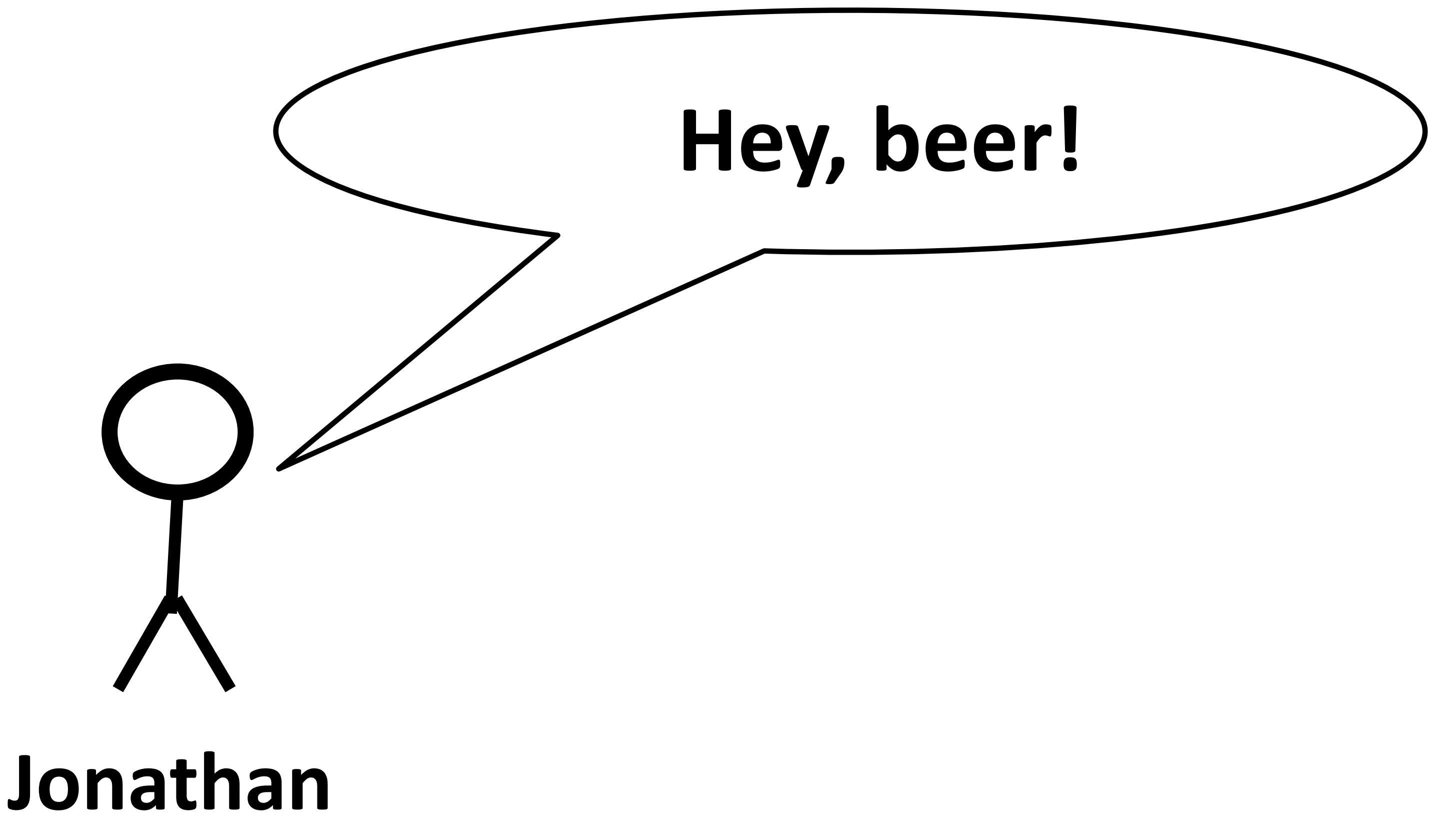
Here’s what happened...

At some party…

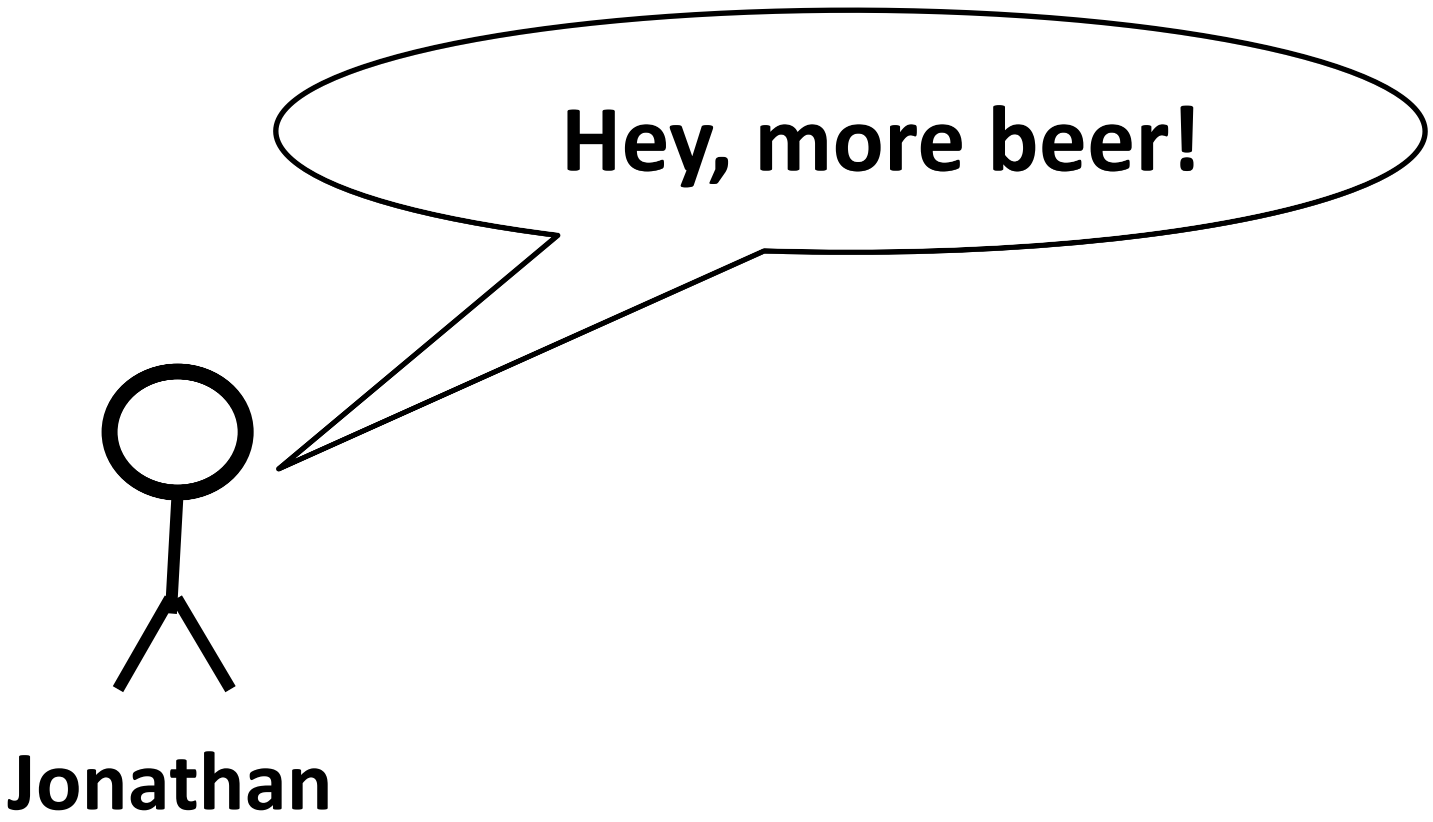


Jonathan

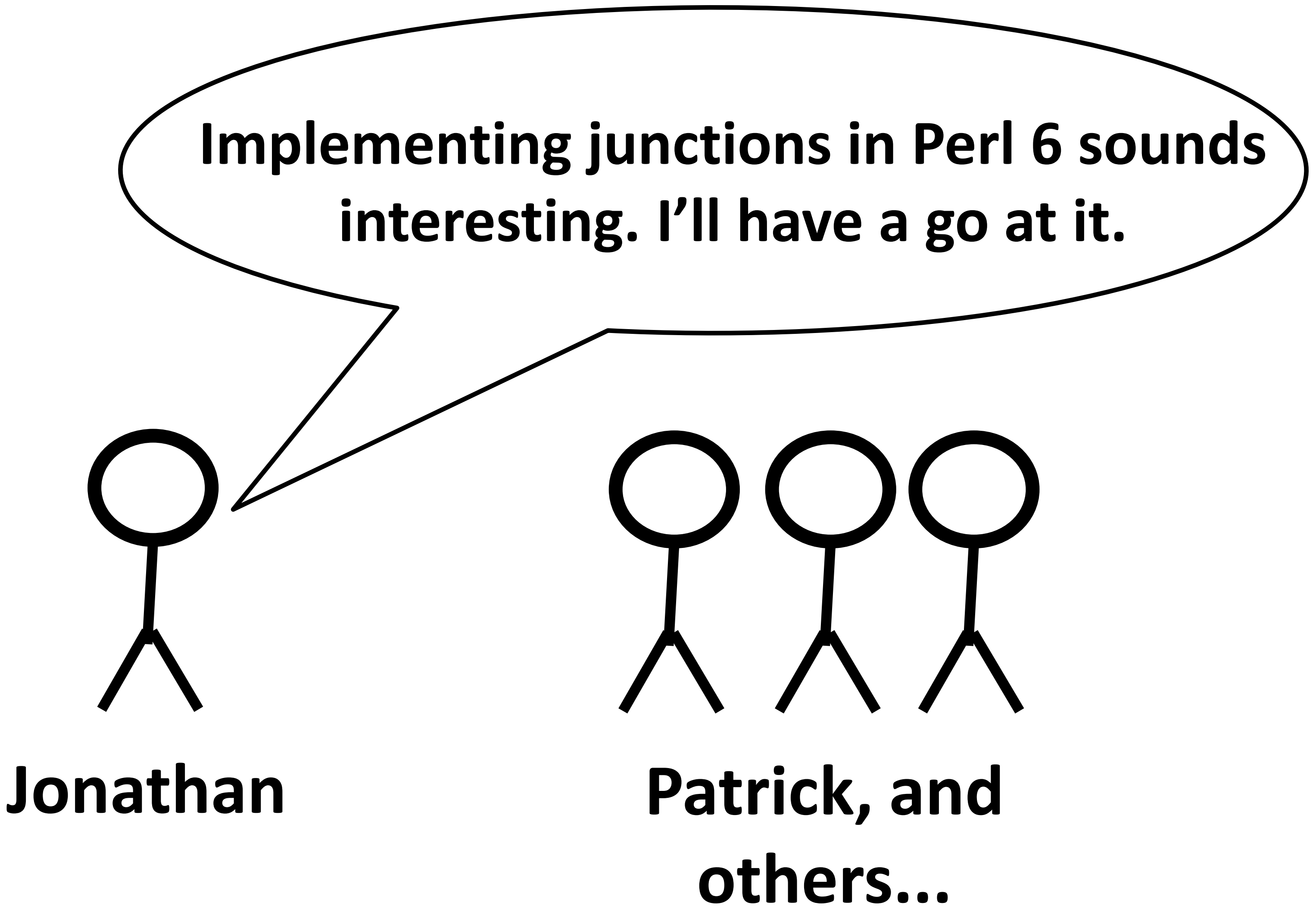
At some party…



At some party…



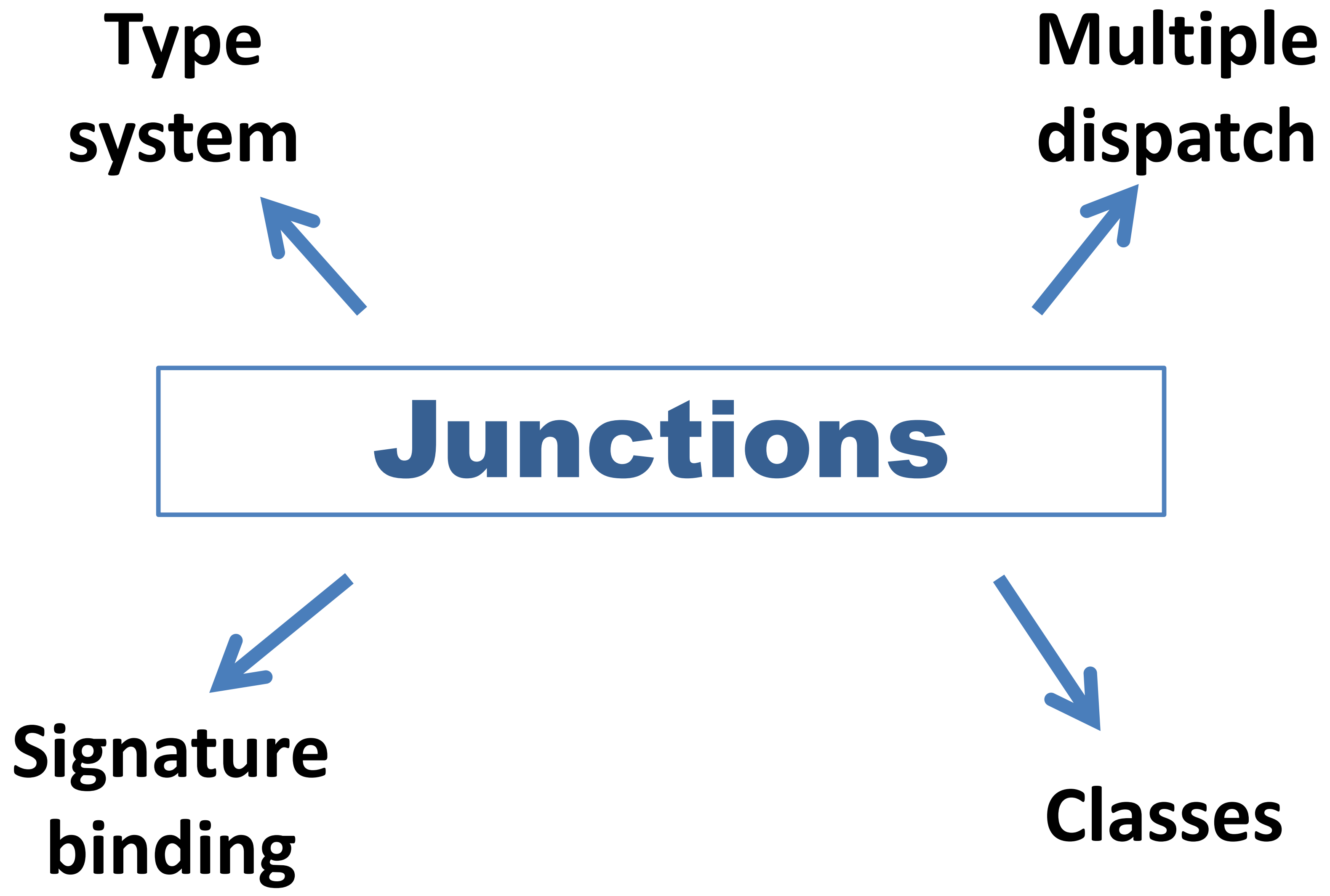
At some party…


Implementing junctions in Perl 6 sounds interesting. I'll have a go at it.
Jonathan
Patrick, and others…

Tip:

If you don’t know how hard it is to implement something…

…be very careful about saying you will do it. ☺


Type system
Multiple dispatch
Junctions
Signature binding
Classes

Hack hack hack...

Over the following year we implemented many, many features.

Good progress, but...

Problems

Significant changes to parsing, thanks to STD arriving

PGE and PCT didn’t integrate so well, creating hard to fix bugs

Complexity was making it hard to make more progress

ng

“next generation”

ng
“next generation”

Parrot Compiler Toolkit
(AST, regex compilation, code generation)

ng
“next generation”

Not Quite Perl
(Now bootstrapping)

Parrot Compiler Toolkit
(AST, regex compilation, code generation)

ng
“next generation”

Rakudo Perl 6 Compiler

Not Quite Perl
(Now bootstrapping)

Parrot Compiler Toolkit
(AST, regex compilation, code generation)

ng

Fixed many long-standing issues

Also was the first time lazy lists were introduced to Rakudo

A lot of progress, but some regressions from “alpha” (the original branch)

Lazy Lists

Really hard to add lazy lists to Perl without surprising people in bad ways

Too lazy → weird action at a distance

Not lazy enough → uses to much memory, or hangs too easily

Lazy Lists

After several designs that failed to work, settled on an immutable iterator model

Resolved the majority of the semantic issues

Initial implementation slow

Rakudo Star

Distributions

Users tend to want more than just a compiler – they want some modules, module installation tools, documentation, etc.

We borrow the notion of “distributions” from Linux

Rakudo Star

Our first series of distribution releases

Aim: attract a wider user base

Separate release schedule and release managers from compiler releases

Did Well On Features

Chained Comparisons Junctions
Classes Signatures Grammars
Perl 6 Regexes Multi-dispatch
Lazy Lists Series Operator
Roles Introspection Traits
Meta-Operators Feeds MAIN
Smart-matching Modules
Native Library Calls Book

Got more...

Users

Modules

Bug Reports

Contributors

But it wasn’t all good news…

Most things run slowly.
Some run glacially slowly.

High memory usage - both base amount and when running

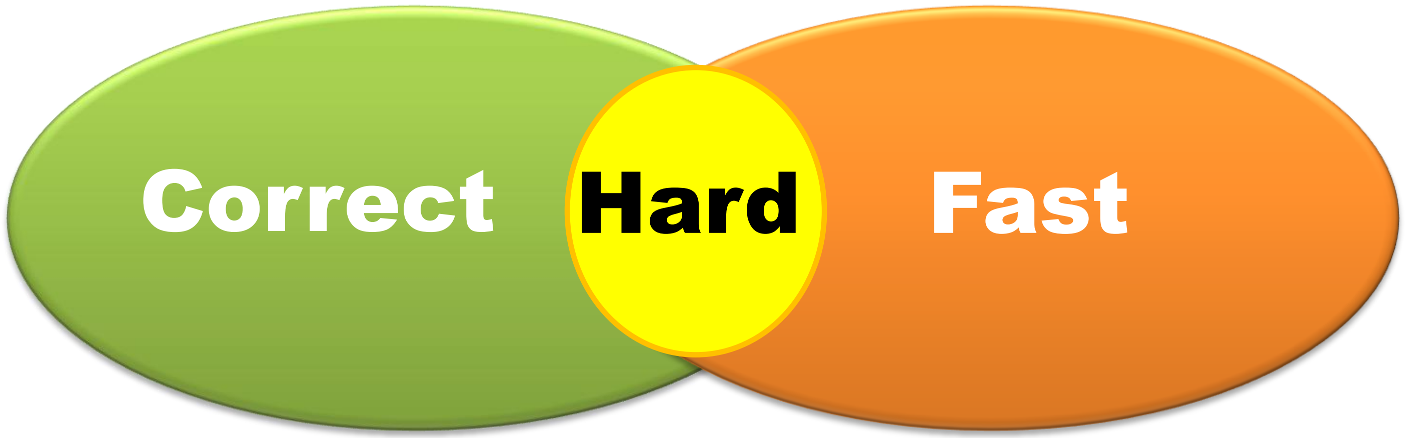
Various unhelpful errors and failure modes

Make it work

THEN

Make it fast

The quick way to implement a feature with the correct semantics is very rarely the optimal one.



Didn't want to waste time making the wrong thing fast.

Now the development focus is changing.

Many implemented features now relatively stable.

Missing features aren't our main adoption blocker, but speed and memory usage are.

Current Work

$$\begin{tikzcd}\textbf{alpha} \arrow[r] & \textbf{ng} \arrow[r] & \textbf{nom}\end{tikzcd}$$

We are here

nom
“new object model”

Replaces the core objects implementations with something that performs far better

Both speed improvements and memory usage reduction

“nom” branch

Rebuild primitives on top of the new object model

In parallel, a big cleanup of the setting and many performance improvements there too

Many more fixes, a few new features

Why is current Rakudo slow?

Various primitives are slow, meaning that everything runs slowly

No optimizer, and not enough information at compile time to write a good one

Performance Improvements

Two sets of performance improvements

First set is just from starting to use the new object model

Second set will come from the optimizer that we will build

Status

Going very well, but still some work to come

Aim to deliver compiler release from “nom” branch in July

Rakudo Star distribution release with it should come in August

Inside Rakudo

Compiler
Grammar, Actions,
Symbol Table,
Module Loader

Metamodel
Classes, roles,
subset types,
OO bootstrap

VM Glue
Signature binder
Multi-dispatcher
Low level guts

CORE.setting
Operators
Built-in classes
Built-in functions

Languages: NQP C Perl 6

The Next Year

Optimizer

Key optimizations:

Statically deciding multi-dispatch

Inlining (especially operators)

Type Inference

Aim to deliver an optimizer by the October release

LTM

Many performance improvements so far have aimed at runtime performance

Longest Token Matching support will make Perl 6 grammars parse faster – meaning that we can parse Perl 6 faster

Bugs and Stability

Focus on fixing bugs, and keeping the bug queue to a reasonable size

Focus on providing a stable platform for developing modules and applications

Backends

Want Rakudo to run on and generate code for multiple VMs

Already some initial work for the .Net CLR / Mono, and for the JVM

Also interest in targeting v8 (Javascript)

Thank you!

Questions?

More on Perl 6: perl6.org

Rakudo: rakudo.org

Blog: 6guts.wordpress.com

Slides: jnthn.net/articles.shtml