

红宝石,编码乐

托马斯大卫

Dave Thomas
@/+pragdave
pragprog.com

红宝石, 编码乐

马大卫

Dave Thomas
@/+pragdave
pragprog.com

红宝石， 编码乐

Code Ruby, Be Happy

红宝石， 编码乐

If you Code Ruby,

You will Be Happy

I First Met Ruby
in 1998

I Fell in Love

一见钟情



I Fell in Love

一见钟情



Is it OK to love a
software tool?



You must love your
software tools

My wife and I first
visited China in 2009

We fell in love

2009: Beijing

2011: Hong Kong

2012: Hangzhou &
Shanghai

2013: Beijing &
Hong Kong



We fell
in love

我妻子学习中文

我也在学

现在我们会说
一点儿中文



And...

I think I love China and
Ruby for the same reasons

And...

I think I love China and
Ruby for the same reasons
(almost)

♥ Ambiguity

♥ Honor the past

♥ Embrace the Future

♥ People

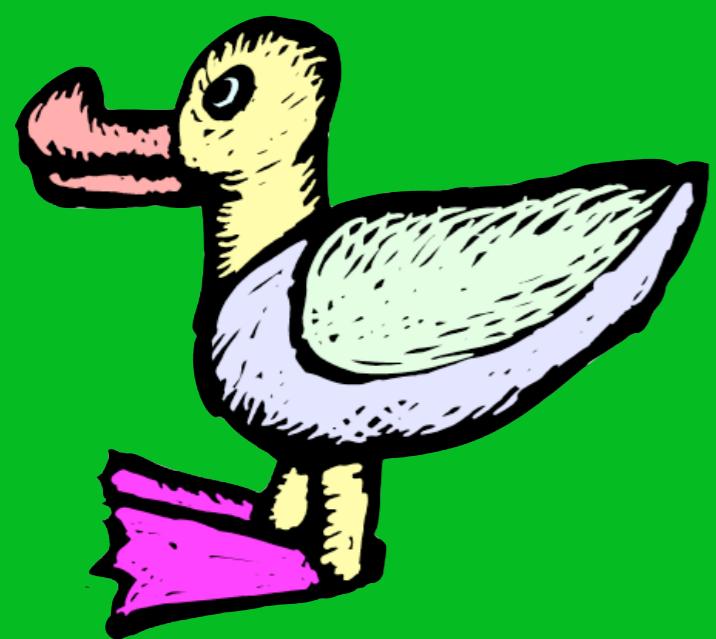
Ambiguity

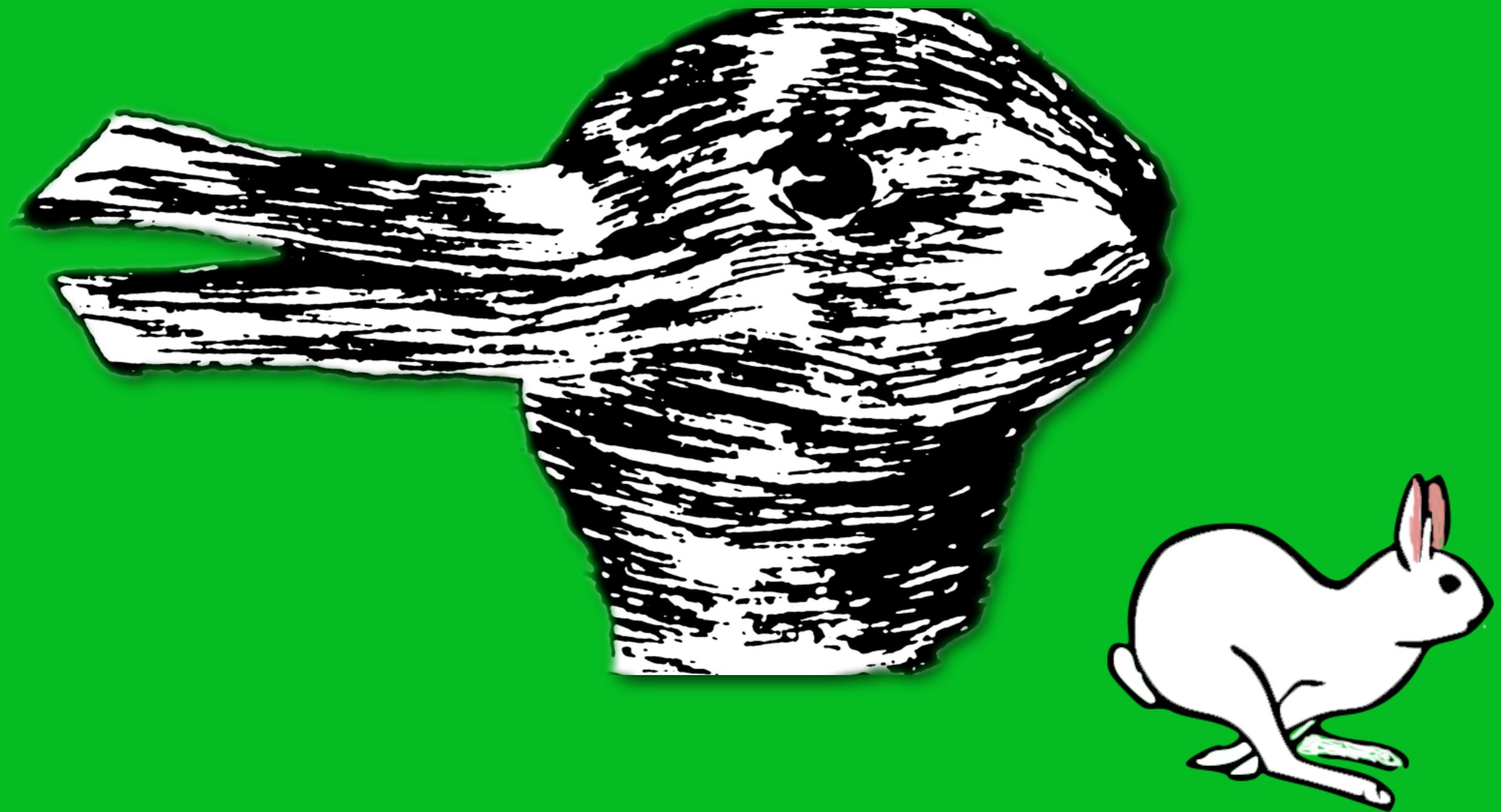
(模棱兩可)

Ambiguity

- The meaning is not 100% certain
- Open to interpretation
- “You get to decide”







中国人



模棱两可

《施氏食狮史》

石室诗士施氏，嗜狮，誓食十狮。

氏时时适市视狮。

十时，适十狮适市。

是时，适施氏适市。

氏视是十狮，恃矢势，使是十狮逝世。

氏拾是十狮尸，适石室。

石室湿，氏使侍拭石室。

石室拭，氏始试食是十狮。

食时，始识是十狮尸，实十石狮尸。

试释是事。



三猿得鹭



三猿得鹭

三元得路



The best poetry
–the best art in any medium–
is ambiguous.

The best poetry
–the best art in any medium–
is ambiguous.

Ambiguity begets participation.

Daniel J. Levitin. *The World in Six Songs*

不确定性引发思考、
讨论和参与

Ambiguity begets participation.

Chinese is ambiguous

Chinese is ambiguous

So is Ruby

Ruby and Ambiguity

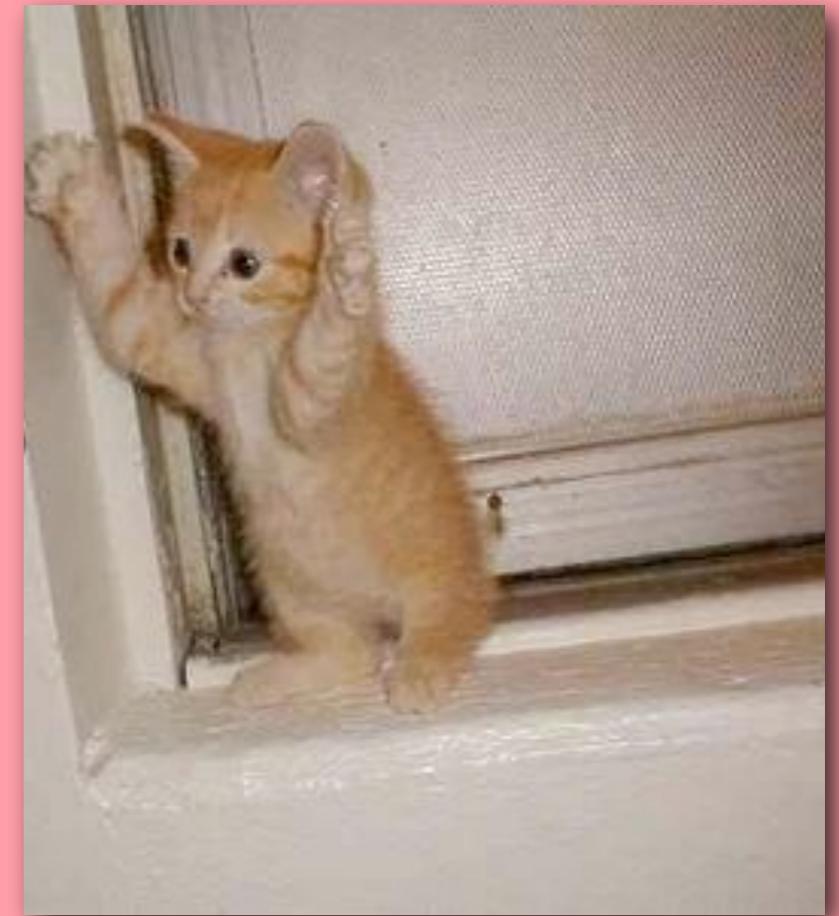
a = 1, a = 2

p a

Ruby and Ambiguity

a = 1, a = 2

p a



Ruby and Ambiguity

$a = 1, a = 2$

p a



Ruby and Ambiguity

```
a = 1, a = 2
```

```
p a # => [1, 2]
```

Ruby and Ambiguity

p a = 1, a = 2

Ruby and Ambiguity

```
p a = 1, a = 2      # => 1  
                    2
```

Ruby and Ambiguity

```
p a = 1, a = 2      # => 1  
                    2
```

```
p a
```

Ruby and Ambiguity

```
p a = 1, a = 2      # => 1  
                    2
```

```
p a                  # => 2
```

Ruby and Ambiguity

```
Integer("3") * 2
```

Ruby and Ambiguity

```
Integer("3") * 2      # => 6
```

Ruby and Ambiguity

```
Integer("3") * 2      # => 6
```

```
Integer ("3") * 2
```

Ruby and Ambiguity

```
Integer("3") * 2      # => 6
```

```
Integer ("3") * 2    # => 33
```

Ruby and Ambiguity

Ruby and Ambiguity

"a" "b"

Ruby and Ambiguity

"a" "b" # => "ab"

Ruby and Ambiguity

"a" "b" # => "ab"

%"a"

Ruby and Ambiguity

"a" "b"
=> "ab"

%"a"
=> "a"

Ruby and Ambiguity

"a" "b"
=> "ab"

%"a"
=> "a"

%"b"

Ruby and Ambiguity

"a" "b"
=> "ab"

%"a"
=> "a"

%"b"
=> "b"

Ruby and Ambiguity

"a" "b"
=> "ab"

%"a"
=> "a"

%"b"
=> "b"

%"a" "b"
=> "a"

Ruby and Ambiguity

"a" "b" **# => "ab"**

%"a" **# => "a"**

%"b" **# => "b"**

%"a" "b" **# => "ab"**

Ruby and Ambiguity

"a" "b" **# => "ab"**

%"a" **# => "a"**

%"b" **# => "b"**

%"a" "b" **# => "ab"**

%"a" %"b"

Ruby and Ambiguity

"a" "b" **# => "ab"**

%"a" **# => "a"**

%"b" **# => "b"**

%"a" "b" **# => "ab"**

%"a" %"b" **# => "a"**

Ruby and Ambiguity

```
if a = 1 && b = 2  
  p a, b  
end
```

Ruby and Ambiguity

```
if a = 1 && b = 2 # => 2
  p a, b
end
```

Ruby and Ambiguity

```
if a = 1 && b = 2  
  p a, b  
end
```

Ruby and Ambiguity

```
if a = 1 && b = 2  
  p a, b  
end
```

Ruby and Ambiguity

```
if a = 1 && b = 2  
  p a, b  
end
```

Ruby and Ambiguity

```
if a = 1 && b = 2      b → 2
  p a, b
end
```

Ruby and Ambiguity

```
if a = 1 && 2           b → 2
  p a, b
end
```

Ruby and Ambiguity

```
if a = 1 && 2           b → 2
  p a, b
end
```

Ruby and Ambiguity

```
if a = 1 && 2           b → 2
  p a, b
end
```

Ruby and Ambiguity

```
if a = 1 && 2           b → 2
  p a, b
end                      a → 2
```

Ruby and Ambiguity

```
if (a = 1) && (b = 2)
  p a, b
end
```

Ruby and Ambiguity

```
if (a = 1) && (b = 2)
  p a, b
end
```

file.rb:1: warning: found = in conditional, should be ==

file.rb:1: warning: found = in conditional, should be ==

Ruby and Ambiguity

```
name { }
```

Ruby and Ambiguity

```
name { }      # pass block to method
```

Ruby and Ambiguity

```
name { }      # pass block to method  
name({ })
```

Ruby and Ambiguity

```
name { }      # pass block to method
```

```
name({ })    # pass hash to method
```

Ruby and Ambiguity

name { } **# pass block to method**

name({ }) **# pass hash to method**

name 1=>2

Ruby and Ambiguity

name { } **# pass block to method**

name({ }) **# pass hash to method**

name 1=>2 **# pass hash to method**

Ruby and Ambiguity

```
name { }      # pass block to method  
name({ })    # pass hash to method  
name 1=>2    # pass hash to method  
name {1=>2}
```

Ruby and Ambiguity

```
name { }      # pass block to method  
name({ })    # pass hash to method  
name 1=>2    # pass hash to method  
name {1=>2}  # syntax error!!!
```

模糊性生参与

Ambiguity begets participation

模糊性生参与

Ambiguity begets participation

- Makes Chinese a richer language and culture

模糊性生参与

Ambiguity begets participation

- Makes Chinese a richer language and culture
- Makes Ruby code more interesting to read and write

模糊性生参与

Ambiguity begets participation

- Makes Chinese a richer language and culture
- Makes Ruby code more interesting to read and write
- *It is good if something you love can still surprise you*

A scenic view of the Great Wall of China winding through lush green mountains under a clear sky.

Honor the Past

China Knows Its Past



Panorama of Departure Herald, painted in Ming Dynasty (1368-1644)
http://commons.wikimedia.org/wiki/File:Departure_Herald-Ming_Dynasty.jpg

China Knows Its Past

History

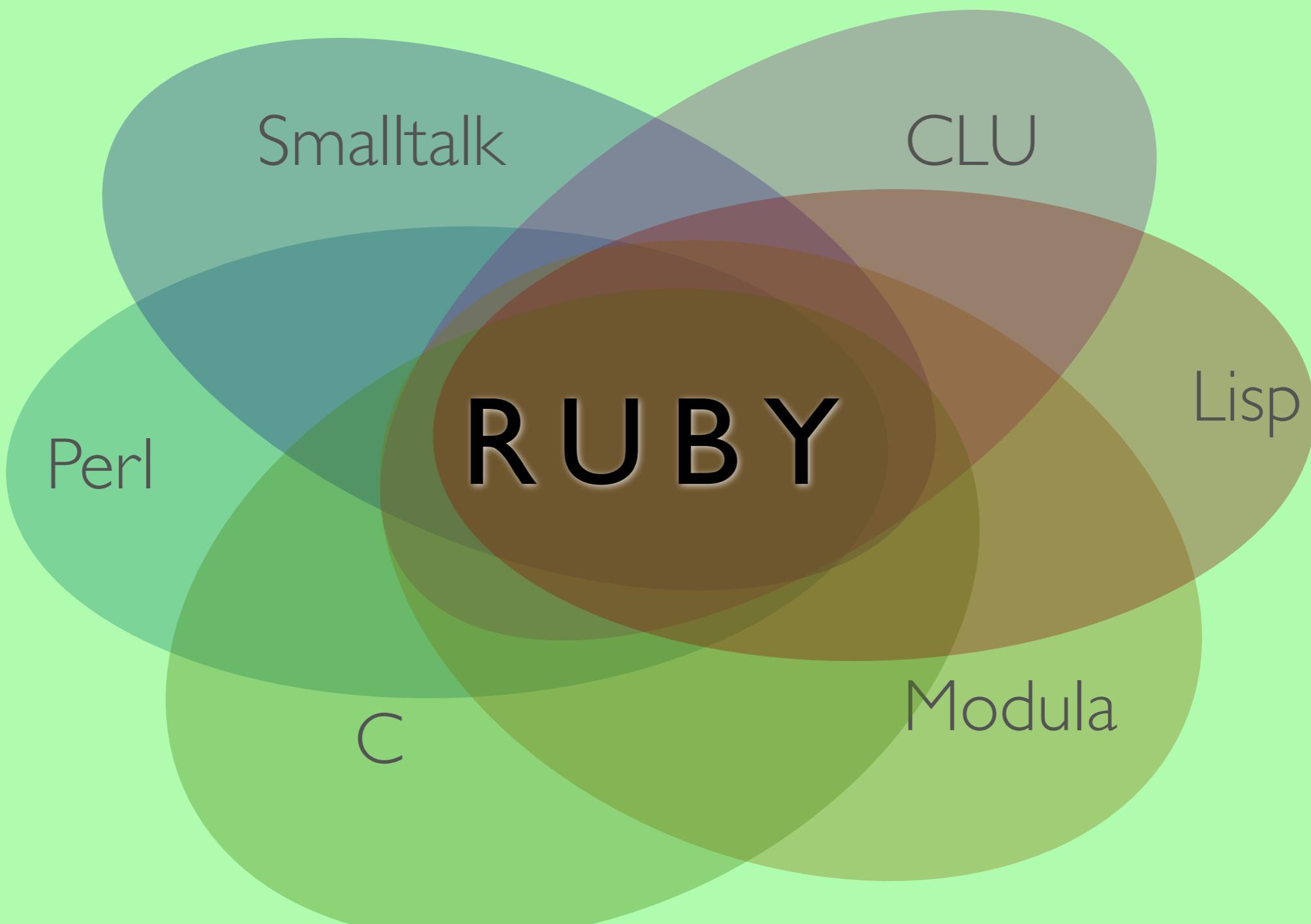


Panorama of Departure Herald, painted in Ming Dynasty (1368-1644)
http://commons.wikimedia.org/wiki/File:Departure_Herald-Ming_Dynasty.jpg

Ruby Knows Its Past

History

Ruby Knows Its Past



Smalltalk

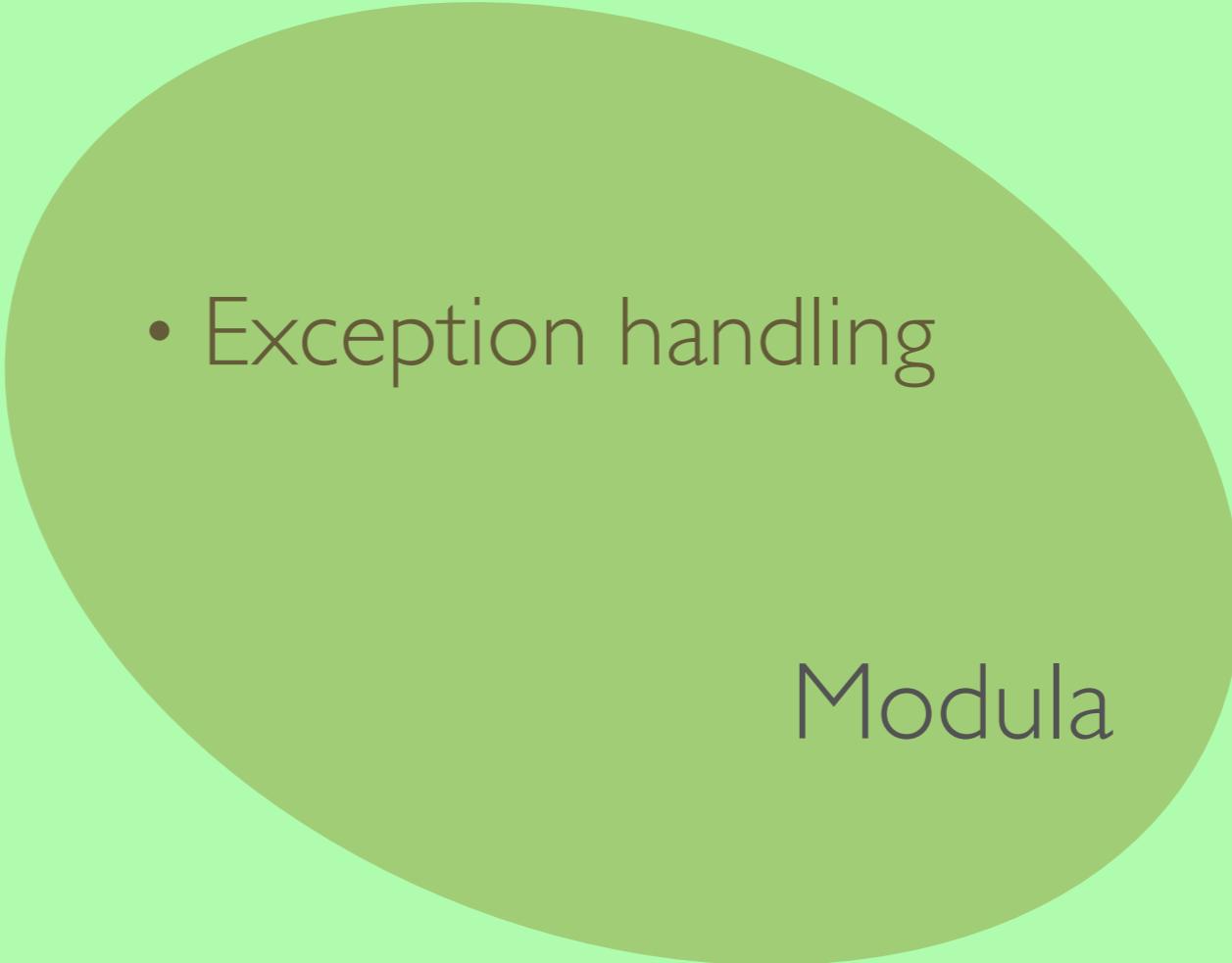
- Metaobject model
- Everything's an object
- Code-block–based programs

Perl

- Lot of syntax
- Operating system integration
- Statement modifiers
- TMTOWTDI

C

- libc etc...



Modula

- Exception handling

Lisp

- Continuations
- Simple data types

CLU

- Iterators
- Object construction (?)

```

getword = iter() yields (string)
  ins:stream := stream$primary_input()
  cs:ac := ac$new()
  w:string  c:char := ' '
  while true do
    while c=' ' do
      c := stream$getc(ins)
    end
    while 0=string$indexc(c, " ") do
      ac$addh(cs,c)
      c := stream$getc(ins)
    end
    w := string$ac2s(cs)
    yield (w)
    ac$trim(cs,1,0)
  end
  except when end_of_file: end
end getword

```

```

for word:string in getword() do
  stream$putl(stdout, word)
end

```

```

getword = iter() yields (string)
  ins:stream := stream$primary_input()
  cs:ac := ac$new()
  w:string c:char := ' '
  while true do
    while c=' ' do
      c := stream$getc(ins)
    end
    while 0=string$indexc(c, " ") do
      ac$addh(cs,c)
      c := stream$getc(ins)
    end
    w := string$ac2s(cs)
    yield (w)
    ac$trim(cs,1,0)
  end
  except when end_of_file: end
end getword

```

```

for word:string in getword() do
  stream$putl(stdout, word)
end

```

```

getword = iter() yields (string)
  ins:stream := stream$primary_input()
  cs:ac := ac$new()
  w:string  c:char := ' '
  while true do
    while c=' ' do
      c := stream$getc(ins)
    end
    while 0=string$indexc(c, " ") do
      ac$addh(cs,c)
      c := stream$getc(ins)
    end
    w := string$ac2s(cs)
    yield (w)
    ac$trim(cs,1,0)
  end
  except when end_of_file: end
end getword

```

```

for word:string in getword() do
  stream$putl(stdout, word)
end

```

```

getword = iter() yields (string)
  ins:stream := stream$primary_input()
  cs:ac := ac$new()
  w:string  c:char := ' '
  while true do
    while c=' ' do
      c := stream$getc(ins)
    end
    while 0=string$indexc(c, " ") do
      ac$addh(cs,c)
      c := stream$getc(ins)
    end
    w := string$ac2s(cs)
    yield (w)
    ac$trim(cs,1,0)
  end
  except when end_of_file: end
end getword

```

```

for word:string in getword() do
  stream$putl(stdout, word)
end

```

```

getword = iter() yields (string)
  ins:stream := stream$primary_input()
  cs:ac := ac$new()
  w:string  c:char := ' '
  while true do
    while c=' ' do
      c := stream$getc(ins)
    end
    while 0=string$indexc(c, " ") do
      ac$addh(cs,c)
      c := stream$getc(ins)
    end
    w := string$ac2s(cs)
    yield (w)
    ac$trim(cs,1,0)
  end
  except when end_of_file: end
end getword

```

```

for word:string in getword() do
  stream$putl(stdout, word)
end

```

1973!

China Knows Its Past

History

China Knows Its Past

History
Tradition



出入平安事事成

大順景方方利









Ruby Knows Its Past

History

Ruby Knows Its Past

History

Tradition

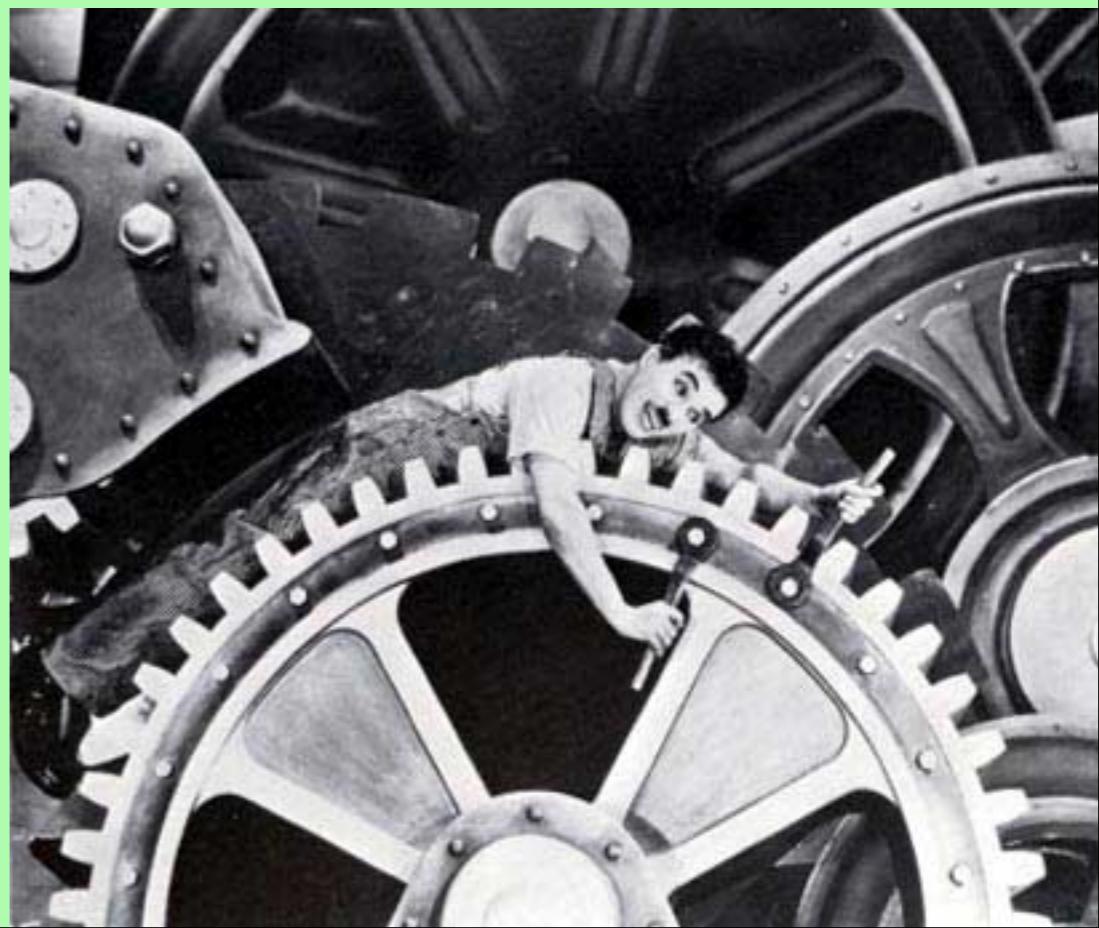
Ruby Knows Its Past

Test-driven development



Ruby Knows Its Past

Test-driven development
Automation



Ruby Knows Its Past



Test-driven development
Automation
Open Source

Ruby Knows Its Past

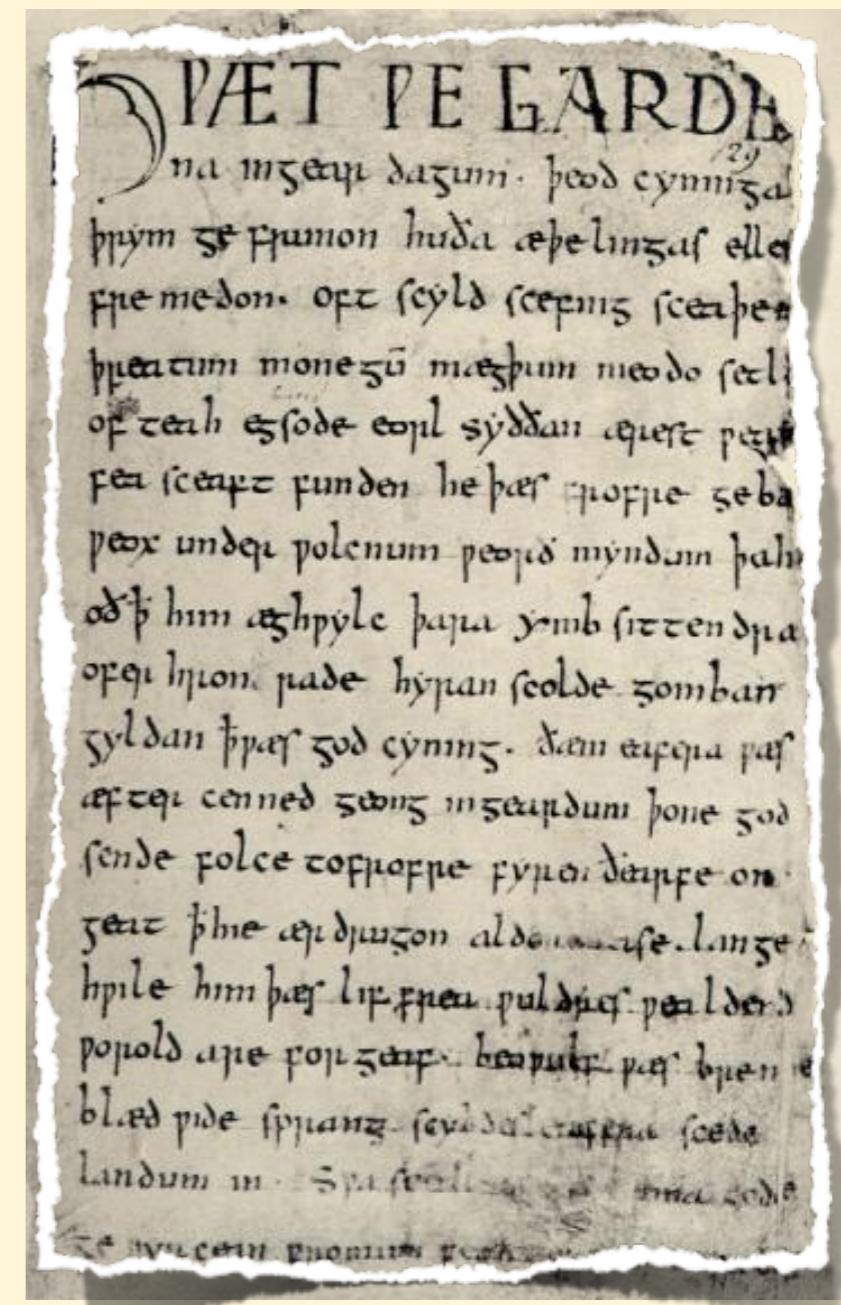
Test-driven development
Automation *Open Source*
Agile Development

个体和互动 高于 流程和工具
工作的软件 高于 详尽的文档
客户合作 高于 合同谈判
响应变化 高于 遵循计划

China Knows Its Past

History
Tradition
Writing

Beowulf ~800–1000AD



China Knows Its Past

History
Tradition
Writing

史記

Records of the Grand Historian

~92BCE



Ruby Knows Its Past

History

Tradition

Ruby Knows Its Past

History

Tradition

Writing

Ruby Knows Its Past

Ruby 0.49
1994

Ruby Knows Its Past

Ruby 0.49
1994

```
freq = {}
while gets()
  while sub(/\w+/, '')
    word = $&
    freq[word] +=1
  end
end

for word in freq.keys.sort
  printf("%s -- %d\n", word, freq[word])
end
```

Ruby Knows Its Past

Ruby 0.49
1994

```
freq = {}
while gets()
  while sub(/\w+/, '')
    word = $&
    freq[word] +=1
  end
end

for word in freq.keys.sort
  printf("%s -- %d\n", word, freq[word])
end
```

Ruby Knows Its Past

Ruby 0.49
1994

```
freq = {}
while gets()
  while sub(/\w+/, '')
    word = $&
    freq[word] +=1
  end
end

for word in freq.keys.sort
  printf("%s -- %d\n", word, freq[word])
end
```

Ruby Knows Its Past

Ruby 0.49
1994

```
freq = Hash.new(0)
while gets()
  while sub(/\w+/, '')
    word = $&
    freq[word] +=1
  end
end

for word in freq.keys.sort
  printf("%s -- %d\n", word, freq[word])
end
```

Ruby Knows Its Past

Ruby 0.49
1994

Ruby Knows Its Past

Ruby 0.49
1994

```
class Foo
  attr("test", %TRUE)
end

foo = Foo.new
foo.test = 10
print(foo.test, "\n")
foo._inspect.print
print("\n")
```

Ruby Knows Its Past

Ruby 0.49
1994

```
class Foo
  attr("test", %TRUE)
end

foo = Foo.new
foo.test = 10
print(foo.test, "\n")
foo._inspect.print
print("\n")
```

Ruby Knows Its Past

Ruby 0.49
1994

```
class Foo
  attr("test", %TRUE)
end

foo = Foo.new
foo.test = 10
print(foo.test, "\n")
foo._inspect.print
print("\n")
```

Ruby 1.0
1996

Ruby 1.0
1996

```
sieve = []
if ! max = ARGV.shift; max = 100; end
max = max.to_i

print "1"
for i in 2 .. max
begin
  for d in sieve
    fail if i % d == 0
  end
  print ", "
  print i
  sieve.push(i)
rescue
end
end
print "\n"
```

Ruby 2.0 2013

```
max = Integer(ARGV.shift || 100)
sieve = []
for i in 2 .. max
  sieve[i] = i
end

for i in 2 .. Math.sqrt(max)
  next unless sieve[i]
  (i*i).step(max, i) do |j|
    sieve[j] = nil
  end
end
puts sieve.compact.join(", ")
```

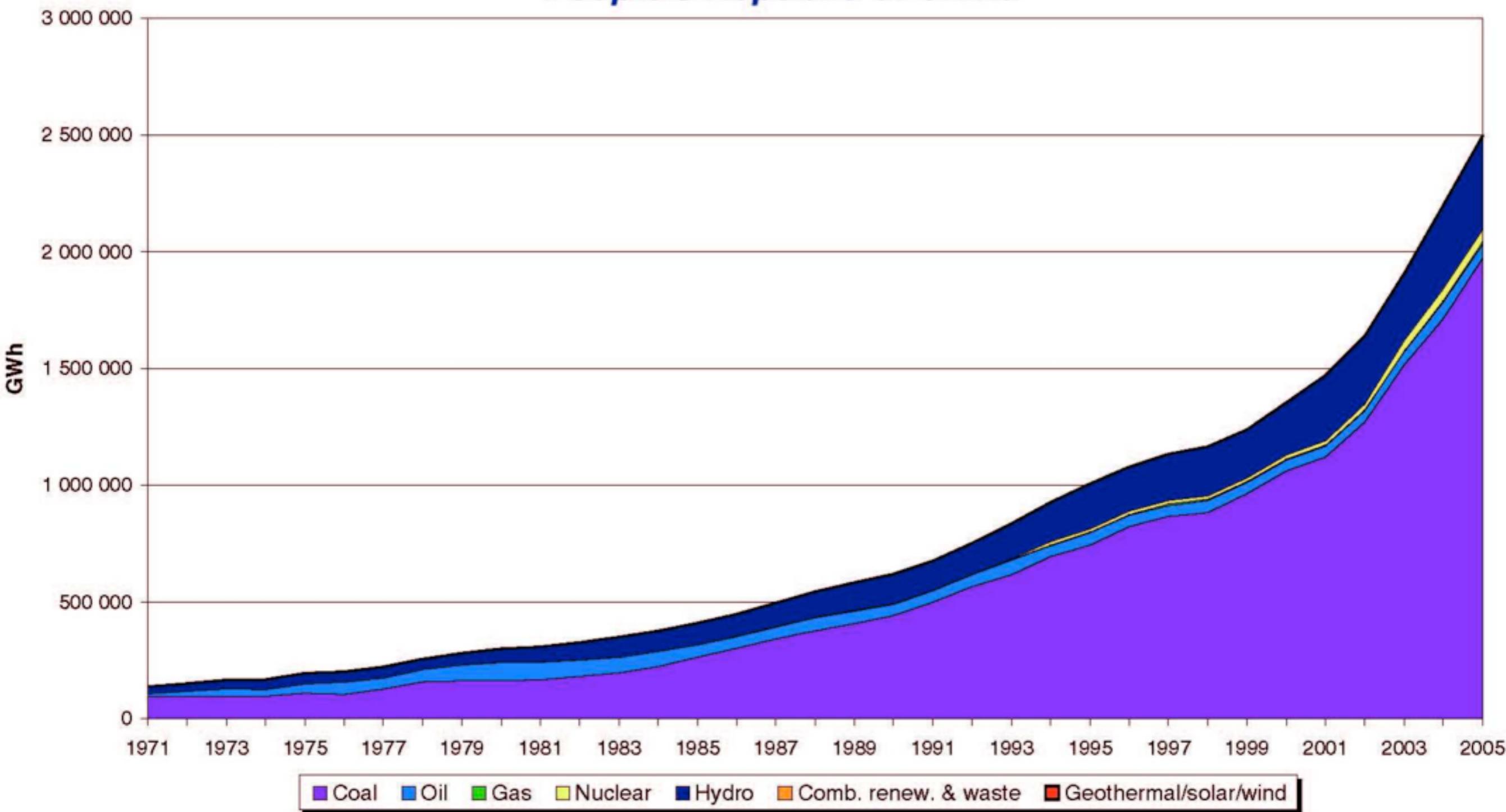
Embrace the Future





Evolution of Electricity Generation by Fuel from 1971 to 2005

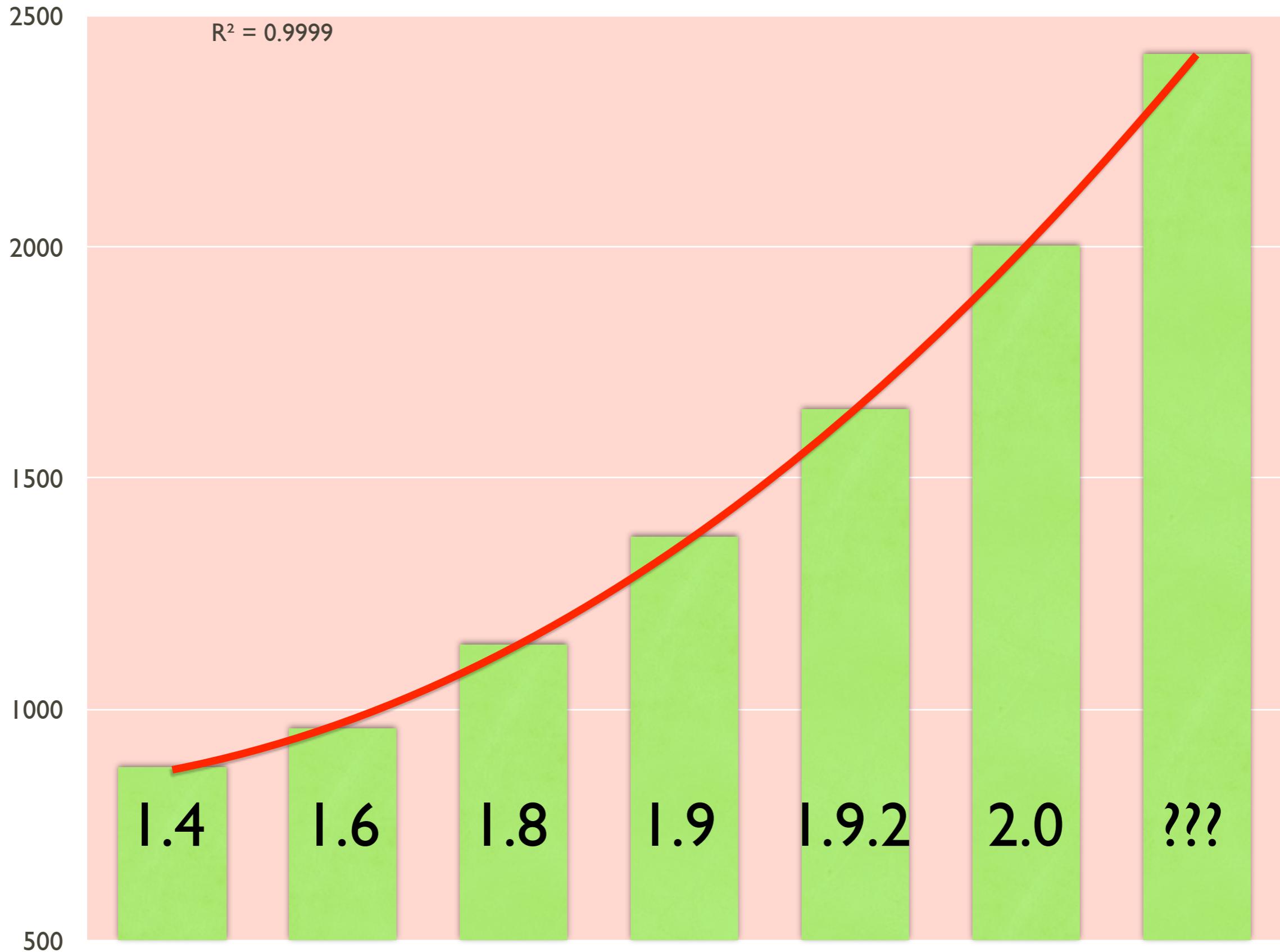
People's Republic of China



■ Coal ■ Oil ■ Gas ■ Nuclear ■ Hydro ■ Comb. renew. & waste ■ Geothermal/solar/wind

Number of API methods

Number of API methods



MRuby

- Embedded, modular Ruby
 - similar in use to Lua
- Large Subset of the language
- Ruby in your washing machine, vending machine, router... !

<https://github.com/mruby/mruby>

Ruby is not Cool

Ruby is not Cool

(太好了！)

Ruby is not Cool

(太好了！)

- Things changed quickly when Ruby was small

Ruby is not Cool

(太好了！)

- Things changed quickly when Ruby was small
- Now things change more slowly, and it is less risky to use

Ruby is not Cool

(太好了！)

- Things changed quickly when Ruby was small
- Now things change more slowly, and it is less risky to use
- Less about people, more about code

The People





Ruby too...

Matz



The Ruby Community



Matz's Rules

Be nice to developers

Be clear and readable

Be flexible and agile

Be open

**China & Ruby
have much
in common**

**Let's Change
the World**

**Let's Change
the World**

**Let's Make
it Better**

謝謝大家

