Perl is not dead, it is a DEAD END.
This will be a circus of profanity, filled with half truths, baseless rumors and bald face lies, any resemblance to reality is purely coincidental.

I don’t usually rant, and I try to not swear in my talks,.. but this is going to be an exception, so if you are sensitive to swearing or ranting, please leave.
So, this year I will be turning 40 ...
The only issue with this is that I really don’t feel like I am 40, in fact I still feel like I am in my early 20s.
But the fact is, I am getting old, just ask my hairline
given my rapidly advancing age, I have been prone to thinking too much ...
which brings me back to the topic of my talk ... Let me first say for the record that I totally asshat Perl
but I don't love everything about Perl
When I program in other @languages
When I program in other @languages

- which I try to do this as often as $work/$life allows
When I program in other @languages

- which I try to do this as often as $work/$life allows
- and you should too
When I program in other languages

- which I try to do this as often as work/life allows
- and you should too
- if you aren’t you are missing out
When I program in other @languages

• which I try to do this as often as $work/$life allows
  • and you should too
    • if you aren’t you are missing out
      • and perhaps living in a bubble
When I program in other languages

- which I try to do this as often as $work/$life allows
- and you should too
  - if you aren’t you are missing out
  - and perhaps living in a bubble
- sorry, back to my point ...
I always find myself mapping everything back to Perl, no matter how insane the mapping
I always find myself mapping everything back to Perl, no matter how insane the mapping.
Syntax is only skin deep because, after all, ...
Syntax is only skin deep

- If you prick us, do we not bleed?

because, after all, ...
Syntax is only skin deep

• If you prick us, do we not bleed?
• If you tickle us, do we not laugh?

because, after all, ...
Syntax is only skin deep

- If you prick us, do we not bleed?
- If you tickle us, do we not laugh?
- If you poison us, do we not die?

because, after all, ...
So everything I am about to say, I say out of love
So, perl is a dead end ... let me count the ways ...
first lets talk about the community
The Community

first lets talk about the community
The community is a massive ECHO chamber where nothing gets out, seriously, no one cares, they think we are all still writing Perl 4 style sysadmin scripts
but even worse ... nothing gets in, we are a desert island, isolated, alone in the sea ... sure there are people like me with Moose and miyagawa with Plack and a few others who pilfer from other languages, but most people who spend time outside in another language, ... they dont come back! THEY HAVE ESCAPED!
and what is left is a spectacular array of freeks, geeks and weirdos. And yet even as we drive people away, we remain so proud of some of our acheivements ...
But it is not just the user community, it is also the development community. You see p5p, they have this simultaneous fear of change... and this love of bikeshedding. And I believe the combination of these two leads to a culture of obstructionism. Perl is proud not only of the back compat, but also our speed (in certain areas) and our flexibility (in certain areas), and there are groups and factions within p5p which care about one of the other of those things. They will fight for their reasons, which may be contrary to other groups. The result, bikeshedding. ... let it finish, ... let it finish ... and so bikeshedding is bad mkay. It leaves us with lots of old technical debt,.. speaking of technical debt ..
But it is not just the user community, it is also the development community. You see p5p, they have this simultaneous fear of change... and this love of bikeshedding. And I believe the combination of these two leads to a culture of obstructionism. Perl is proud not only of the back compat, but also our speed (in certain areas) and our flexibility (in certain areas), and there are groups and factions within p5p which care about one of the other of those things. They will fight for their reasons, which may be contrary to other groups. The result, bikeshedding. ... let it finish, ... let it finish ... and so bikeshedding is bad mkay. It leaves us with lots of old technical debt,.. speaking of technical debt ...
Here is a perfect example of our isolation and refusal to accept the real world
Really?!?!

really?!?!
are we still caring about CGI?
CG - fucking - I ... other languages, they have moved on
Java
Java

- Problem
Java

• Problem
  • JVM is a memory hog
Java

- Problem
  - JVM is a memory hog
  - JVM is slow to start up
Java

- Problem
  - JVM is a memory hog
  - JVM is slow to start up
- Solution
Java

• Problem
  • JVM is a memory hog
  • JVM is slow to start up

• Solution
  • Build expensive middleware products
Java

• Problem
  • JVM is a memory hog
  • JVM is slow to start up

• Solution
  • Build expensive middleware products
  • problem (mostly) solved
Ruby
Ruby

• Problem
Ruby

• Problem
  • the interpreter is slow
Ruby

- Problem
  - the interpreter is slow
  - OMG, my RoR app takes forever to load
Ruby

- Problem
  - the interpreter is slow
  - OMG, my RoR app takes forever to load
- Solution
Ruby

- Problem
  - the interpreter is slow
  - OMG, my RoR app takes forever to load
- Solution
  - embrace the ancient art of FastCGI
Ruby

• Problem
  • the interpreter is slow
  • OMG, my RoR app takes forever to load

• Solution
  • embrace the ancient art of FastCGI
  • problem (mostly) solved
Node.JS
Node.js

- Problem
Node.JS

- Problem
  - there is no problem
Node.JS

- Problem
  - there is no problem
  - semicolons
Node.JS

- Problem
  - there is no problem
  - semicolons

- Solution
Node.JS

• Problem
  • there is no problem
  • semicolons

• Solution
  • there is no problem
Node.JS

• Problem
  • there is no problem
  • semicolons

• Solution
  • there is no problem
  • problem (mostly) solved
Perl
Perl

• Problem
Perl

- Problem
- WE MUST SUPPORT CGI
Perl

• Problem
  • WE MUST SUPPORT CGI
  • This is why we can’t have nice things!
Perl

• Problem

• WE MUST SUPPORT CGI

• This is why we can’t have nice things!

• Solution
Perl

• Problem
• WE MUST SUPPORT CGI
• This is why we can’t have nice things!

• Solution
• move the fuck on!
Perl

• Problem

• WE MUST SUPPORT CGI

• This is why we can’t have nice things!

• Solution

• move the fuck on!

• really!
Perl

• Problem
  • WE MUST SUPPORT CGI
  • This is why we can’t have nice things!

• Solution
  • move the fuck on!
    • really!
      • no seriously, move on!
Seriously People!
Perl is a great language, it can be used to build big systems. People are doing it with Ruby, with Python, shit there are a bunch of hipsters out there doing it in Node.JS (see also: meteor). Perl has a bad rap on this which in some ways is undeserved, but ... it is not entirely undeserved. Which brings me to my next set of slides ...
Any serious “enterprise” scale application will need to do some kind of multitasking. Unfortunately for us, the history of Perl and threading is pretty bad, iThreads were a complete mess.
Processes

We got by with processes for a while, and since OS procs improved, we were okay
We are behind

but at this point we are behind, really behind, ...
We are behind

- really behind

but at this point we are behind, really behind, ...
We are behind

• really behind
• really really behind

but at this point we are behind, really behind, ...
We are behind

• really behind

• really really behind

• really really really really really really really behindly

but at this point we are behind, really behind, ...
We are behind

• really behind
• really really behind
  • really really really rea behindlly
    • really rebehallyind rrealeally behirellay

but at this point we are behind, really behind, ...
We are behind

- really behind
- really really behind
  - really really really behindly
    - really rebehallyind rreaaleally behirellay
  - oh fuck it, if you don't get the joke

but at this point we are behind, really behind, ...
We are behind

- really behind
- really really behind
  - really really really behind!
  - really rebehallyind rrealeally behirellay
  - oh fuck it, if you don't get the joke
  - then you've illustrated my point exactly!

but at this point we are behind, really behind, ...
Don’t think I am forgetting about events, but the reality is that event based programming is not the same as true threading. POE was great, and in many ways still is, it was nominated the best module of 1999, it is now 2012 (sorry Rocco, but I don’t think I am telling you anything you don’t already realize yourself).
Tonight I'm gonna party like it's 1999

Don’t think I am forgetting about events, but the reality is that event based programming is not the same as true threading. POE was great, and in many ways still is, it was nominated the best module of 1999, it is now 2012 (sorry Rocco, but I don’t think I am telling you anything you don’t already realize yourself).
So where do we go from here? Cause a lot of thinking and work has been going into this area of programming for a while now. We need to leapfrog ...
Well, Rocco has a new project called Reflex. It is pretty cool, all about Functional Reactive programming, I highly recommend you check it out. That said, it is still a work in progress and while being built on POE, it does not yet have the stability of POE. And, it still isn’t true threads ... also, little known fact,.. <space> that is a young Rocco
Well, Rocco has a new project called Reflex. It is pretty cool, all about Functional Reactive programming, I highly recommend you check it out. That said, it is still a work in progress and while being built on POE, it does not yet have the stability of POE. And, it still isn’t true threads ... also, little known fact,.. <space> that is a young Rocco
Then there is AnyEvent, *sigh*, AnyEvent. It is actually not a bad API (not great, but not bad), it doesn’t provide real threads, but with Coro is gets fairly close. But I can’t honestly recommend it, <space> because as a project, it is so surrounded by controversy and bitterness that its stability is questionable at best.
Then there is AnyEvent, *sigh*, AnyEvent. It is actually not a bad API (not great, but not bad), it doesn’t provide real threads, but with Coro is gets fairly close. But I can’t honestly recommend it, because as a project, it is so surrounded by controversy and bitterness that its stability is questionable at best.
Now there are some other modules out there, IO::Async is a popular choice of late, and there is a secret cabal working on what they called “the Plack of Event Programming”. But the reality of the situation is that we are not addressing a base need ... which is decent language level support for multiprogramming
What do we want?

Now there are some other modules out there, IO::Async is a popular choice of late, and there is a secret cabal working on what they called “the Plack of Event Programming”. But the reality of the situation is that we are not addressing a base need ... which is decent language level support for multiprogramming
What do we want?

Interpreter level threading!

Now there are some other modules out there, IO::Async is a popular choice of late, and there is a secret cabal working on what they called “the Plack of Event Programming”. But the reality of the situation is that we are not addressing a base need ... which is decent language level support for multiprogramming.
What do we want?
Interpreter level threading!
When do we want it?

Now there are some other modules out there, IO::Async is a popular choice of late, and there is a secret cabal working on what they called “the Plack of Event Programming”. But the reality of the situation is that we are not addressing a base need ... which is decent language level support for multiprogramming
What do we want?
Interpreter level threading!

When do we want it?

hmm, like 5-10 years ago actually, but at this point really I would take anything, what have you got?

Now there are some other modules out there, IO::Async is a popular choice of late, and there is a secret cabal working on what they called “the Plack of Event Programming”. But the reality of the situation is that we are not addressing a base need ... which is decent language level support for multiprogramming
speaking of being behind the times, let's talk about @_ and subroutine signatures, ... do you know any other language that doesn't have proper subroutine signatures? Let's stop for a second and examine the 3 oldest programming languages out there.
FORTRAN has had them since at least the late 60s (FORTRAN 66), hell they didn’t even have keyboards, but they had subroutine signatures.
COBOL, the people who gave us the year 19100 had parameters, they were clunky and awkward, but it had parameters.
Of course LISP had them, its lambdas all the way down.
PURL, Y U NO HAZ

REAL SUBROUTINE SIGNATURES?!WTF
Oh yeah, backwards compatibility and speed concerns... ain't that a bitch
Now let's talk about OOP
<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larry</td>
<td>5.000beta3</td>
<td>1994-Sep-??</td>
</tr>
<tr>
<td>Andy</td>
<td>5.000b3a</td>
<td>1994-Sep-18</td>
</tr>
<tr>
<td></td>
<td>5.000b3b</td>
<td>1994-Sep-22</td>
</tr>
<tr>
<td></td>
<td>5.000b3c</td>
<td>1994-Sep-23</td>
</tr>
<tr>
<td></td>
<td>5.000b3d</td>
<td>1994-Sep-27</td>
</tr>
<tr>
<td></td>
<td>5.000b3e</td>
<td>1994-Sep-28</td>
</tr>
<tr>
<td></td>
<td>5.000b3f</td>
<td>1994-Sep-30</td>
</tr>
<tr>
<td></td>
<td>5.000b3g</td>
<td>1994-Oct-04</td>
</tr>
<tr>
<td>Andy</td>
<td>5.000b3h</td>
<td>1994-Oct-07</td>
</tr>
<tr>
<td>Larry?</td>
<td>5.000gamma</td>
<td>1994-Oct-13?</td>
</tr>
<tr>
<td>Larry</td>
<td>5.000</td>
<td>1994-Oct-17</td>
</tr>
<tr>
<td>Andy</td>
<td>5.000a</td>
<td>1994-Dec-19</td>
</tr>
<tr>
<td></td>
<td>5.000b</td>
<td>1995-Jan-18</td>
</tr>
<tr>
<td></td>
<td>5.000c</td>
<td>1995-Jan-18</td>
</tr>
</tbody>
</table>

Let's take a quick look here at the history of OOP in Perl. Perl 5 is when we got objects, that was in Oct of 1994. That is almost almost 18 years ago ...
Now I want to put some perspective in place here, in 1994, Java was not yet in Alpha. It wasn’t until 1996 (little over one year later) that the JDK hit 1.0, and then not until 1997 (about 2.5 years) before the JDK 1.1 came out (which is largely considered the first usable one). Keep this in mind, I will come back to this ...
Now I want to put some perspective in place here, in 1994, Java was not yet in Alpha. It wasn’t until 1996 (little over one year later) that the JDK hit 1.0, and then not until 1997 (about 2.5 years) before the JDK 1.1 came out (which is largely considered the first usable one). Keep this in mind, I will come back to this ...
Now I want to put some perspective in place here, in 1994, Java was not yet in Alpha. It wasn’t until 1996 (little over one year later) that the JDK hit 1.0, and then not until 1997 (about 2.5 years) before the JDK 1.1 came out (which is largely considered the first usable one). Keep this in mind, I will come back to this ...
Now I want to put some perspective in place here, in 1994, Java was not yet in Alpha. It wasn’t until 1996 (little over one year later) that the JDK hit 1.0, and then not until 1997 (about 2.5 years) before the JDK 1.1 came out (which is largely considered the first usable one). Keep this in mind, I will come back to this ...
Perl OOP feature additions

Now you would think, given that we were there early on, we would be innovating and evolving and really pushing the boundaries of OOP. So I have compiled a list of new OOP related features that Perl has gotten since 1994.
Perl OOP feature additions

- mro

Now you would think, given that we were there early on, we would be innovating and evolving and really pushing the boundaries of OOP. So I have compiled a list of new OOP related features that Perl has gotten since 1994.
Perl OOP feature additions

• mro

• fieldhashes

Now you would think, given that we were there early on, we would be innovating and evolving and really pushing the boundaries of OOP. So I have compiled a list of new OOP related features that Perl has gotten since 1994.
Java OOP feature additions

• inner classes
• reflection API
• RMI (remote method invocation)
• generics (type parameterization)
• annotations
• autoboxing
• enumerations

Now in contrast, here is Java. Keep in mind that Java is not even that dynamic a language in terms of new features and changes, its pretty slow to move, but here are some of the new features that came out since the initial release that specifically enhance the OOP experience. And keep in mind that many other releases built on and improved these features over the years as well.
Perl OOP on the CPAN

Now, wait a second you say, we do our OOP innovation on the CPAN, ... <space> yup, yup we do ...
Perl OOP on the CPAN

Now, wait a second you say, we do our OOP innovation on the CPAN, ... <space> yup, yup we do ...
About a year ago we started a project to try and come up with a MOP for Perl 5 that could be added to the core.
(The project is kinda on ice right now, if you want to know why, come buy me a beer and I will tell you.)
Next,... CPAN,... horray!
But we have CPAN!!
But we have CPAN!!

- "Perl’s killer app"
But we have CPAN!!

- "Perl’s killer app"
- "Perl is not dead, just look at CPAN"
But we have CPAN!!

- "Perl’s killer app"
- "Perl is not dead, just look at CPAN"
- "Python/Ruby may be cool, but they don't have CPAN"
But we have CPAN!!

- "Perl’s killer app"
- "Perl is not dead, just look at CPAN"
- "Python/Ruby may be cool, but they don't have CPAN"
- "I will give up Perl when they pry it from my cold dead CPAN prompt"
I call bullshit, ... if you still believe that CPAN is why Perl is still relevant, then you have clearly not played around with other systems.
Don’t be fooled

- Gems (Ruby)
- PyPI (Python)
- npm (Node.JS)
- Nuget (.NET)
- Hackage (Haskell)

There are other nice systems out there, sure they might not have the number of modules (although how much of CPAN is pure shit, really), and they might not have the infrastructure (testers, etc), it is only a matter of time until they do. To put it simply, we’ve made plenty of mistakes, so they don’t have to.
So, is there no hope?
For the record, I think Perl 6 is the future of Perl.
but, what worries me is the hole that Perl 6 must climb out of
but, what worries me is the hole that Perl 6 must climb out of
but why do I think this? well, ... because of the Perl 5 core
Hey, how would you like to maintain a 17 year old pile of very complex C code written by multiple authors and which has to maintain bug for bug compatibility with any random 25 year old codebases?

The old code base is just too insane and unwieldy. At this point there are maybe 2 people (Nicholas Clark and Dave Mitchell (perhaps Father C)) who know the whole thing, after that it another maybe 10–15 people who can hack certain portions of the code. But seriously ...
Hey, how would you like to maintain a 17 year old pile of very complex C code written by multiple authors and which has to maintain bug for bug compatibility with any random 25 year old codebases?

Oh, and did i mention? You won't be getting paid! (at least not very much)

The old code base is just too insane and unwieldy. At this point there are maybe 2 people (Nicholas Clark and Dave Mitchell (perhaps Father C)) who know the whole thing, after that it another maybe 10–15 people who can hack certain portions of the code. But seriously ...
WHO IN THEIR RIGHT FUKING MIND WOULD WANT TO MAINTAIN THE PERL 5 CORE ?!!?
So is 6 our only hope?
I believe we are at a crossroads, in one direction, Perl 5 (death and irrelevance), the other is PHP/Ruby/Python/C#/Java/etc. (another job, but no more Perl culture/community), and the third way is Perl 6 (ready by christmas). And I am not alone ...
11:44 <Nicholas> please could I be encouraging you to get people to fork perl

11:44 <Nicholas> and demonstrate how easy it is to do it better :-)

11:45 <Nicholas> although I have to assume that this isn't what your talk is about, and I shall have to wait and see

11:45 <Nicholas> I, genuinely, would be very curious to see what happened to a(nother) fork of Perl 5

This was a discussion I had with Nicholas Clark earlier this afternoon after he found my talk abstract on the Perl Oasis site.
fork of perl
fork of perl
rewrite of perl
My Perl Wishlist

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

- A more consistent syntax and sane grammar
  - with a real AST

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar
  • with a real AST
  • and better tooling support

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar
  • with a real AST
  • and better tooling support
• A less insane runtime

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime
  - proper MOP (everything is an object)

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar
  • with a real AST
  • and better tooling support
• A less insane runtime
  • proper MOP (everything is an object)
  • slimmer core

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar
  • with a real AST
  • and better tooling support
• A less insane runtime
  • proper MOP (everything is an object)
  • slimmer core
  • easy (non-XS) extension mechanism

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar
  • with a real AST
  • and better tooling support
• A less insane runtime
  • proper MOP (everything is an object)
  • slimmer core
  • easy (non-XS) extension mechanism
• On a modern VM platform

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar
  • with a real AST
  • and better tooling support
• A less insane runtime
  • proper MOP (everything is an object)
  • slimmer core
  • easy (non-XS) extension mechanism
• On a modern VM platform
  • JVM / CLR / LLVM / V8

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
My Perl Wishlist

• A more consistent syntax and sane grammar
  • with a real AST
  • and better tooling support
• A less insane runtime
  • proper MOP (everything is an object)
  • slimmer core
  • easy (non-XS) extension mechanism
• On a modern VM platform
  • JVM / CLR / LLVM / V8
  • cross language sharing

Note that this is pretty much what Perl 6 is promising, the difference is that I want to do it for Perl 5. Perl 6 is a space age language from the future, I want (we as a community need) an upgrade for our existing language.
Which brings me to ... moe
Well, I originally was gonna call it Larry, ... after... well, Larry, ... but when I asked him if he felt that was okay, he was not so sure (I don’t blame him) and he suggested either Curly or maybe Moe. And since Moe sounds so much like Mo and Ingy released at least two modules that sound like Moose, I felt this was a pretty good trollback.
What (the fuck) is Moe?

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

• A more consistent syntax and sane grammar
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
- with a real AST

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime
  - proper MOP (everything is an object)

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime
  - proper MOP (everything is an object)
  - slimmer core

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime
  - proper MOP (everything is an object)
  - slimmer core
  - easy (non-XS) extension mechanism

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime
  - proper MOP (everything is an object)
  - slimmer core
  - easy (non-XS) extension mechanism
- On a modern VM platform

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime
  - proper MOP (everything is an object)
  - slimmer core
  - easy (non-XS) extension mechanism
- On a modern VM platform
  - JVM / CLR / LLVM / V8

What is Moe you ask, let me tell you ... (sound familiar)
What (the fuck) is Moe?

- A more consistent syntax and sane grammar
  - with a real AST
  - and better tooling support
- A less insane runtime
  - proper MOP (everything is an object)
  - slimmer core
  - easy (non-XS) extension mechanism
- On a modern VM platform
  - JVM / CLR / LLVM / V8
  - cross language sharing

What is Moe you ask, let me tell you ... (sound familiar)
Wait, WTF?!!?!?

yes, thats right, I want to prototype a new version of Perl 5
My first thought was to try and write it in Perl 5, but I don’t know if you’ve heard, its a dead language. Also, it doesn’t have all the features I want (see also: this talk)
My first thought was to try and write it in Perl 5, but I don’t know if you’ve heard, its a dead language. Also, it doesn’t have all the features I want (see also: this talk)
My next thought was to use a nice FP language since they are so nice to write compilers in, like OCaml or Standard ML, but I think on some level that was a mistake that was made in the Pugs project. Haskell was just not accessible enough as a language.
My next thought was to use a nice FP language since they are so nice to write compilers in, like OCaml or Standard ML, but I think on some level that was a mistake that was made in the Pugs project. Haskell was just not accessible enough as a language.
> node perl.js -e 'print join " " => "Hello", "World";'

Then I thought, hey, Javascript is all the rage for runtimes ... <space> but honestly, I wanted to write this in something strongly typed
> node perl.js -e 'print join " " => "Hello", "World";'

throws JShirleyUnimpressedException at <eval> line 0

Then I thought, hey, Javascript is all the rage for runtimes ... but honestly, I wanted to write this in something strongly typed
Then I thought about Java, which has it's pros and its cons. And I actually made some headway on a version in Java. It had a grammar in Antlr and a semi complete runtime, but ultimately I ran into a wall on this one ...
Yo Dawg, I heard you like dead languages so I implemented your dead language in a dead language so you can program a dead language while you program your dead language.

Then I thought about Java, which has its pros and its cons. And I actually made some headway on a version in Java. It had a grammar in Antlr and a semi complete runtime, but ultimately I ran into a wall on this one ...
<stevan>: auuh fuck it, I am gonna write this shit in Scala
<stevan>: Java's type system is just a bitch
<genehack>: somewhere, a gphat just got its wings

And you know what, Scala is actually pretty Perlish (in that it allows you to shoot off your entire leg if you are not careful).
And you know what, Scala is actually pretty Perlish (in that it allows you to shoot off your entire leg if you are not careful).
seriously?!?!?

so yeah, seems a little crazy, but hey, it just might work. If nothing more it might produce some kind of spec/test-suite/fire-under-our-collective-asses. And this is not something new,.. in fact, I’ve been talking about this with a number of people for a while now ...
I’ve been discussing something like this with Matt Trout for years ... though he might not remember (you know, short term memory loss and all)
And I know I’ve been talking with Yuval Kogman about this for even longer then that ...
Jesse Luehrs, while doing his best to be un-impressed, has (if given enough beers) been known to listen to my rant on the subject ...
And I know Shawn Moore, if given a pair of sunglasses and enough cider, would also agree with me ...
and so, I want your help too, ... Moe is –Ofun, very much taking inspiration from Pugs, Moe is a place to experiment with what a Modern Perl 5 might look like.
http://github.com/stevan/moe

and, if you go to ... wait one second (actually go push code),... you will see ...
Really?!????1?!
Really?!????1?!?

- Yes, seriously!
Really?!????1?! 

- Yes, seriously!
- Come hack!!
Really?!????1?! 

- Yes, seriously!
- Come hack!!
- this is all about -OFun
Really?!??!!1?!?

• Yes, seriously!
• Come hack!!
• this is all about -OFun
  • Pugs Inspired
Really?!????1?! 

- Yes, seriously!
- Come hack!!
- this is all about -OFun
  - Pugs Inspired
- Learn a useful skill (Scala)
Really?!????1?! 

- Yes, seriously!
- Come hack!!
- this is all about -OFun
  - Pugs Inspired
- Learn a useful skill (Scala)
  - Twitter is hiring (I think, ask gphat)
Really?!????1?!?

• Yes, seriously!
• Come hack!!
• this is all about -OFun
  • Pugs Inspired
• Learn a useful skill (Scala)
  • Twitter is hiring (I think, ask gphat)
• Lets build the new foundation!