Worst. Ideas. Evar.

TODO: Subtitle

Scope

- We're not here to trash on other ppl's projects.
- Not a worst practices show and tell.
- Well engineered, but horrible ideas.

Bios

- We both work at AT&T Interactive.
- We've done ruby for a long time.
- Blah blah blah boring...

Bios

- We both work at AT&T Interactive.
- We've done ruby for a long time.
- Blah blah blah boring...

NOTE:
<u>none</u> of this presentation or software was supported, sanctioned, endorsed, or even tolerated by our employer.

AT&T Interactive: Please don't fire us!

Aaron Patterson

- Likes kittens, ponies, and vim.
- Is totally in love with his mustache.
- Prefers to write C code over ruby. Think about it:
    - nokogiri? C
    - johnson? C
    - never_say_die? C
    - psych? C
    - phuby? C
    - nfc? C
    - earworm? C
    - qrtools? C

Ryan Davis

- Likes kittens, ponies, and emacs.
- Is totally in love with his ponytail.
- Wishes Sting never left the Police.
- Fears nothing except:
  - mushrooms
  - asparagus
- Hates it when I sing.
- Same colors, everyday.

Redacted

What is a Bad Idea?

Meta: These Slides

Not These Slides

These Slides

Field Guide

Spotting bad ideas in the wild

- <u>W</u>ell engineered & tested.
- <u>U</u>seless-ish.
- <u>P</u>oe's Law.
- <u>S</u>piral nature.

WUPS

Field Guide

Spotting bad ideas in the wild

- W̲ell engineered & tested.
- U̲seless-ish.
- P̲oe's Law.
- S̲piral nature.

Well Engineered & Tested

W U P S

- Bad Ideas != Bad Code.
- Good Code + Tests > Horrible Idea.
- Be wary of the well tested project!

Useless-ish

- They must be useless… ish.
- They should do something.
- Just not anything you need/want.

Poe's Law

W
U
P
S

"Without a winking smiley or other blatant display of humour, it is impossible to create a parody of fundamentalism that someone won't mistake for the real thing," [1]

"...it is hard to tell parodies of fundamentalism from the real thing, since they both seem equally insane. Conversely, real fundamentalism can easily be mistaken for a parody of fundamentalism." [2]

[1] http://en.wikipedia.org/wiki/Poe's_law
[2] http://rationalwiki.com/wiki/Poe's_Law
[awesome] http://conservapedia.com/Poe's_law

Poe's Law

- From high level, they can sound perfectly reasonable.
- Solution looking for a problem.
  - They don't solve any immediate problems at hand. They generate them.
- Everyone always ignores that guy in the corner asking "Really???". I mean, c'mon... what does he know?

Spiral Nature

- The best part of Bad Ideas is that they build upon one another.
- Hopefully with cyclic dependencies.

Hypothetical Examples

(Read: We haven't finished them... yet)

- XML multi file format (a la java's jar format)
- DRB over RFID
- DRB over QR/Bar Code via webcams
- Assembly optimized web pages
- FFI

Yoda

Yoda

- Bad ideas can be high level.
- Test frameworks are the new IRC bot!
- Yoda defines a spec language in the direction we think they should be.

```
Bowling.yoda {
  "score 0 for gutter game".it_will {
    bowling = Bowling.new
    20.times { bowling.hit(0) }

    bowling.score 0.it_is?
    bowling.score 42.it_is_not!
  }
}
```

```
./lib/yoda.rb:5:in `fail_me': Fail me you
did: 1 != 0 (Yoda::FailMe)
   from ./lib/yoda.rb:19:in `matches'
   from (eval):5:in `score'
   from example.rb:19
   from ./lib/yoda.rb:45:in `it_will'
   from example.rb:15
   from ./lib/yoda.rb:63:in `yoda'
   from example.rb:14
```

So Simple It Can't Break

- 65 lines of Jedi Master Ruby.
- 3 conditionals, 1 loop.
- Flogs to 41.1, avg 4.1 / method. LOW!
- How can you go wrong?

Useless, am I?

- Not exactly very expressive.
- Then again, doesn't need to be.

Poe's got nothin' on Yoda

- Really? Another test framework?
- Really?

The Force is Everywhere

W
U
P
S

- It is a test framework...
- You can use it **everywhere**!

Wilson

Ruby is SLOW

C is not!

But, why write in C?

When you can write in Assembly?

Wilson

- Bad ideas can be very low level.
- Generates x86 machine code via a "natural" feeling ruby DSL.
- Named after the very metal Wilson Bilkovich.

Inline-C

```
class Counter
  inline do |builder|
    builder.c "
      long cee(int n) {
        long i;
        for (i = 0;i<n+1;i++) {};
        return i;
      }"
  end
end
```

Wilsasm

```
class Counter
  defasm :asm2, :n do
    eax.xor eax

    edx.mov arg(0)
    from_ruby edx
    edx.inc

    count = self.label
    eax.inc
    eax.cmp edx
    jnz count

    to_ruby eax
  end
end
```

Wilsasm

```
class Counter
  defasm :asm2, :n do
    eax.xor eax

    edx.mov arg(0)
    from_ruby edx
    edx.inc

    count = self.label
    eax.inc
    eax.cmp edx
    jnz count

    to_ruby eax
  end
end
```

Benchmarks!

```
% rm -r ~/.ruby_inline; ./bench.rb 1_000_000 1_000
# of iterations = 1000000
$n = 1000
                          user     system      total        real
null_time             0.120000   0.000000   0.120000 (  0.122507)
cee_nil               0.280000   0.000000   0.280000 (  0.279552)
asm_nil               0.280000   0.000000   0.280000 (  0.275498)
ruby_nil              0.370000   0.000000   0.370000 (  0.372142)
cee                   0.830000   0.010000   0.840000 (  0.837607)
asm2                  0.830000   0.000000   0.830000 (  0.839430)
asm                   3.520000   0.000000   3.520000 (  3.542521)
ruby                 98.970000   0.430000  99.400000 (101.903256)
```

Whip-Smart

- Generates machine code directly:
  - No dependencies.
  - No external resources.
  - Parses the 60 page x86 spec into instructions and their opcodes.

"Uses"

- Good at writing really fast code.
- And crashing. Fast.
- I don't <u>think</u> anyone uses this.
  - (I hope not)

Poe Man's Dispatch

W
U
P
S

- The original intent was to write a pure ruby method dispatch function.
  - But that is hard.
    - And we got laid off of Rubinius.

WUPS

Spiral?

- It was <u>intended</u> to be the core of a new ruby implementation.
- How much more spiral do you want?

Wank

Wank

- Human Readable Marshal Format

Why Wank?

Marshal data is too hard to read

```
Marshal.dump nil # => "\004\b0"
```

Unreadable,
therefore useless

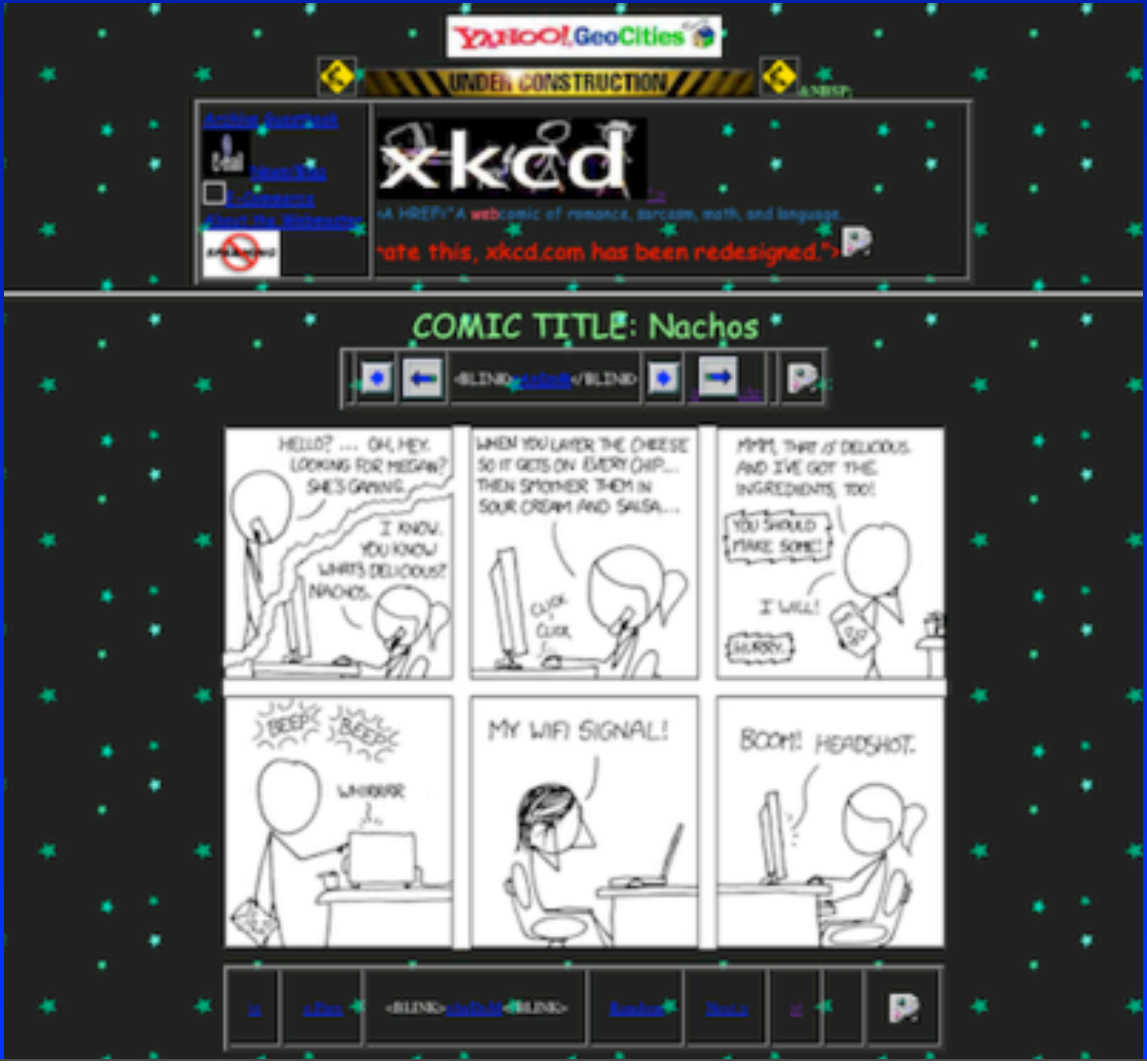
YAML is too hard to read

```
YAML.dump nil # => "--- \n"
```

Unreadable,
therefore useless

Websites are Readable

Exhibit A


YAHOO! GeoCities
UNDER CONSTRUCTION
&NBSP;
xkcd
<A HREF="A webcomic of romance, sarcasm, math, and language.
ate this, xkcd.com has been redesigned.">
COMIC TITLE: Nachos
<BLINK>[illegible]</BLINK>
HELLO? ... OH, HEY. LOOKING FOR MEGAN? SHE'S GAMING.
I KNOW. YOU KNOW WHAT'S DELICIOUS? NACHOS.
WHEN YOU LAYER THE CHEESE SO IT GETS ON EVERY CHIP... THEN SMOTHER THEM IN SOUR CREAM AND SALSA...
CLICK CLICK
MMM, THAT IS DELICIOUS. AND I'VE GOT THE INGREDIENTS, TOO!
YOU SHOULD MAKE SOME!
I WILL!
HURRY.
BEEP BEEP
WHIRRR
MY WIFI SIGNAL!
BOOM! HEADSHOT.
<BLINK>[illegible]</BLINK>

The Guts

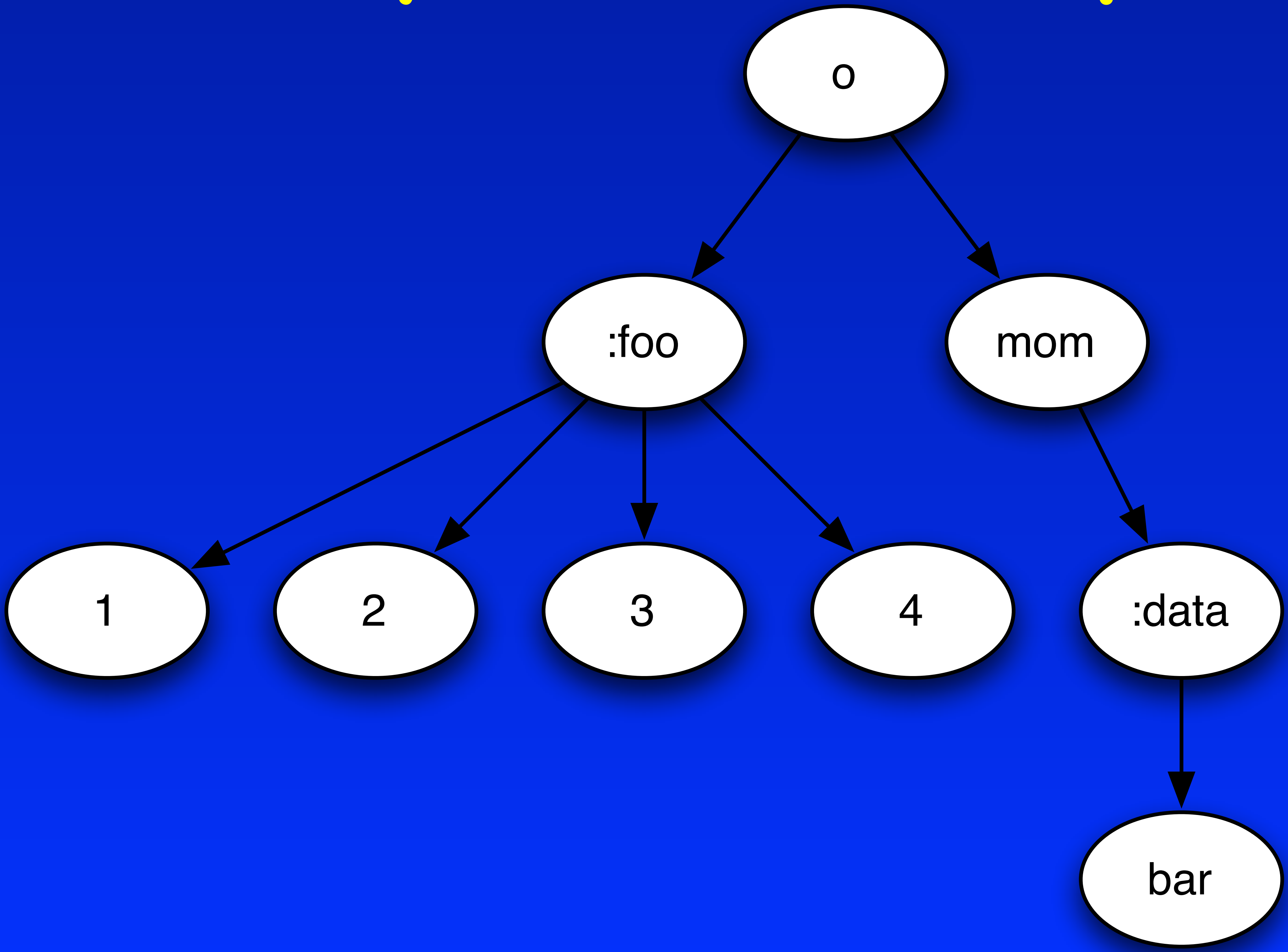
Language Dependencies

- Ruby (of course)
- YAML (psych)
- XML / HTML (nokogiri)

Data is a Tree

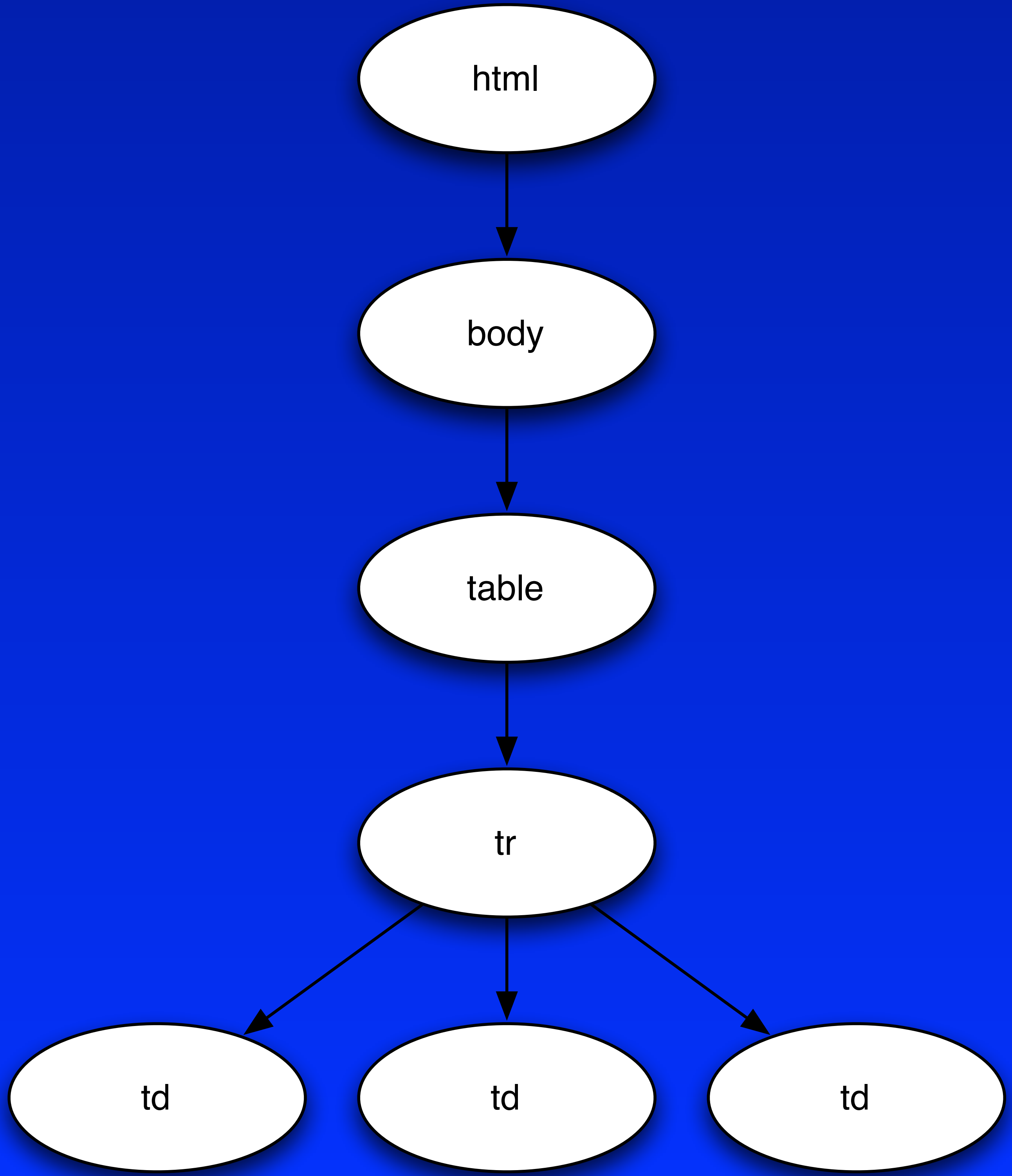
```
o = {
   :foo  => [1,2,3,4],
   'mom' => Struct.new(:data).new('bar')
}
```

Ruby Data Graph



WUPS

HTML data graph



W
U
P
S

Translation

翻訳

WUPS

HTML data is a subset of Ruby

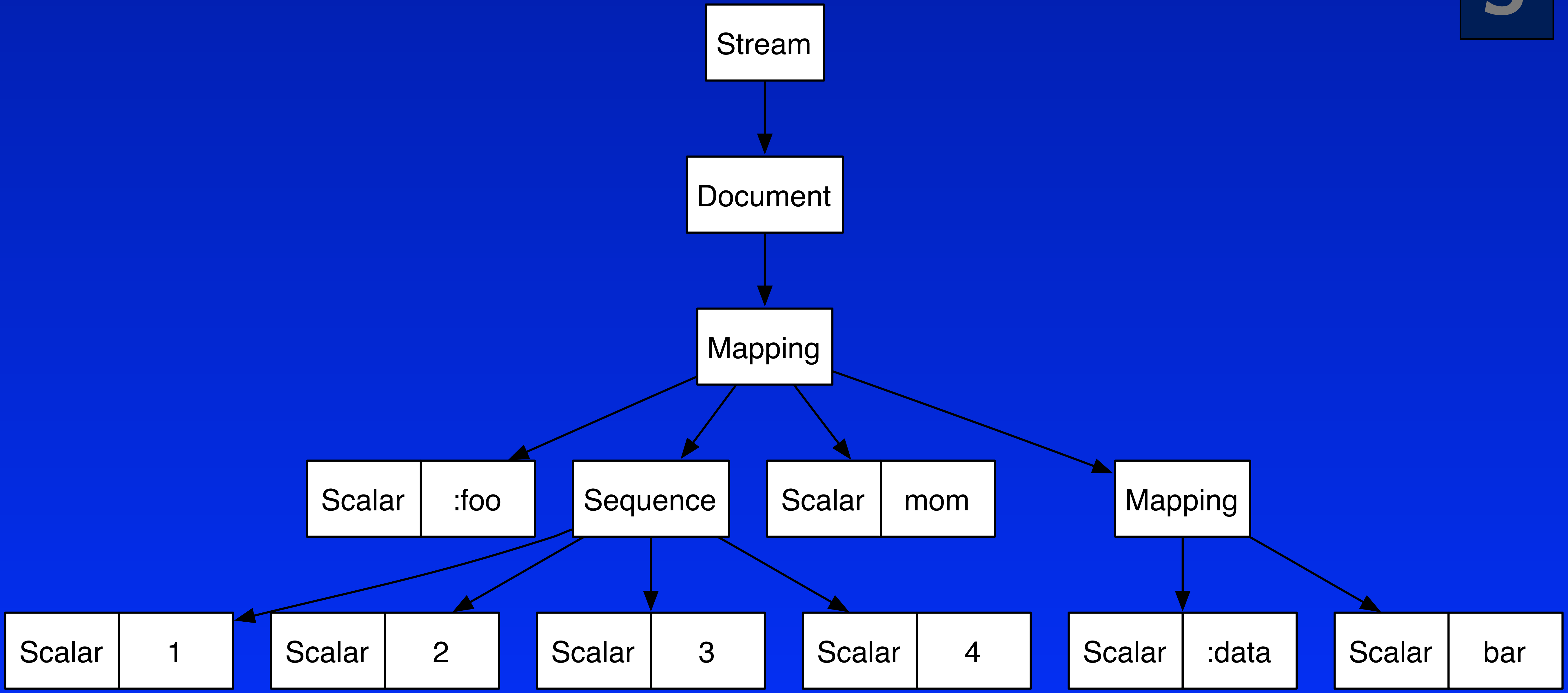
W
U
P
S

YAML to the rescue

YAML Representation

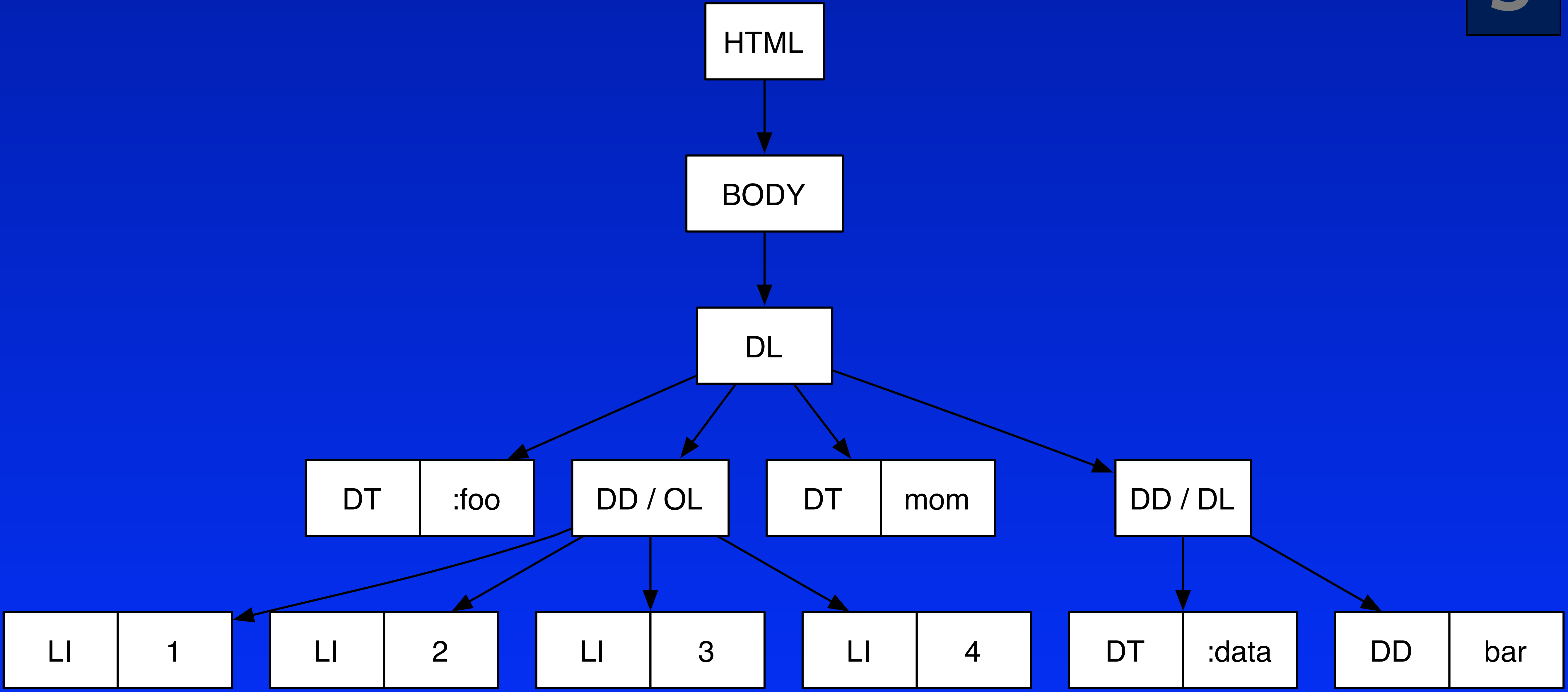
```
---
:foo:
- 1
- 2
- 3
- 4
mom: !ruby/struct
  :data: bar
```

Ruby => YAML AST



YAML AST => HTML

W U P S

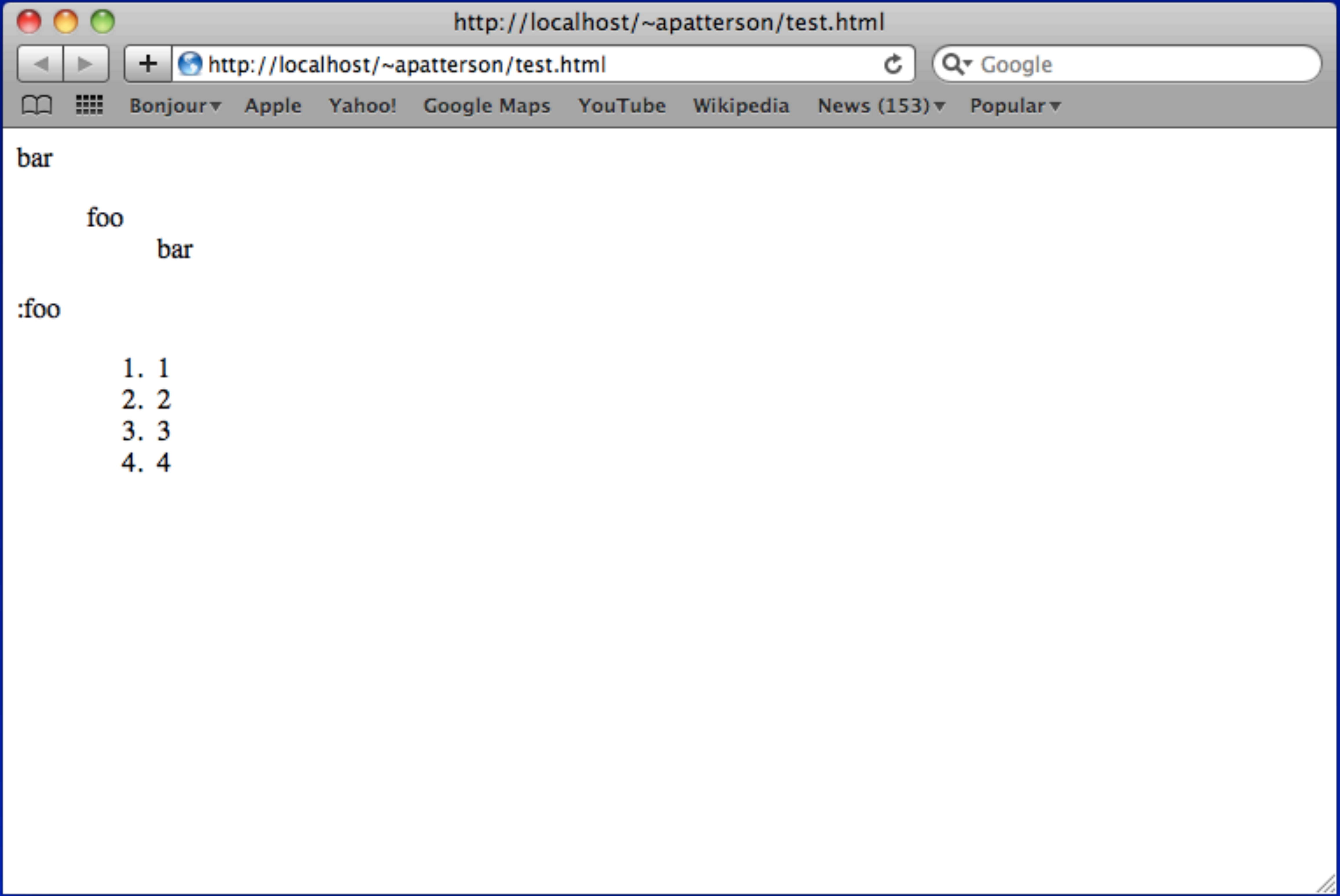


Sample Use

```
o = {
  :foo  => [1,2,3,4],
  'bar' => Struct.new(:foo).new('bar')
}

Wank::HTML::Marshal.dump(o)
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
</head>
  <body>
    <dl>
      <dt>bar</dt>
      <dd>
        <dl class="!ruby/struct ">
          <dt>foo</dt>
          <dd>bar</dd>
        </dl>
      </dd>
      <dt>:foo</dt>
      <dd>
        <ol>
          <li>1</li>
          <li>2</li>
          <li>3</li>
          <li>4</li>
        </ol>
      </dd>
    </dl>
  </body>
</html>
```

W
U
P
S

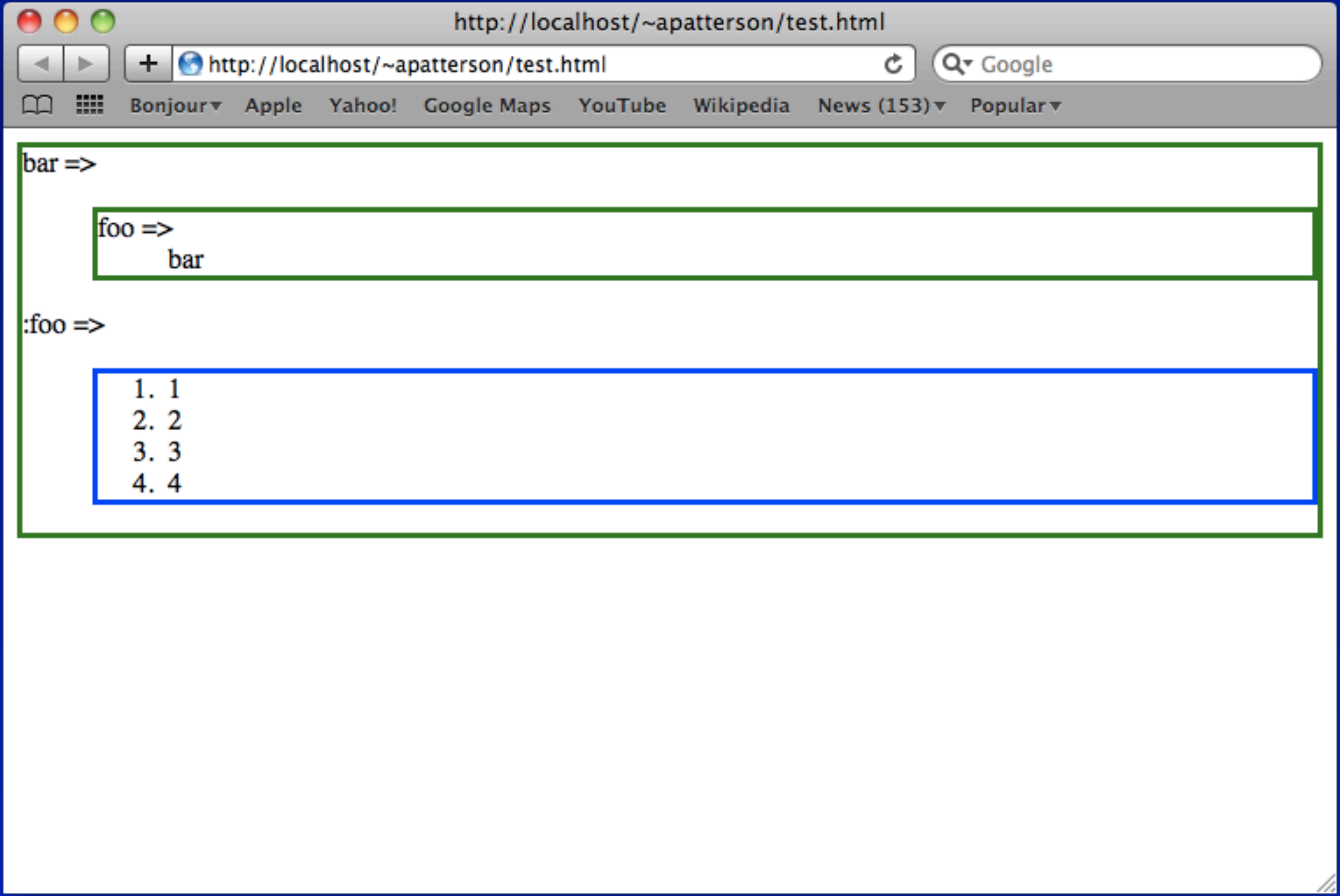

http://localhost/~apatterson/test.html
http://localhost/~apatterson/test.html
Google
Bonjour
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News (153)
Popular

```
bar

      foo
            bar

:foo

         1. 1
         2. 2
         3. 3
         4. 4
```

W U P S

Wank in Style!

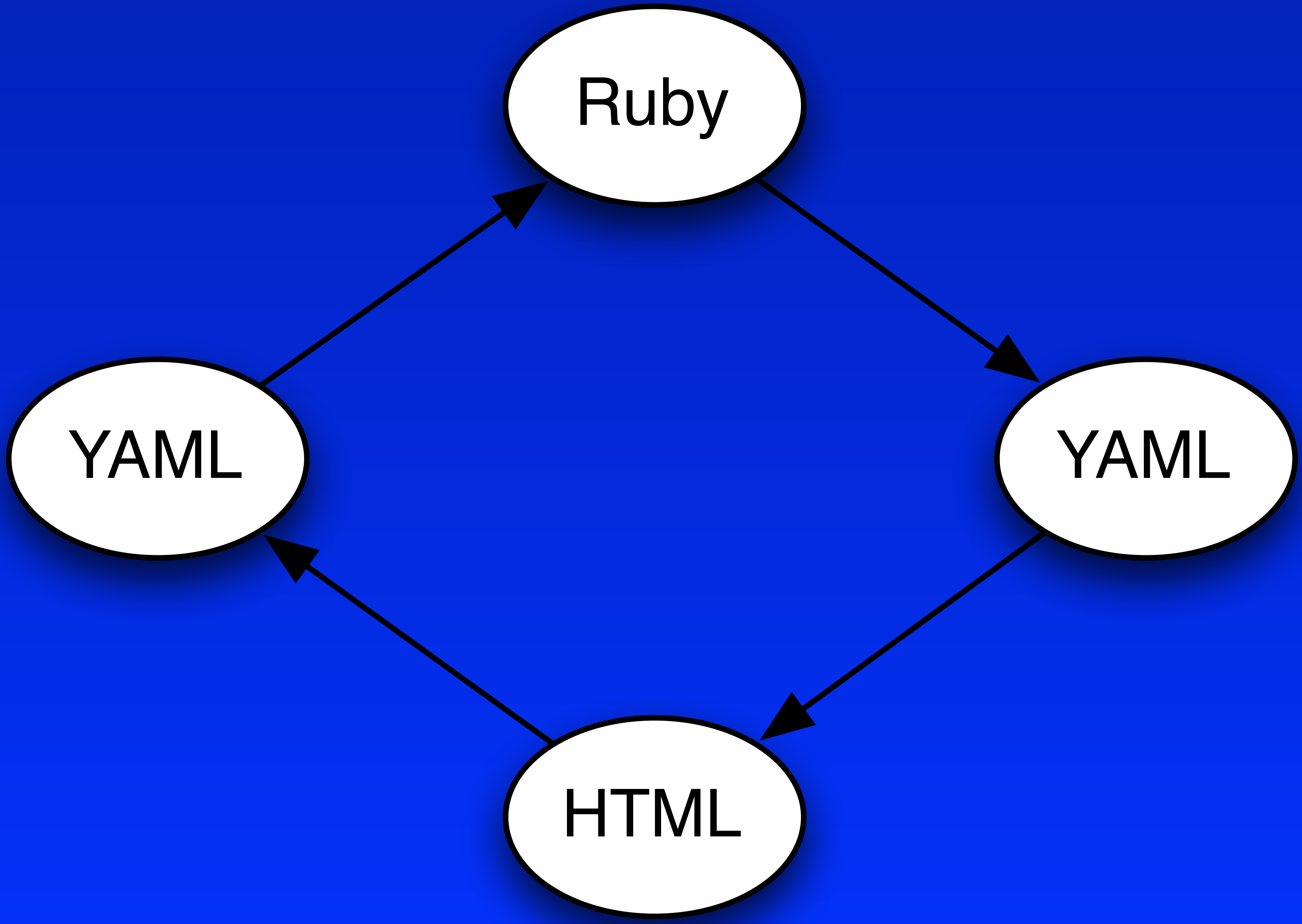
```
dl {
  border: solid green;
}

dt:after {
  content: " => ";
}

ol {
  border: solid blue;
}
```



```
bar =>

    foo =>
        bar

:foo =>

    1. 1
    2. 2
    3. 3
    4. 4
```

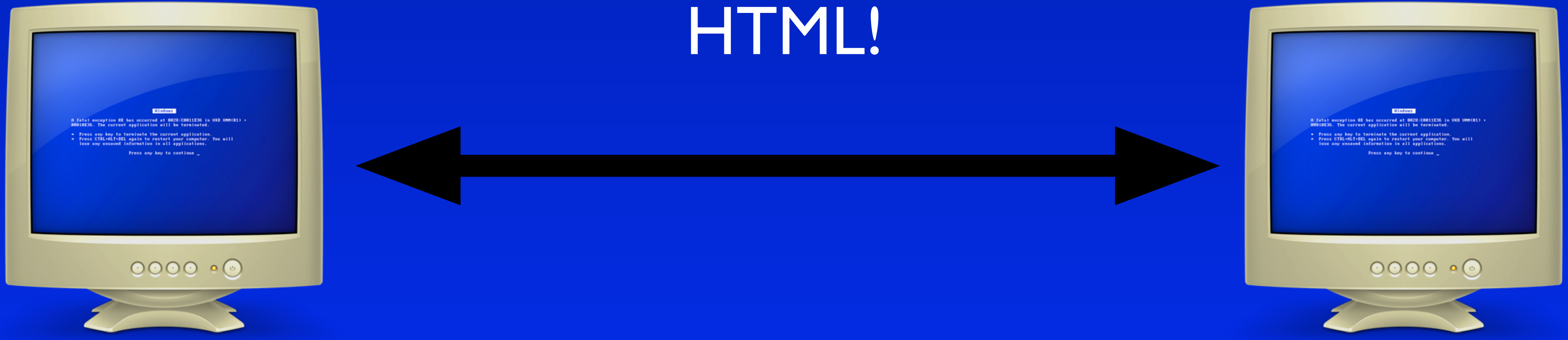
WUPS

The Circle of Terrible



$$\begin{tikzcd}
& \text{Ruby} \arrow[dr] & \\
\text{YAML} \arrow[ur] & & \text{YAML} \arrow[dl] \\
& \text{HTML} \arrow[ul] &
\end{tikzcd}$$

Wank over DRb



Wank over DRb

```
DRb::Marshal = Wank::HTML::Marshal
```

Wank on Rack

```
module Rack
  class Wank
    include Wank::HTML

    def call env
      [
        200,
        {},
        Marshal.dump("hello")
      ]
    end
  end
end
```

Wank on Rails

```
class WankController
  def show
    @wanker = Wanker.find(params[:id])
    render :text =>
     Wank::HTML::Marshal.dump(@wanker)
  end
end
```

W
U
P
S

You can Wank

<u>Anywhere</u>!

Never Say Die

Never Say Die

- Rescue from segfaults
- Create segfaults

libsigsegv

- pageable virtual memory
- memory mapped access to databases
- generational garbage collectors
- stack overflow handlers
- distributed shared memory

libsigsegv

- pageable virtual memory
- memory mapped access to databases
- generational garbage collectors
- stack overflow handlers
- distributed shared memory

HUH?

Trap INT

```
trap(“INT”) do
  puts "haha! no!"
end
```

Trap SEGV

```
begin
  ... # something dangerous
rescue NeverSayDie
  ... # Fix your memory!
end
```

W
U
P
S

Fully Tested

NeverSayDie::segv

W
U
P
S

```
begin
  NeverSayDie.segv
rescue NeverSayDie => maverick
  return maverick          # =>
end
```

W
U
P
S

```
begin
  NeverSayDie.segv
rescue NeverSayDie => maverick
  return maverick           # =>
end
```

Uselessish

W
U
P
S

W
U
P
S

Uselessish

- If you think you need this…

Uselessish

- If you think you need this...
  - well, you probably do.

Poe's Law

```
begin
  asm :thuper_optimized do
    eax.mov 0
    ecx.mov 10
    count = self.label
    eax.add 1
    count.loop

    to_ruby eax
  end
rescue NeverSayDie   # run the slow one :(
  1.upto(10) do; end
end                  # really??
```

Never Say Die, on Rails

```
class SegvController < ...
  def index
    ...
  rescue NeverSayDie
    ... # Yay!  More uptime!
  end
end
```

Phuby

Because Rails programmers are secretly PHP programmers

Hire PHP Programmers!

They're cheap!

Phuby

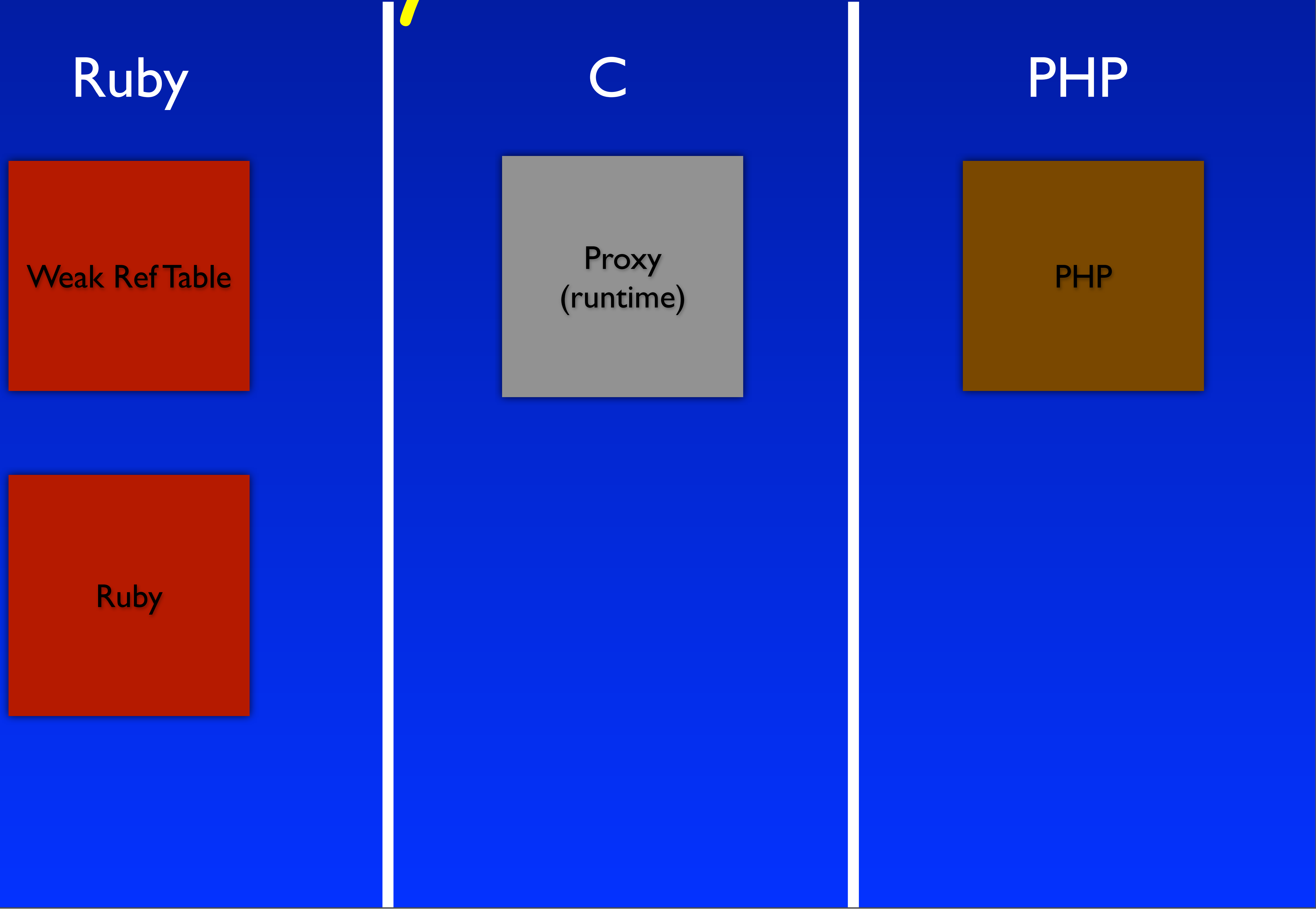
- PHP embedded in Ruby

Source of Bad Ideas

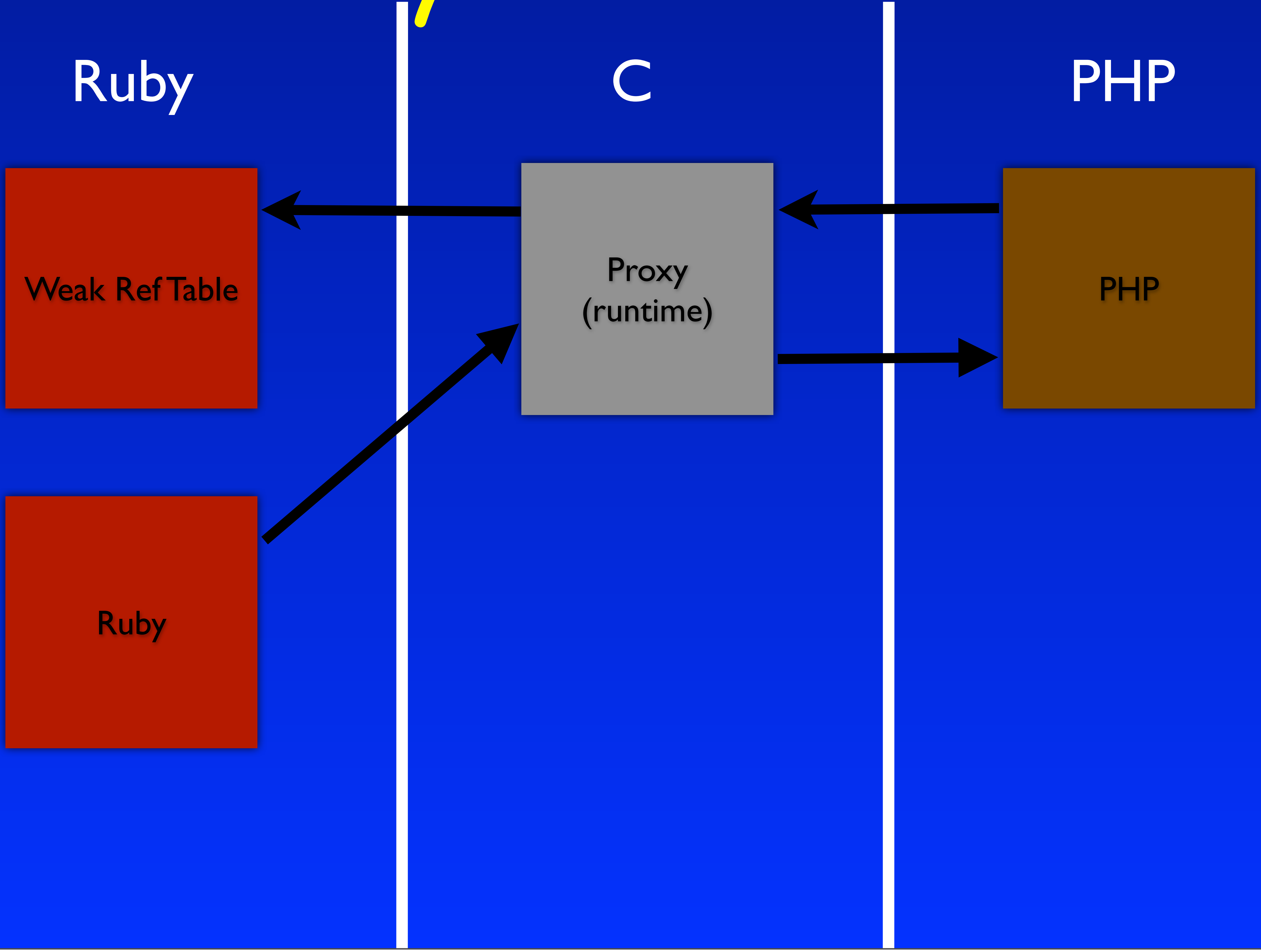
- Ryan came up with the idea of Phuby.
- Aaron implemented phuby.
- Ryan is counting this as a win x 2.

Well Engineered

Ruby var => PHP



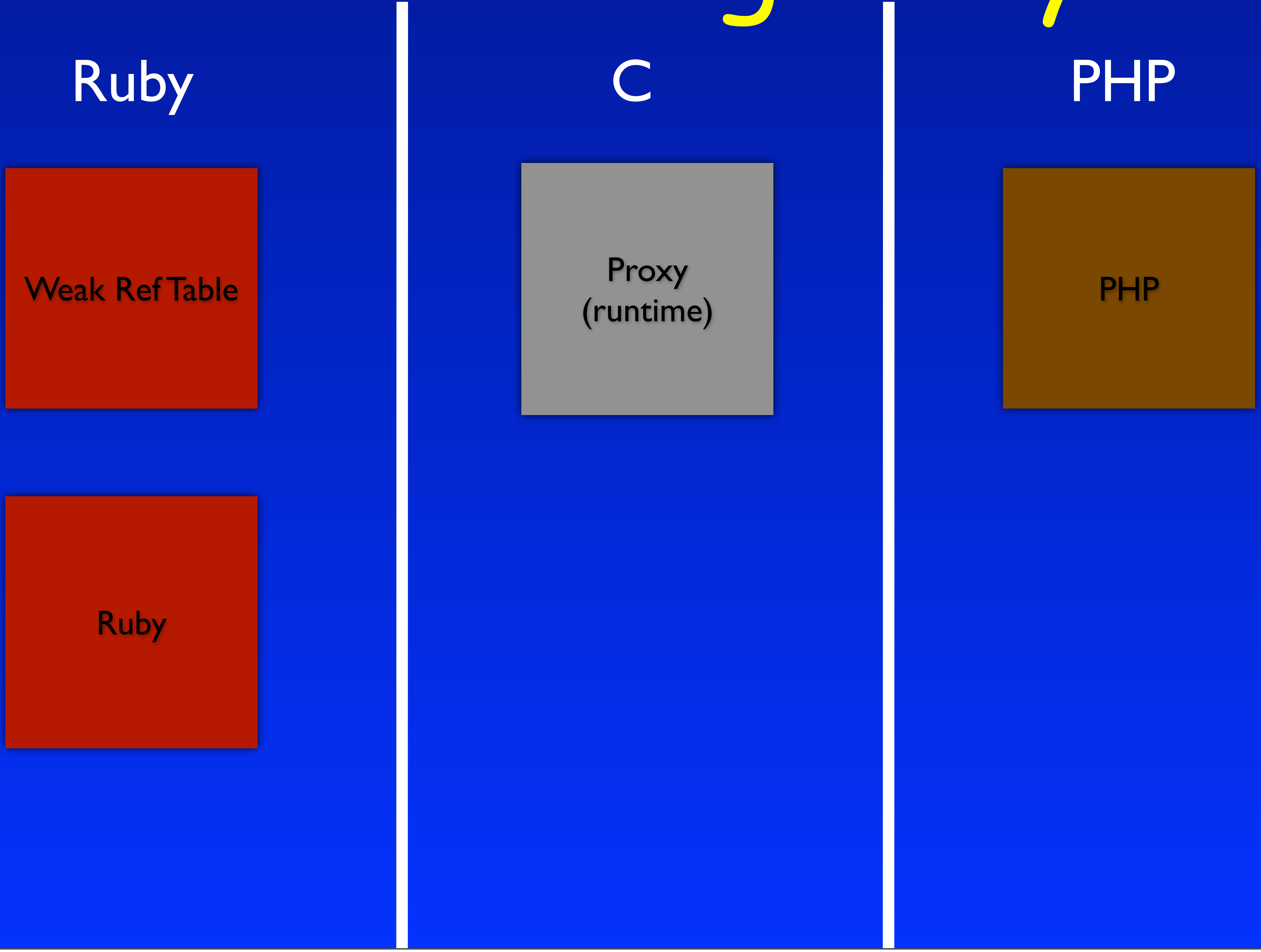
Ruby var => PHP



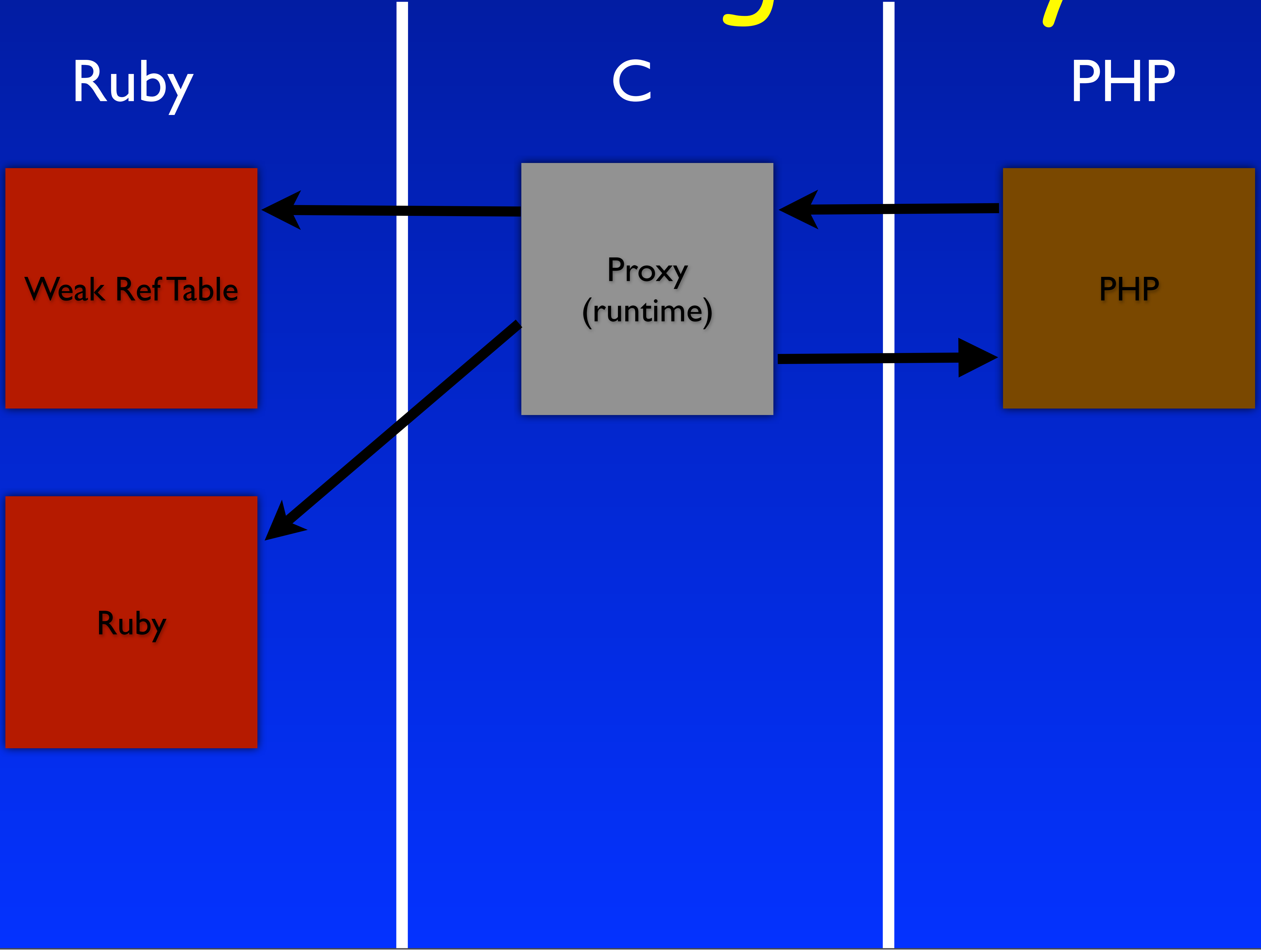
Weak Ref Table

- PHP object memory location (INT)
- VALUE

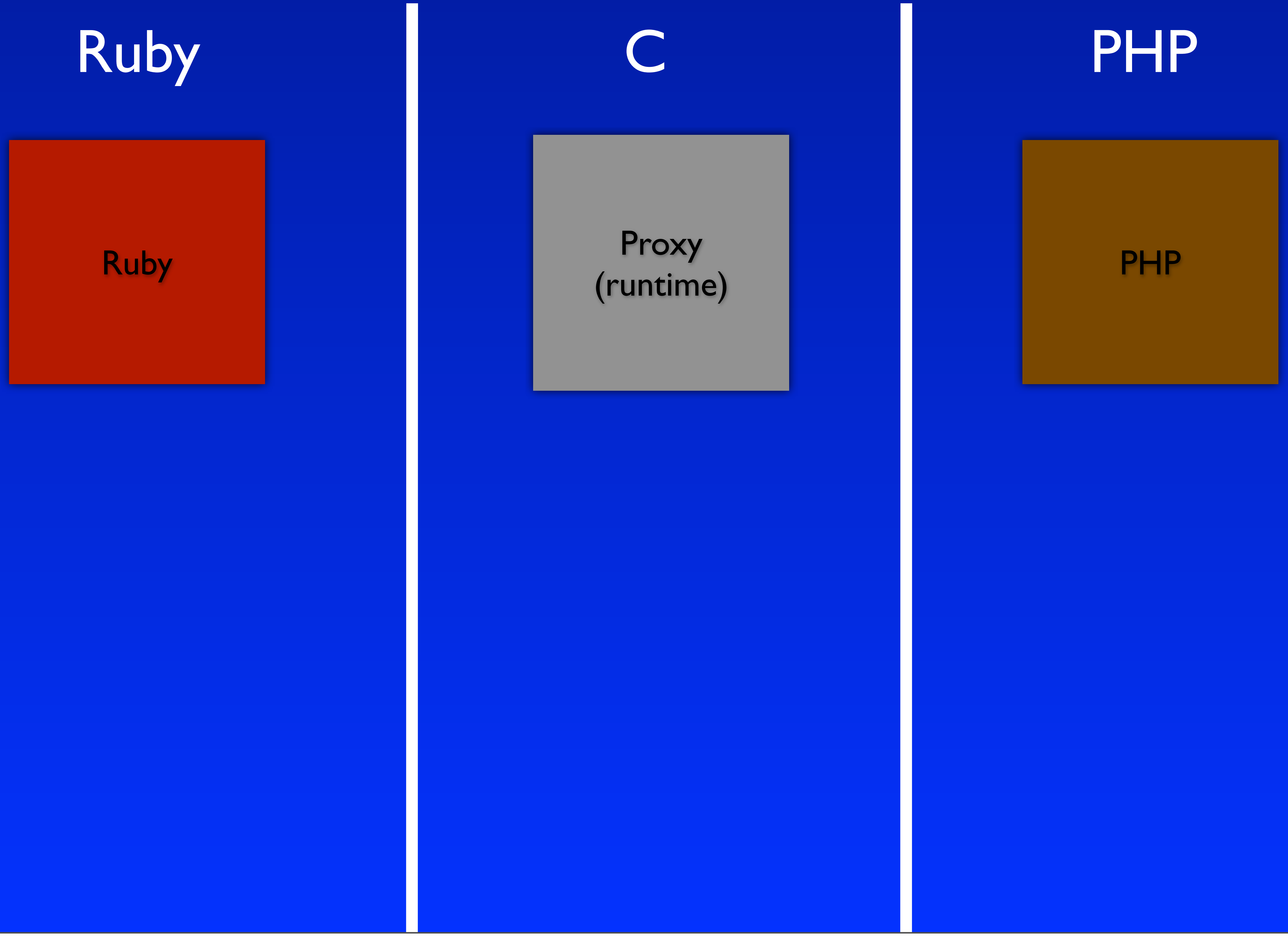
PHP calling Ruby



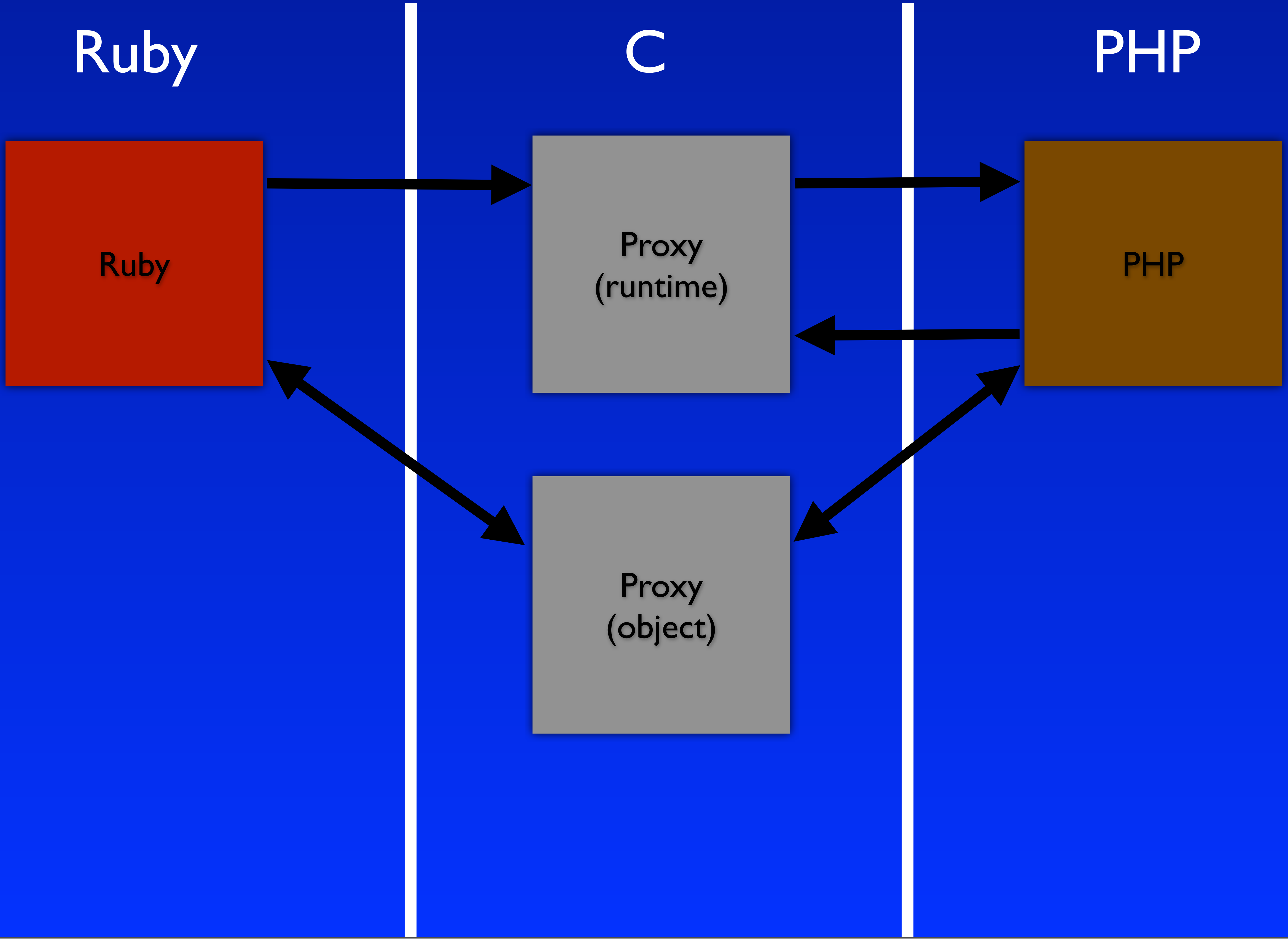
PHP calling Ruby



Ruby calling PHP



Ruby calling PHP



Ruby.PHP()

```
Phuby::Runtime.php do |rt|
  rt.eval('$v = strlen("PHP IS AWESOME");')
  puts rt['v']  # => 14
end
```

Ruby.PHP()

```
Phuby::Runtime.php do |rt|
  rt.eval('$foo = array();')
  rt.eval('$foo["hello"] = "world";')

  foo = rt['foo'] # => #<Phuby::Array:0x101f8f848>
  p foo['hello']  # => ‘world’
end
```

$PHP->Ruby();

W
U
P
S

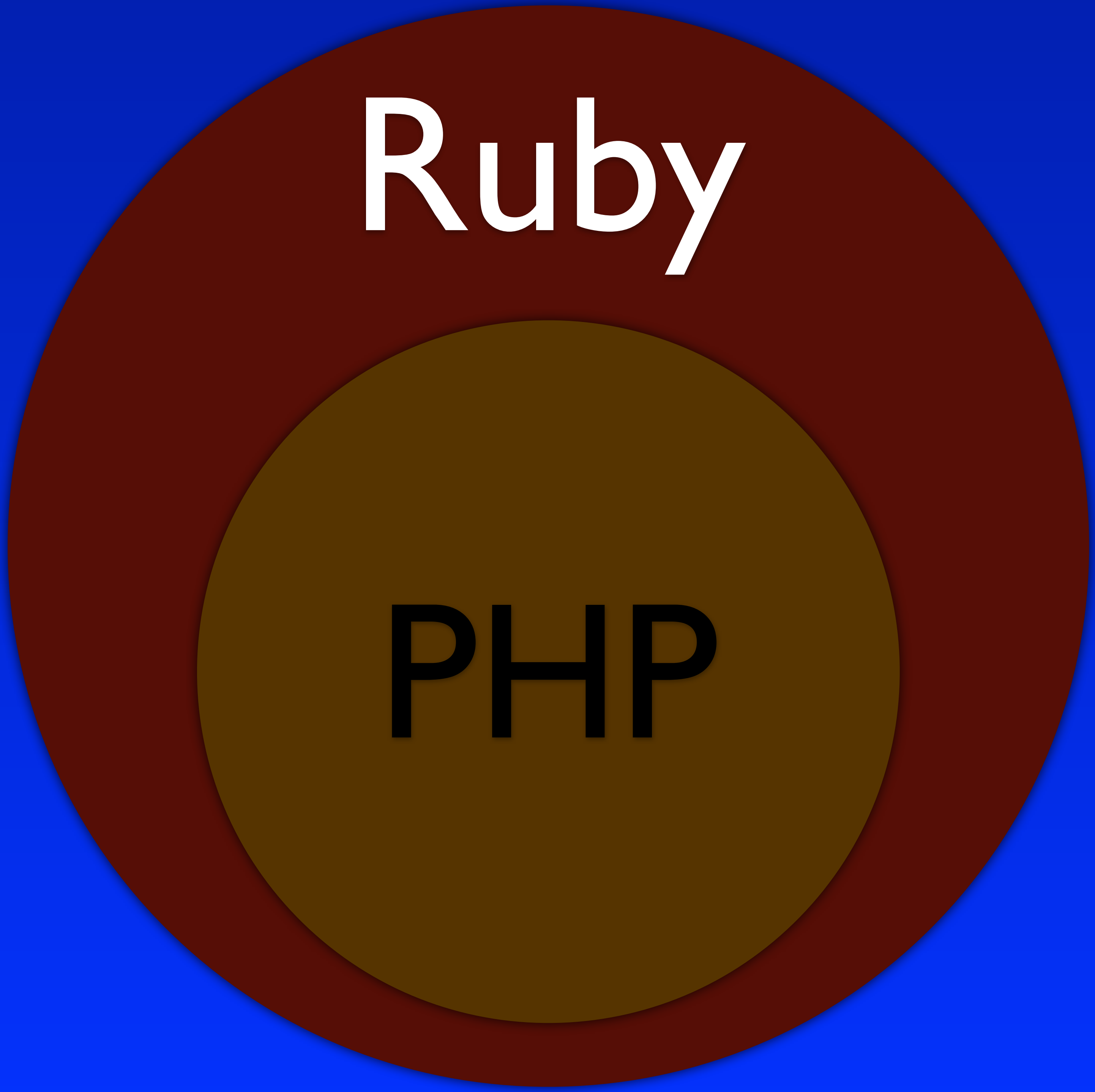
```
class FUN
  def times
    puts "hello"
  end
end

Phuby::Runtime.php do |rt|
  rt['fun'] = FUN.new
  rt.eval('$fun->times();') # => hello
end
```

You got your PHP in my…

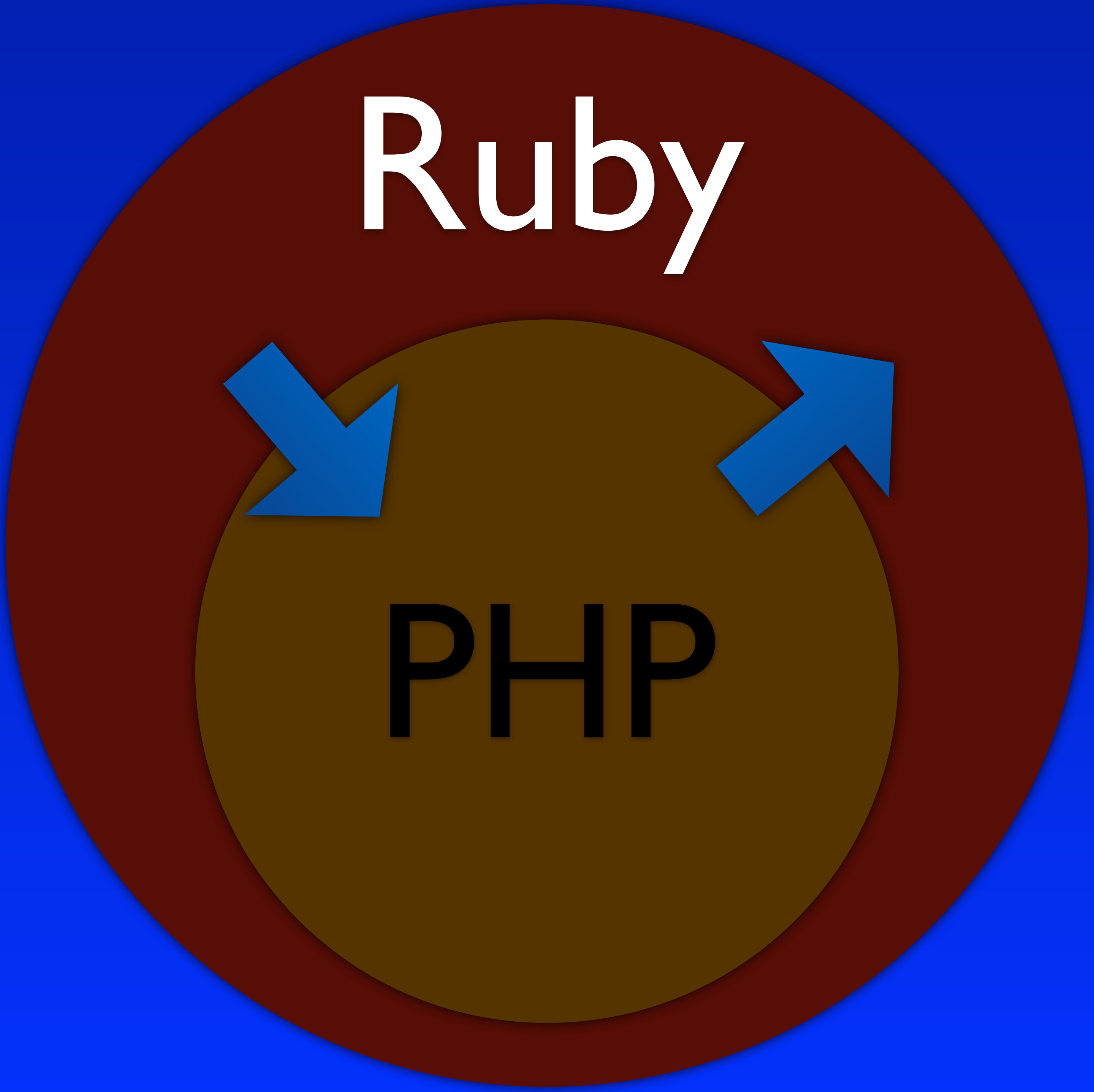
WUPS

Runtimes

Embedded Runtimes



Embedded Runtimes



Circle of Terrible

(In a fail bowl)

W
U
P
S

Web Adapters

WEBRick

W
U
P
S

PHP Events

```
class Events < Phuby::Events
  def initialize req, res
    @req = req
    @res = res
  end

  def header value, op
    k, v = *value.split(':', 2)
    if k.downcase == 'set-cookie'
      @res.cookies << v.strip
    else
      @res[k] = v.strip
    end
  end

  def write string
    @res.body ||= ''
    @res.body << string
  end

  def send_headers response_code
  end
end
```

W
U
P
S

W
U
P
S

Adapter

```
module Phuby
  class PHPHandler < WEBrick::HTTPServlet::FileHandler
    def process verb, req, res
      file = File.join(@root, req.path)

      Dir.chdir(File.dirname(file)) do
        Phuby::Runtime.php do |rt|
          rt.eval("date_default_timezone_set('America/Los_Angeles');")

          rt['logger'] = Logger.new($stdout)
          req.request_uri.query.split('&').each do |pair|
            k, v = pair.split '='
            rt["_GET"][k] = v
          end if req.request_uri.query

          req.query.each do |k,v|
            rt["_#{verb}"][k] = v
          end if :POST == verb

          # Set CGI server options
          req.meta_vars.each do |k,v|
            rt["_SERVER"][k] = v || ''
          end
          rt["_SERVER"]['REQUEST_URI'] = req.request_uri.path

          req.cookies.each do |cookie|
            rt["_COOKIE"][cookie.name] = CGI.unescape(cookie.value)
          end

          events = Events.new req, res

          rt.with_events(events) do
            File.open(file, 'rb') { |f| rt.eval f }
          end
        end
      end
      if res['Location']
        res['Location'] = CGI.unescape res['Location']
        res.status = 302
      end
    end
  end
end
```

WUPS

W
U
P
S

Rack

Rack is Hip

Rack is totally sweet bro

http://www.flickr.com/photos/chromewavesdotorg/528814269/

Phrack

- 50 Lines
- Totally Hip

```
class Rack::Phrack < Rack::File
  class Events < Struct.new(:code, :headers, :body)
    def write string; body << string; end
    def send_headers response_code;   end

    def header value, op
      k, v = value.split(': ', 2)
      self.code = 302 if k == 'Location'
      headers[k] = [headers[k], Rack::Utils.unescape(v)].compact.join "\n"
    end
  end

  def call env
    events = Events.new 200, {}, ''
    file   = File.join @root, env['PATH_INFO']
    file   = File.join file, "index.php" if File.directory?(file)

    return super unless file =~ /php$/

    Dir.chdir(File.dirname(file)) do
      Phuby::Runtime.php do |rt|
        rt.eval "date_default_timezone_set('America/Los_Angeles');" # *shrug*

        { Rack::Utils.parse_query(env['QUERY_STRING'])     => "_GET",
          Rack::Utils.parse_query(env['rack.input'].read)  => "_POST",
          Rack::Utils.parse_query(env['HTTP_COOKIE'], ';') => "_COOKIE",
        }.each do |from, to|
          from.each { |k,v| rt[to][k] = v }
        end

        env.each { |k,v| rt['_SERVER'][k] = v || '' unless k =~ /^rack/ }
        rt["_SERVER"]['REQUEST_URI'] = env['PATH_INFO']

        rt.with_events(events) { open(file) { |f| rt.eval f } } # RUN!
      end
    end
    events.to_a
  end
end

Rack::Handler::WEBrick.run(Rack::Phrack.new(ARGV[0] || Dir.pwd), :Port => 10101)
```

W
U
P
S

W
U
P
S

Phuby Blargh

DHH did it in 15 minutes

W
U
P
S

We can do it in 2

Wordpress on Ruby Video

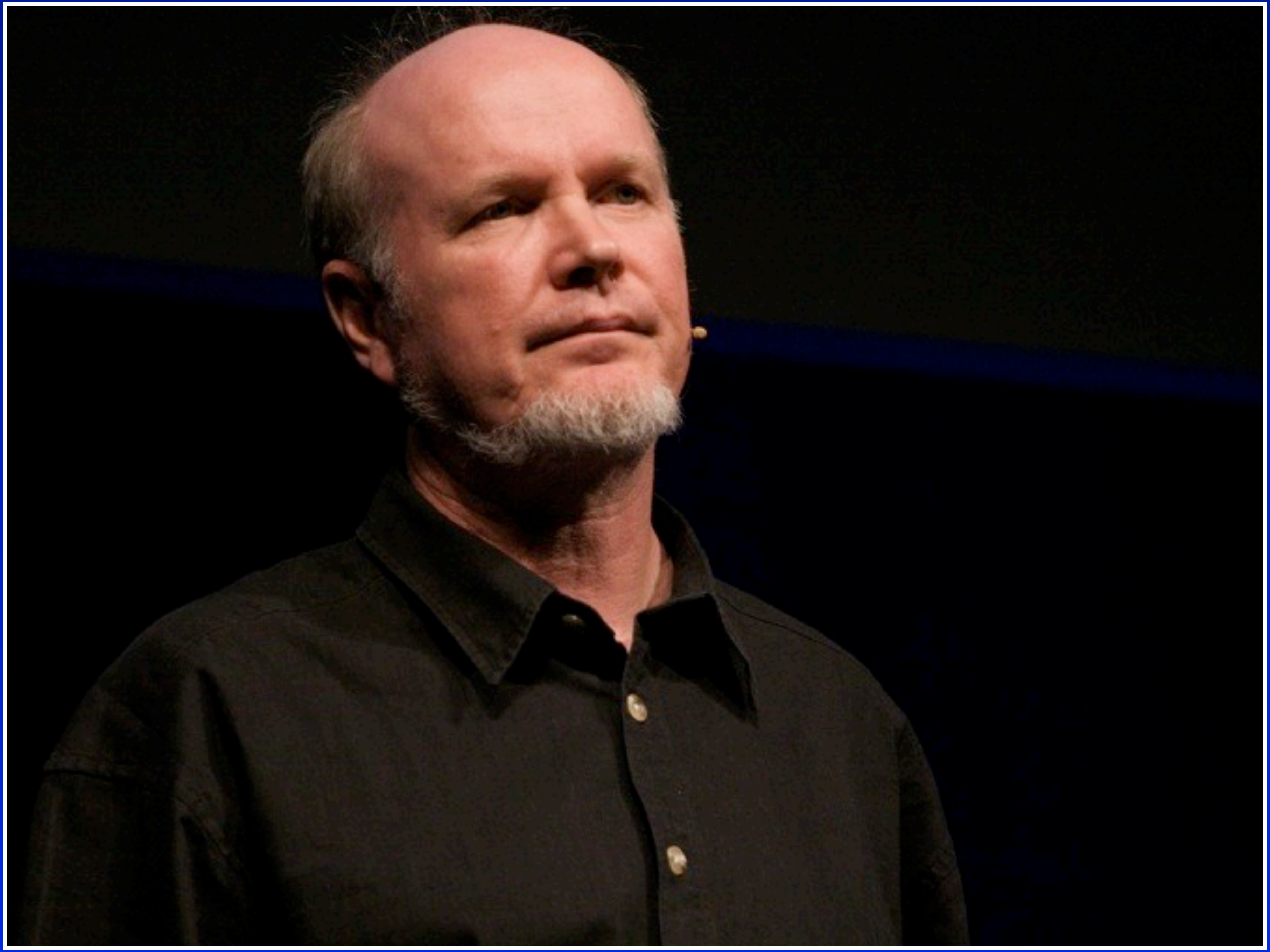
http://www.youtube.com/watch?v=MXERy8Y2eVo

Enterprise

Scalable software at its finest

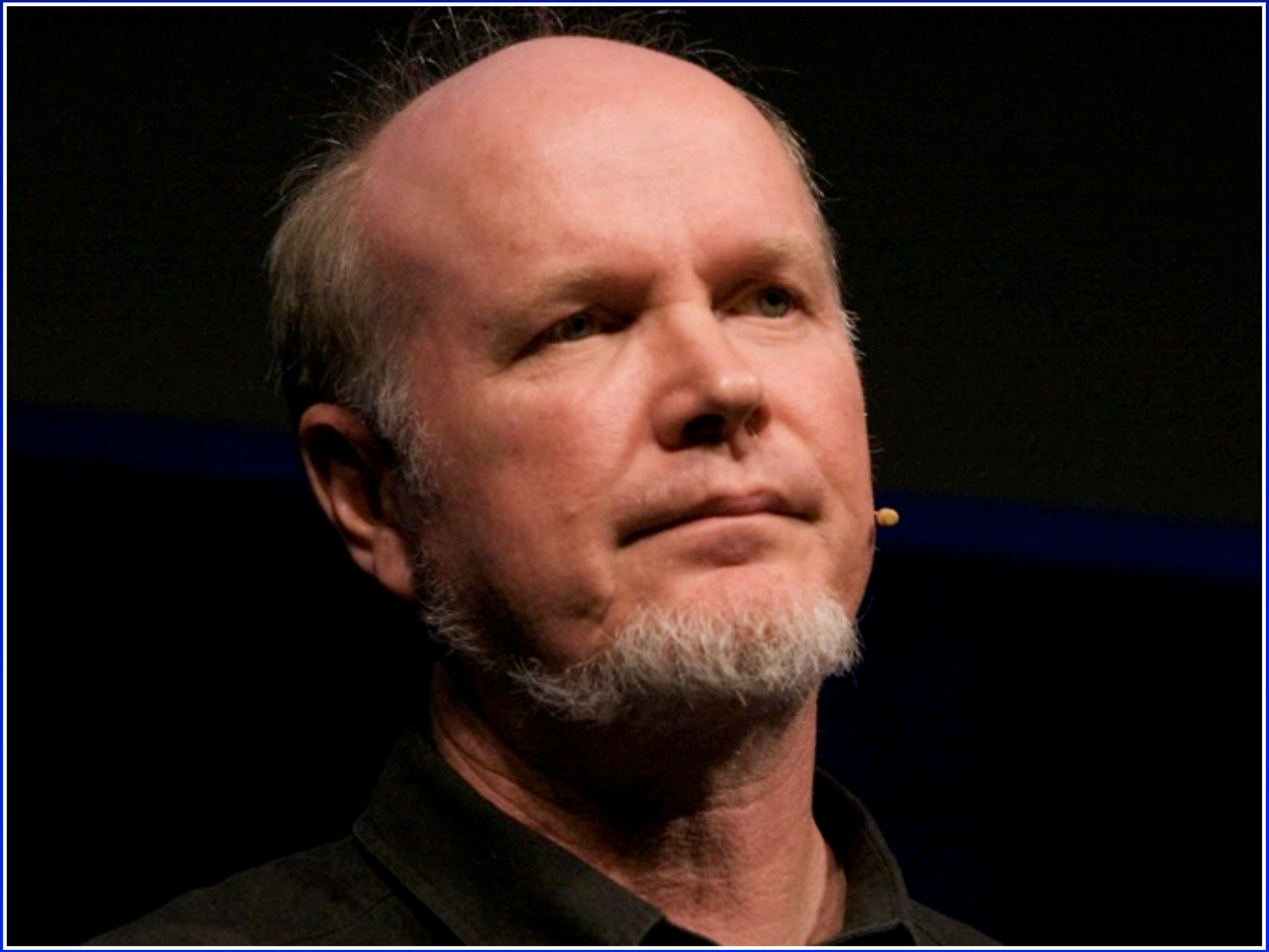
Guiding Principles

Ruby does not scale



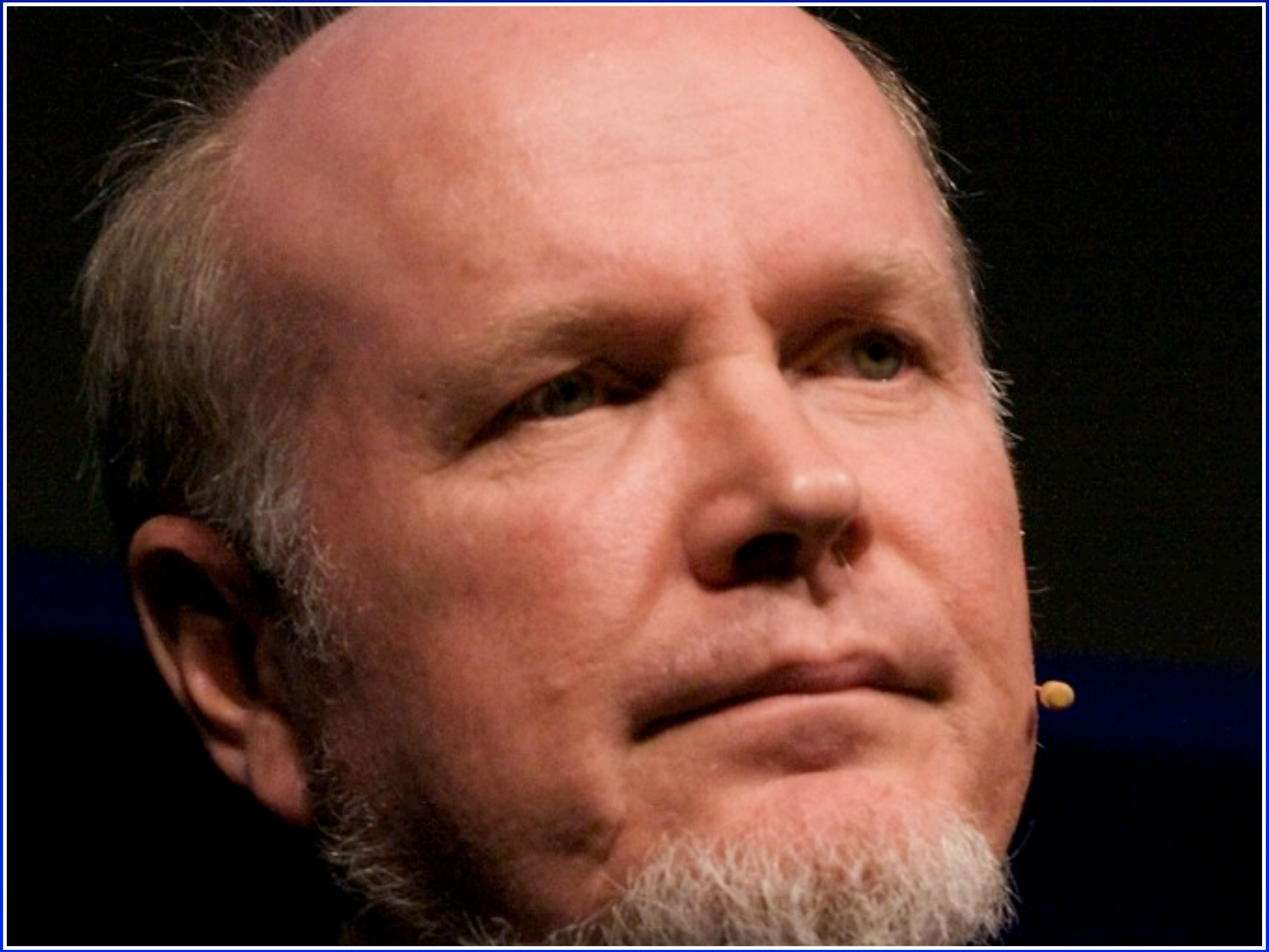
XML

http://www.flickr.com/photos/drewm/3016905054/



scales

http://www.flickr.com/photos/drewm/3016905054/



like

http://www.flickr.com/photos/drewm/3016905054/

a

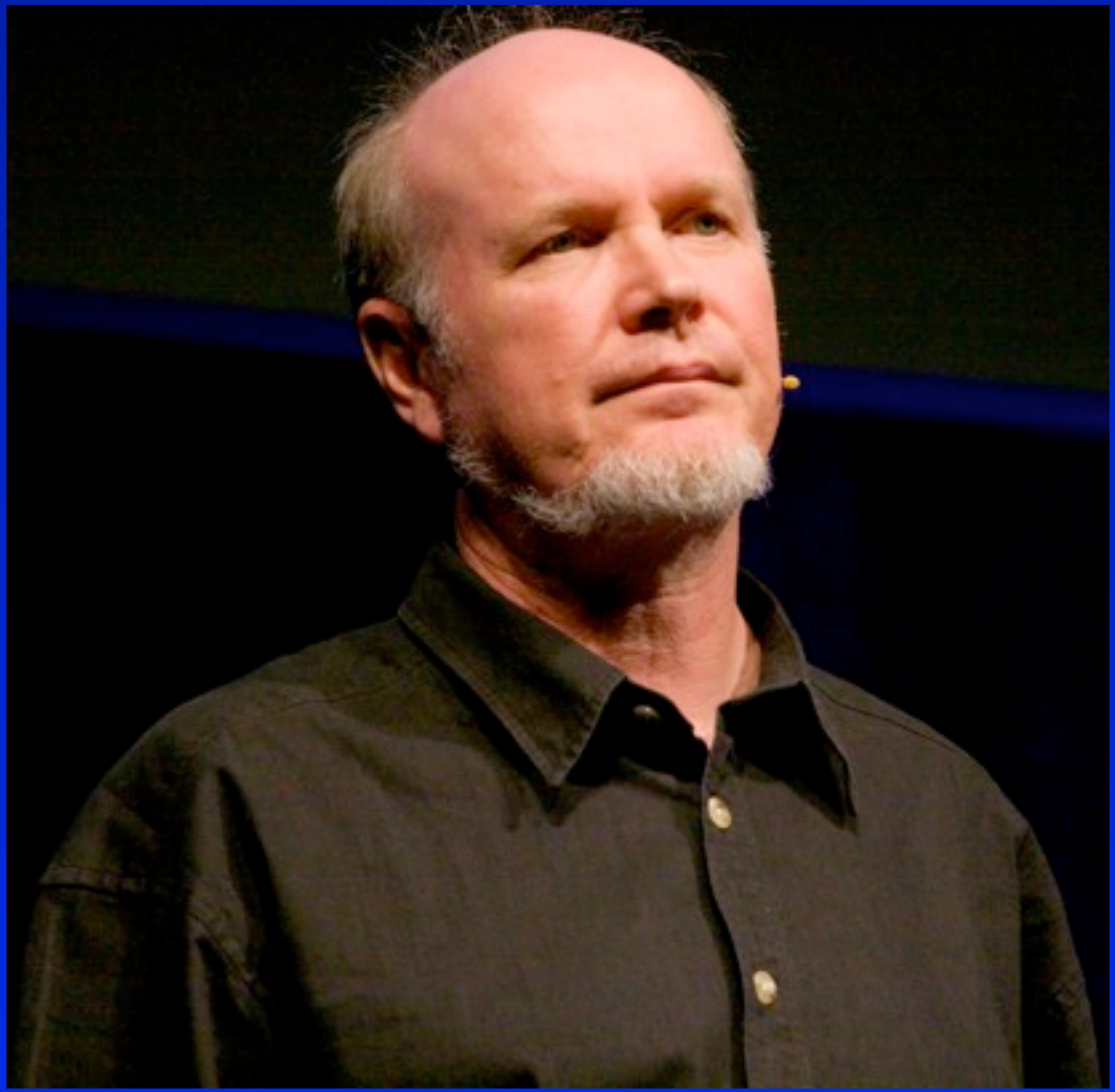
http://www.flickr.com/photos/drewm/3016905054/

boss

http://www.flickr.com/photos/drewm/3016905054/

boss

http://www.flickr.com/photos/drewm/3016905054/



W
U
P
S

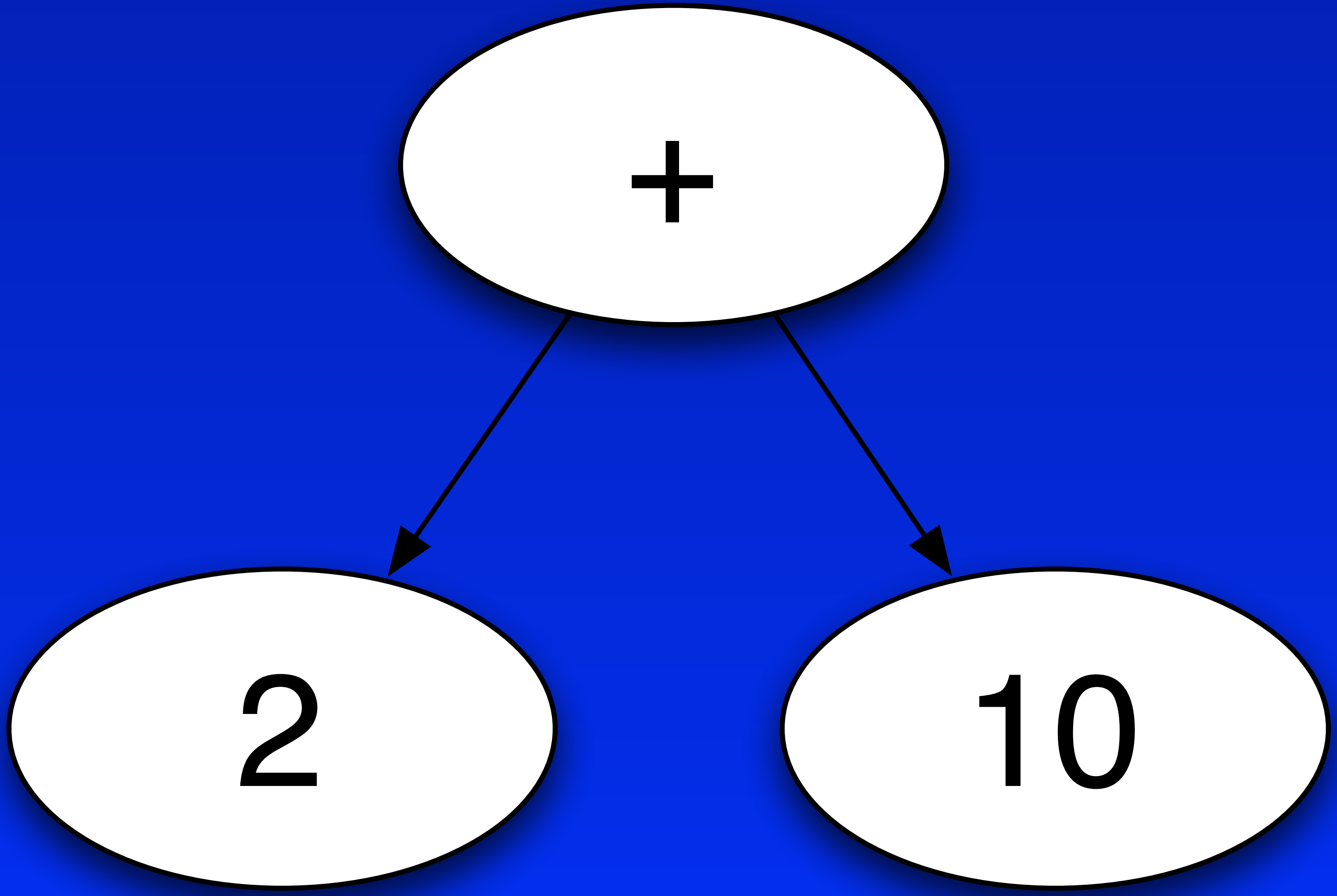
木

Trees

Ruby

2 + 10

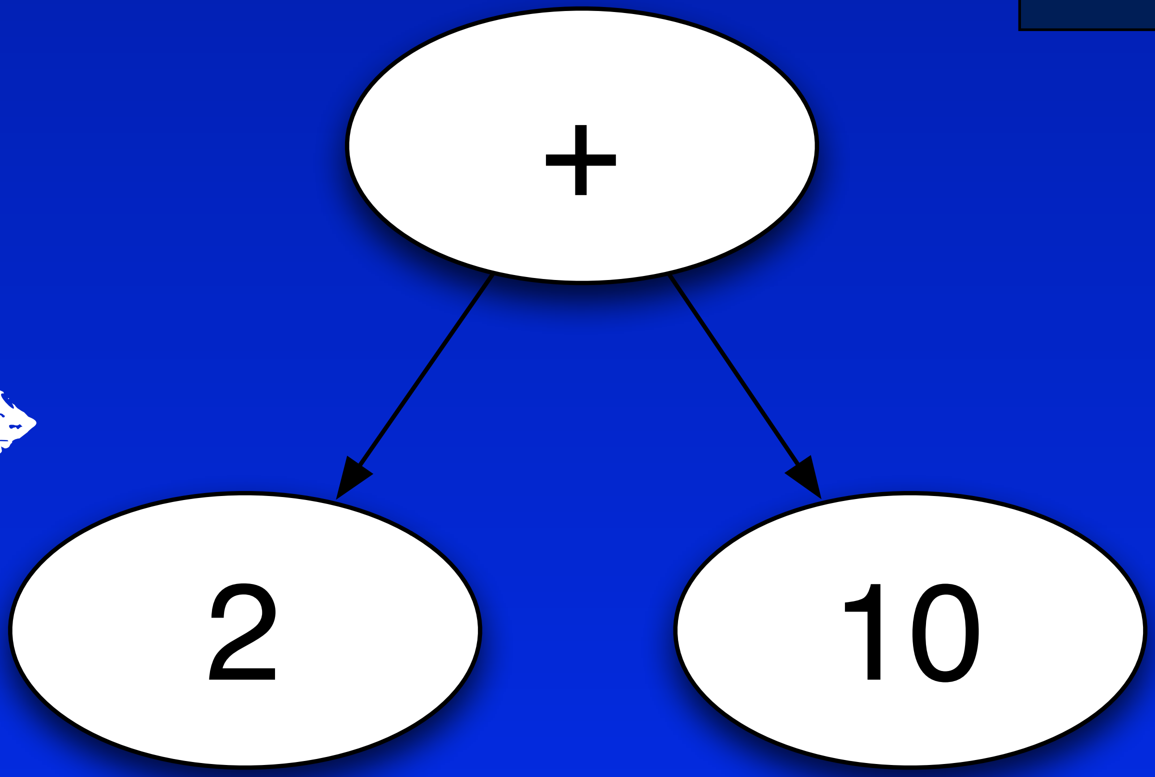
Ruby



Ruby Parser

W U P S

2 + 10 ⟷

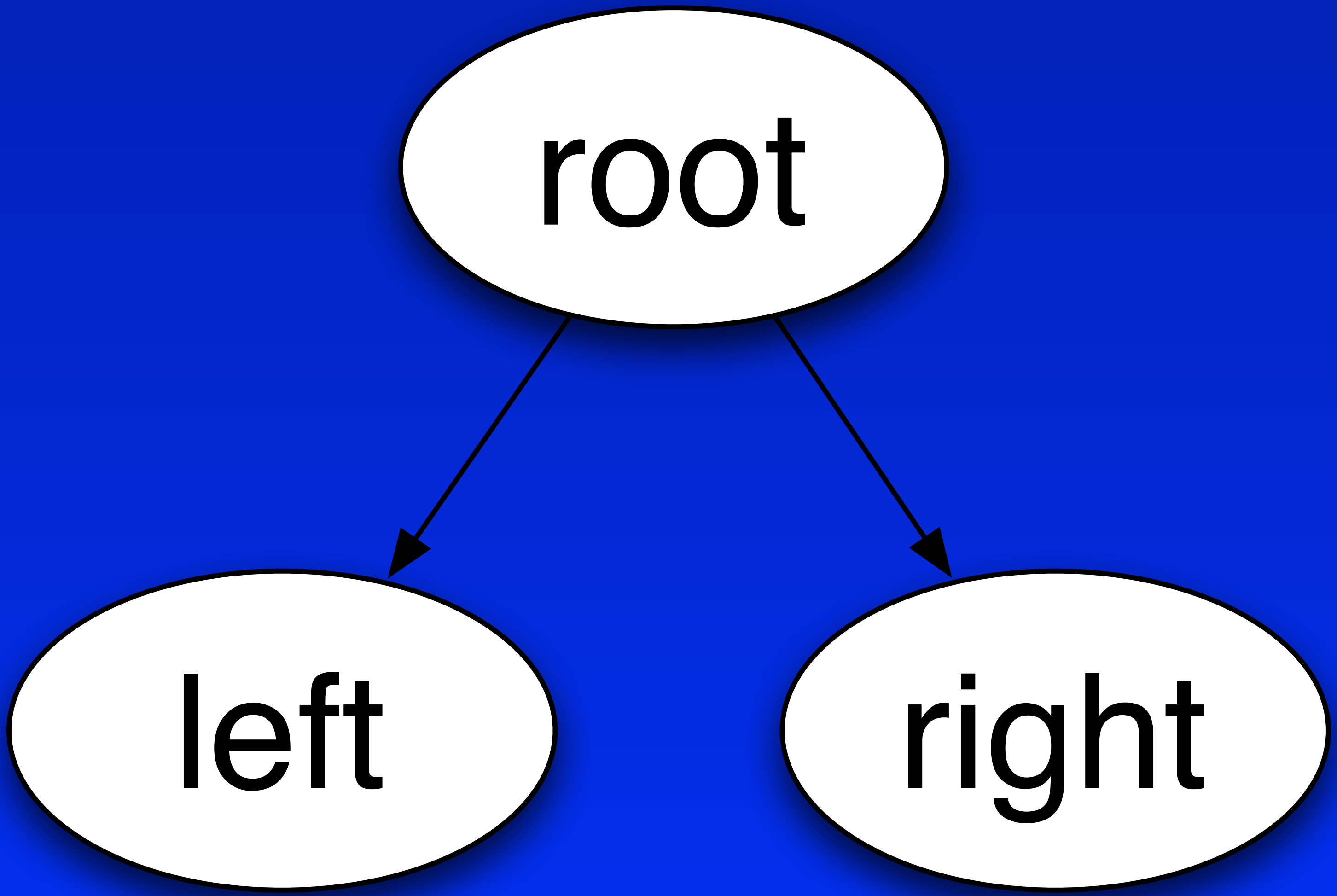


XML

W U P S

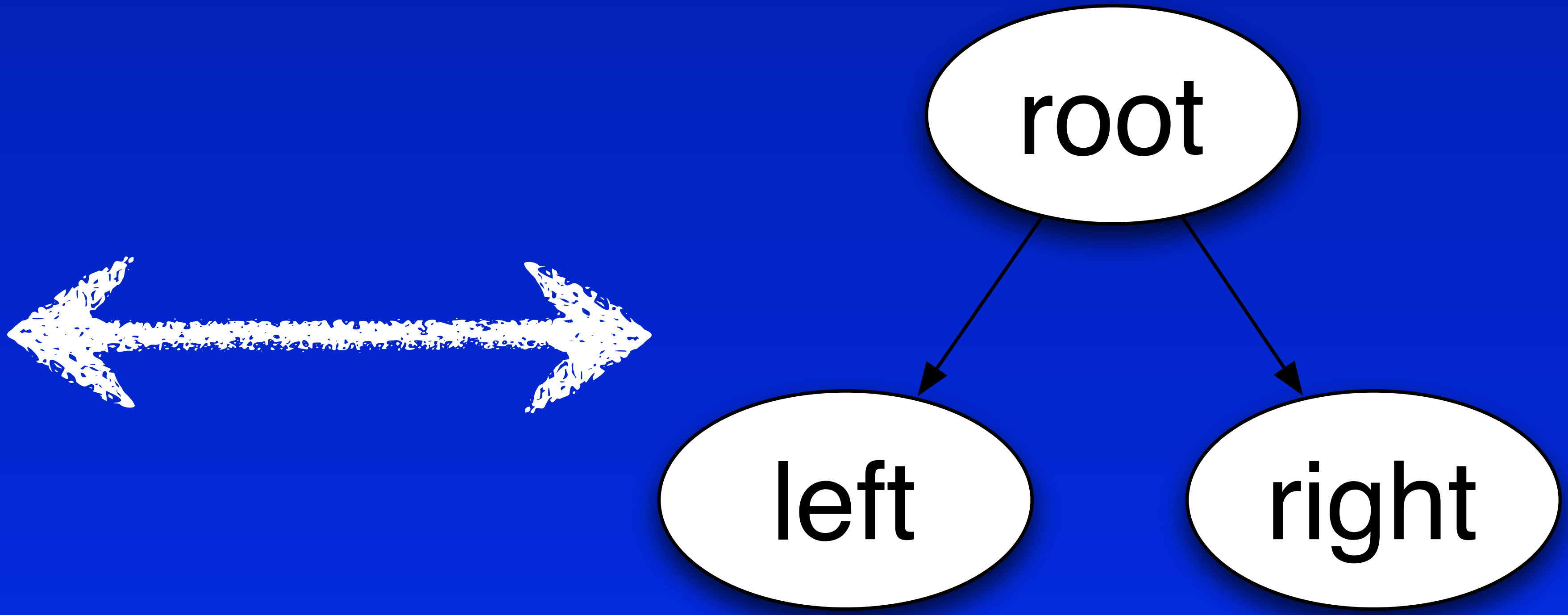
```
<?xml version=”AWESOME”
      encoding=”’MERKIN” ?>
<root>
  <left />
  <right />
</root>
```

XML

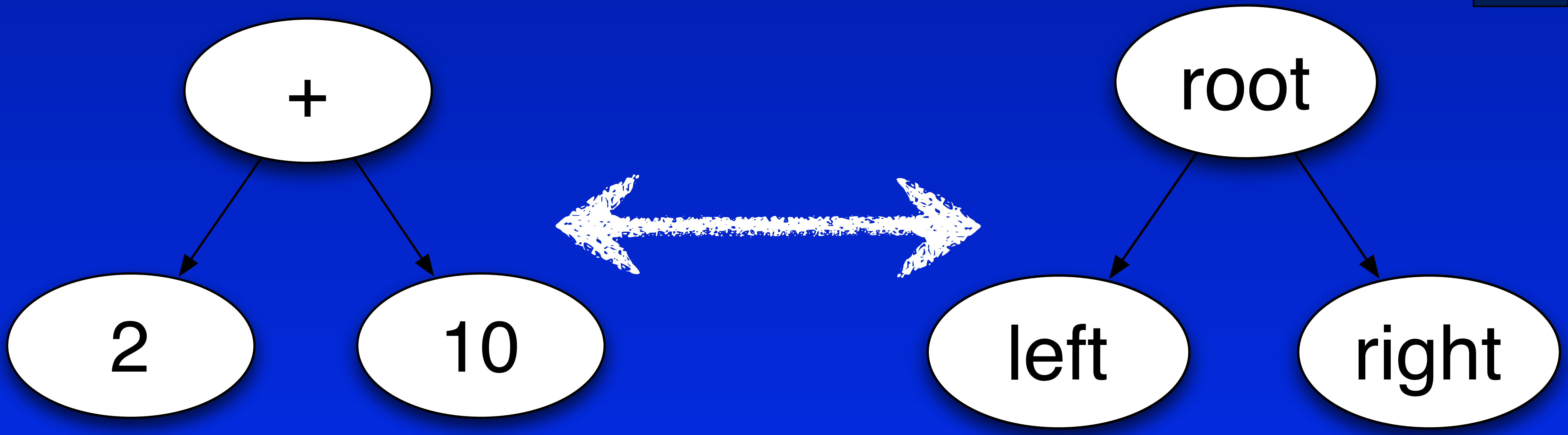


Nokogiri

```
<root>
  <left />
  <right />
</root>
```

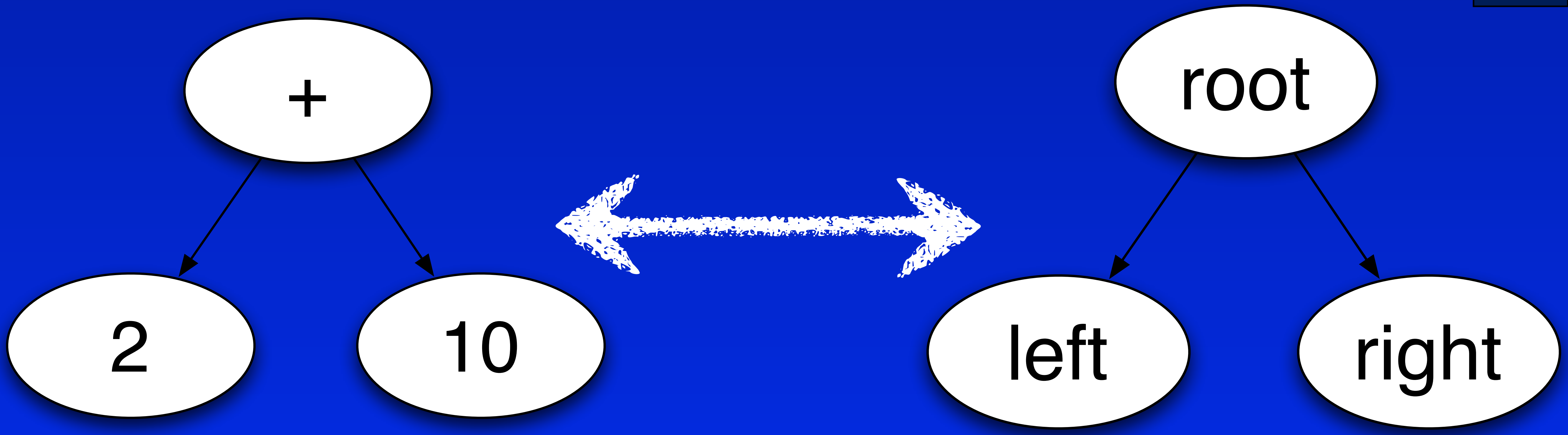


????


+
2
10
root
left
right

W
U
P
S

Enterprise



“Uses”

Meta-programming

Convert “foo” to “bar”

```
sexml = Enterprise::SEXML DATA.read
sexml.xpath('//*[@value = "foo"]').each do |node|
  node['value'] = 'bar'
end

puts sexml.to_ruby

__END__
class Foo
end
foo = Foo.new
foo.hello
```

```
class Foo
end
bar = Foo.new
bar.hello
```

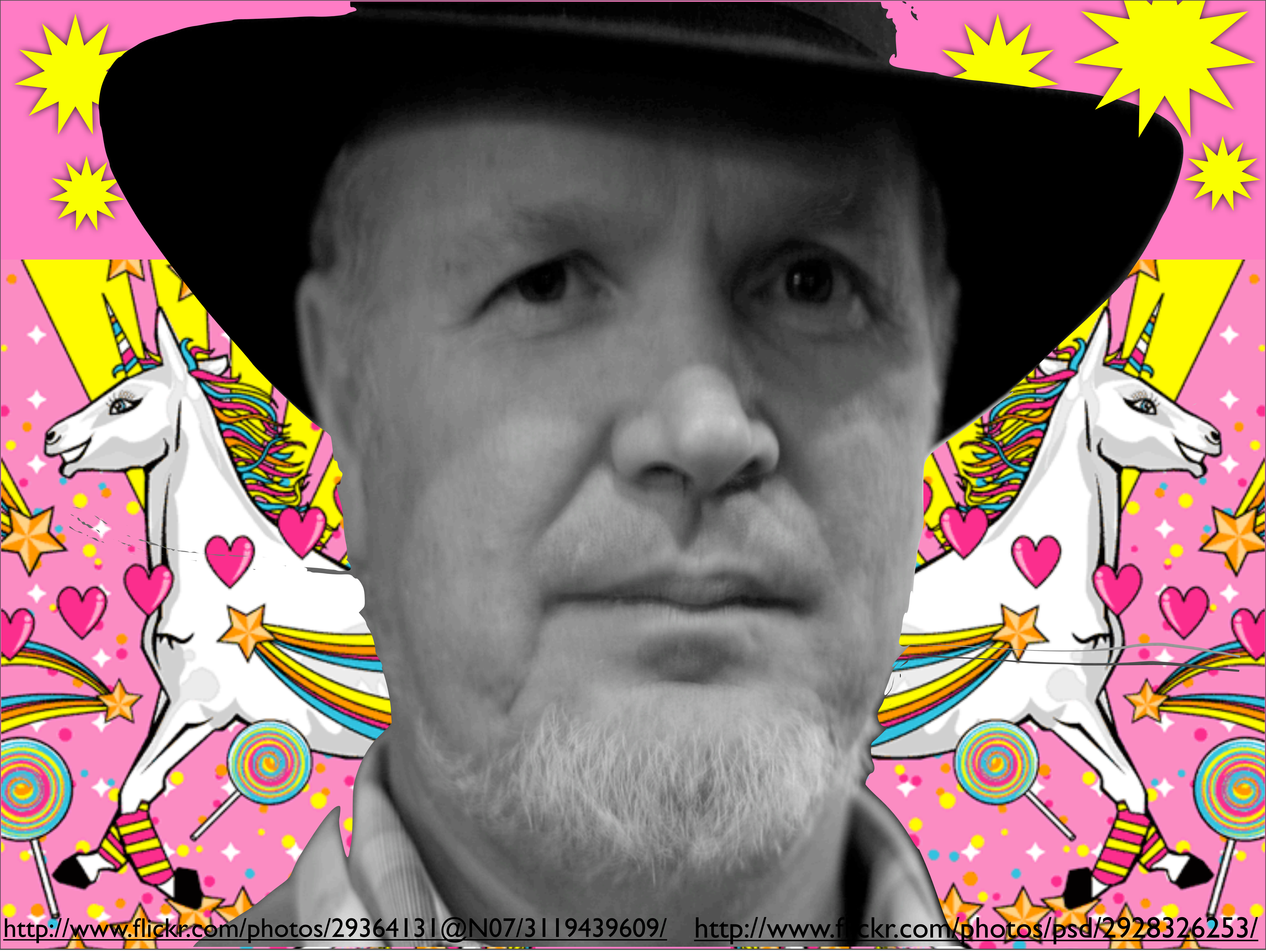
Not Enterprise Enough

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/
1999/XSL/Transform">
  <xsl:template match="//*">
    <xsl:copy>
      <xsl:if test="@type">
        <xsl:attribute name="type">
          <xsl:value-of select="@type" />
        </xsl:attribute>
      </xsl:if>
      <xsl:if test="@value">
        <xsl:attribute name="value">
          <xsl:choose>
            <xsl:when test="@value = 'foo'">bar</xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="@value"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:attribute>
      </xsl:if>
      <xsl:apply-templates />
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

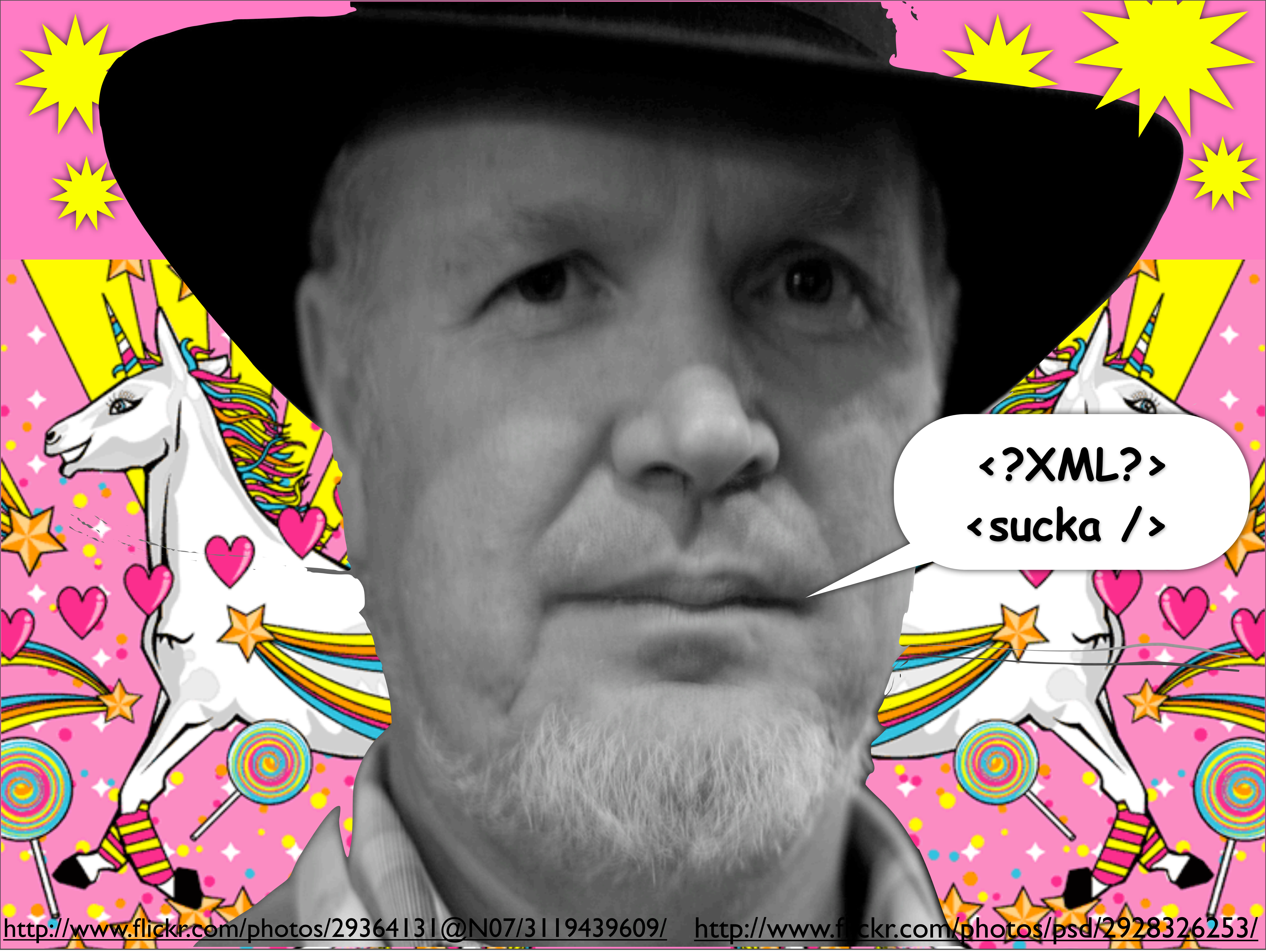
W
U
P
S

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/
1999/XSL/Transform">
  <xsl:template match="//*">
    <xsl:copy>
      <xsl:if test="@type">
        <xsl:attribute name="type">
          <xsl:value-of select="@type" />
        </xsl:attribute>
      </xsl:if>
      <xsl:if test="@value">
        <xsl:attribute name="value">
          <xsl:choose>
            <xsl:when test="@value = 'foo'">bar</xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="@value"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:attribute>
      </xsl:if>
      <xsl:apply-templates />
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

```
sexml = Enterprise::SEXML DATA.read

xslt = Nokogiri::XSLT(File.read(ARGV[0]))

doc = xslt.transform sexml
puts doc.to_ruby

__END__
class Foo
end
foo = Foo.new
foo.hello
```



http://www.flickr.com/photos/29364131@N07/3119439609/ http://www.flickr.com/photos/psd/2928326253/



http://www.flickr.com/photos/29364131@N07/3119439609/ http://www.flickr.com/photos/psd/2928326253/

W
U
P
S

Poe's Law

- Code as XML?
- The name “Enterprise”
- Everything about this project?

Spiral Downward

WUPS

Enterprise Rails

- Rails isn't Enterprisey enough.
- We "fixed" that.

Enterprise Rails Video

http://www.youtube.com/watch?v=ar2eqEoMUTw

Actual Benefits!

- Several bugs in ruby2ruby and nokogiri were found while working on this.
- I could have fixed these bugs at any time, but I wasn't looking for them.
- Apparently a bad idea is a good reason to fix things.

Bringing it All Together

Phuby on Phails

Phuby on Phails Video

http://www.youtube.com/watch?v=lsWKjS6Vufw

Enterprise Phuby Rails

- We haven't written this yet...
  - It should only take about 30 min.
- How much should we charge for it?

Conclusion

It's OK if your idea is bad

Just practice Good Engineering

You know, for fun!

2009: Worst Year for Ruby Ever.

Together we can make 2010 even worse

Thank You