RubyConf China

Why Ruby?

Yukihiro "Matz" Matsumoto
まつもと ゆきひろ
matz@ruby-lang.org

thou

Moore’s Law

The number of Transistors in LSI Doubles Every 18 Months

Moore’s Law Means:

Computer Grows Exponentially

- Faster
- Cheaper
- More Common

Faster Computer

PCs are Faster than Super Computers of 20 Years Ago

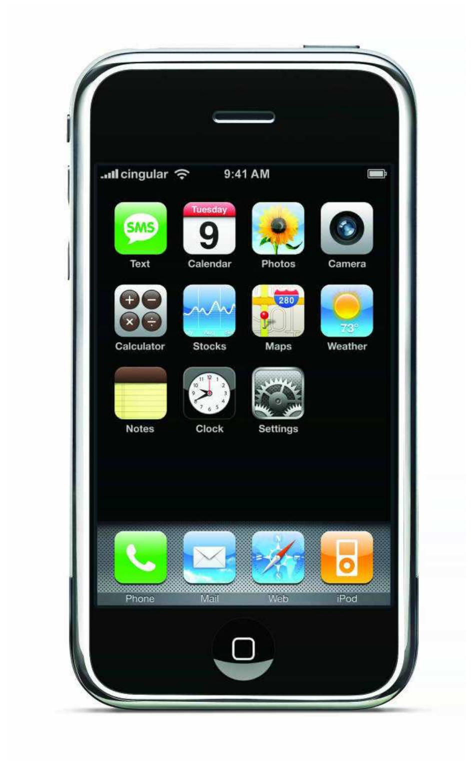
Cheaper Computers

We can Buy a PC for $400 Now

Common Computers

- Now Everyone Owns Computers
  - Personal Computers
  - Cell Phones

Cell Phone as a Computer



Cell Phone as a Computer

Everyone is Connected

- Broadband
- WiFi
- Mobile Networks

Influence in Programming

Moore’s Law Changes

- Software Complexity
- Programming Languages

Software Complexity

- No Business Can Be Run without Software
- We Need More Software
  - Quicker
  - Cheaper

Humans Don’t Improve

- Moore’s Law Does Not Apply to Humans

Productivity

- We Need More Software with Limited Resources

Productivity

- We Have Faster Computers
- Development Efficiency At the Cost of Runtime Efficiency

Productivity

- The Most Important Factor of Language Evolution
- Languages are One of the Tools for Productivity

How Languages Help Productivity

Sapir-Whorf hypothesis

Language determines the way we think.

Theorem #1

Languages influence
human thought,
more than you think

Programming Languages

Do programming languages influence human thoughts?

Thinking in Programming Language

- Natural languages are too ambiguous.
- Or, too verbose.
- Or, too indirect.

Thinking in Programming Language.

If programmers think in
programming languages,
They must influence thoughts
as much as
natural languages do.

Theorem #2

"languages" in
Theorem #1 includes programming
languages.

Why don't you choose a good language?

Programming languages are so easy to learn.

What is a good language?

The language that helps thinking

Recursion

BASIC did not allow recursion

Factorial

```
def fact(n)
  if n == 0
    1
  else
    n * fact(n - 1)
  end
end
print "6!=", fact(6), "\n"
```

```
6!=740
```

A good Language

does not restrict our thought

Factorial Again

```
print "200!=", fact(200), "\n"
```

```
200!=78865786736479050355236321393218506
2295135977687173263294742533244359449963
4033429203042840119846239041772121389196
3883025764279024263710506192662495282993
1113462857270763317237396988943922445621
4516642402540332918641312274282948532775
2424240757390324032125740557956866022603
1904170324062351700858796178922222789623
7038973747200000000000000000000000000000
00000000000000000000
```

Less Restriction

```
print "200!=", fact(200), "\n"
```

```
200!=788657867364790503552363213932185062295
13597768717326329474253324435944996340334292
03042840119846239041772121389196388302576427
90242637105061926624952829931113462857270763
31723739698894392244562145166424025403329186
41312274282948532775242424075739032403212574
05579568660226031904170324062351700858796178
92222278962370389737472000000000000000000000
00000000000000000000000000000
```

Ruby

A good language

Consise

Succinctness is Power

by Paul Graham

- Fred Brooks’ Law
- Less Lines ≒ Effective

Succinctness is Power

- Less Code, Less Bugs
- Less Bugs, You Feel Yourself Smarter.
- You can be 10 times (or even 1000 times) more productive

More Factorial

```
class Factorial {
  private static int fact(int n) {
    if (n == 1) return 1;
    return n * fact(n - 1);
  }
  public static void main(String[] argv) {
    System.out.println("6!="+fact(6));
  }
}
```

```
6!=740
```

Succinctness Example

```
def fact(n)
  if n == 1
    1
  else
    n * fact(n - 1)
  end
end
print "6!=", fact(6), "\n"
```

```
6!=740
```

Ruby

A good language

Inspiring

Functional Factorial

```
def fact(n)
  (1..n).inject(:*)
end
print "6!=", fact(6), "\n"
```

```
6!=740
```

A good language

makes better programming experience

For Better Programming Experience

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

According to Dr. Jacob Nielsen

Learnability

How easy is it for users to accomplish basic tasks the first time they encounter the design?

Learnability

- Usability for Beginners
- Important to Acquire New Users
- "Common Sense" is the Key

Efficiency

Once users have learned the design, how quickly can they perform tasks?

Efficiency

- More important than learnability
- Efficiency is the top purpose of languages

Memorability

When users return to the design after a period of not using it, how easily can they reestablish proficiency?

Memorability

- Association
- Consistency
- Orthogonality
- Common Sense
- No Radical

Errors

How many errors do users make, how severe are these errors, and how easily can they recover from these errors?

Errors

- When you see repeated errors, you have to do something.
- Errors are the source of design inspiration.

Satisfaction

How pleasant is it to use the design?

Satisfaction

- We program to have fun.
- Even when we program for money, we want to have fun as well.

How Ruby Serves

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

How Ruby Serves

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

Learnability

Ruby is very conservative except for a few places

- Quick to learn
- Quick to try

Learnability Example

Hello World!

```
print "Hello World\n"
Hello World
```

How Ruby Serves

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

Efficiency

No Run-Time Efficiency

Ruby focuses on the cost of programming.

Devlopment Efficiency

Ruby focuses on the cost of programming by

- Simplicity
- Consistency
- Smartness

Simplicity

Do you like programming language to be simple?

- Probably you do.

Simplicity

Does language simplicity really help you?

- not always.
- need more complex tool sometimes

Need More Complex Tool

- Knife vs Chain Saw
- Bicycle vs Airplane

Human Heart: No Simple

- We love simplicity
- We love complexity
- We love easy problems
- We hate easy problems

Pseudo-Simplicity

Ruby is NOT a simple language.

Simplicity Example

Rakefile

- Rake = Ruby Make

```
task :default => [:test]
task :test do
  ruby "test/unittest.rb"
end
```

Simplicity Example

In Simpler Syntax

```
task({:default => [:test]})
task(:test, lambda(){
  ruby "test/unittest.rb"
})
```

Solution-Simplicity

Tool Complexity is OK

if it makes the Solution Simple

Efficiency Example

/bin/cat in Ruby

```
  puts ARGF
```

It would be more than 50 lines of code in C

How Ruby Serves

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

Memorability

- Conservativeness helps here too
- Easy-to-remember syntax
  - Ruby looks like other languages

Memorability Example

Can you write /bin/cat -n without looking anything?

I can, if I use Ruby.

```
ARGF.each_with_index{|line,i|
  printf "%4d %s",i,line
}
```

How Ruby Serves

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

Errors

You will see less errors due to

- Consistent Syntax Rules
- Succinct Code
  - Less code, Less bug.

How Ruby Serves

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

Satisfaction

As a result, Ruby is fun to use.

It makes you feel smarter.

Ruby the Language

Ruby is Good for

- Text Processing
- Web Programming
- XML Programming
- GUI Applications

Ruby is Good for

- Bioinformatics
- eXtreme Programming

"I love it. Conceptually it is really clean and sweet."
-- Kent Beck

Why I created Ruby

- Just for Fun
- Tool for me myself
- Ideal tool for Everyday Task

How I created Ruby

- Combine Good Things from the Past
- Design Conservative
- Design a Tool I Want to Use
- To Have Fun

The History of the Ruby Language

Pre-History

- OO Fanboy
- Language Geek

In 1993

- Project Started
- Mere Hobby

Goals

- Scripting
  - a la Perl
- Nice Clean Syntax
- With OO

Real Goal

To Enjoy

- Making Language
- Implementation
- Programming

Process

- Lisp Semantics
- Smalltalk OO
- Conservative Syntax

Process

- Deconstruct Perl
- Reorganize into Class Library

Process

- Iterators from CLU
- Higher-order Functions using Blocks

Process

- Some Spice from Python
- ..and other languages

Released

1995-12-21

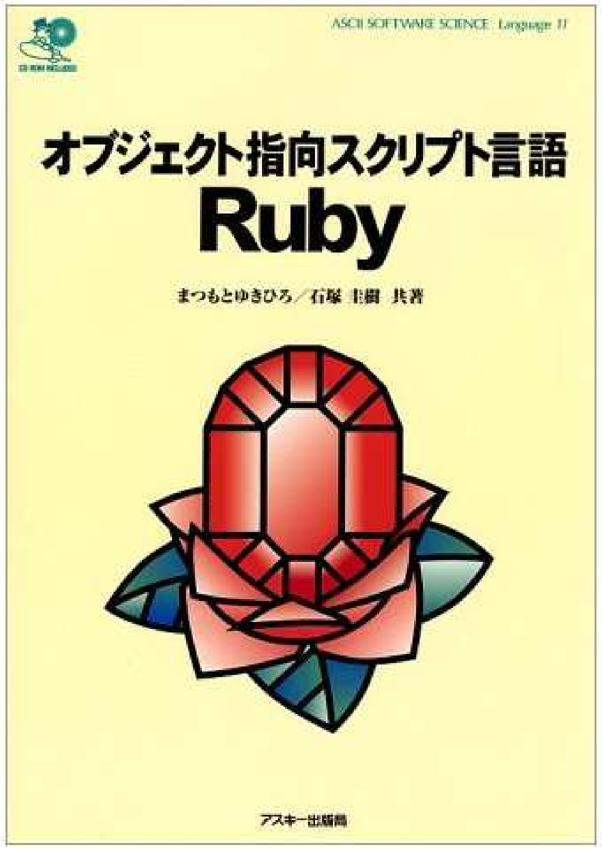
fj.sources

In 1997

- Hired by NaCl
- Became Full-time OSS Developer

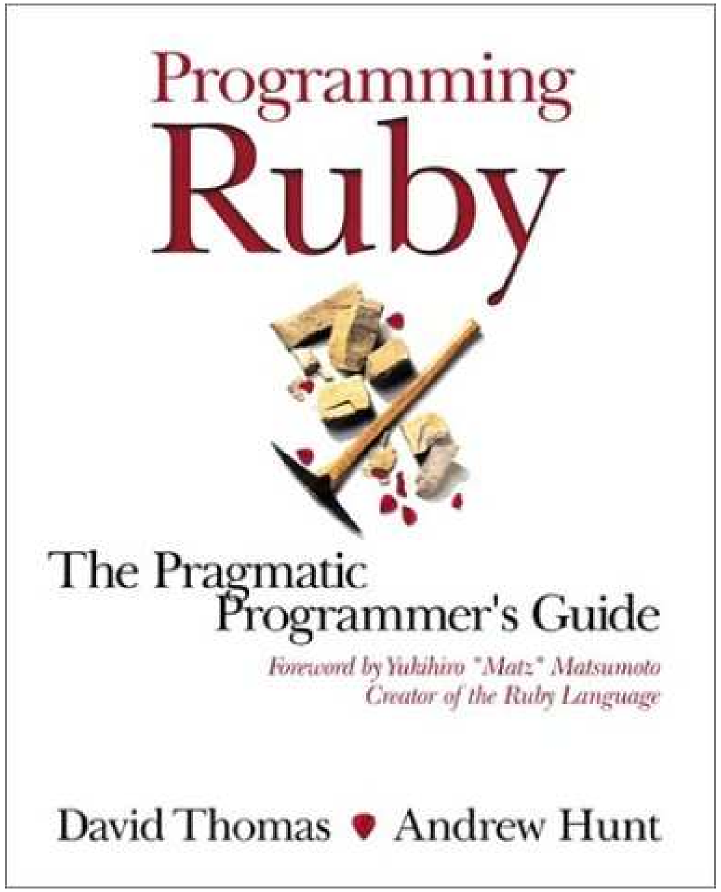
In 1999

First Book


ASCII SOFTWARE SCIENCE Language 11
オブジェクト指向スクリプト言語
Ruby
まつもとゆきひろ／石塚圭樹 共著
アスキー出版局

In 2000

First English Book


Programming
Ruby
The Pragmatic Programmer's Guide
Foreword by Yukihiro "Matz" Matsumoto
Creator of the Ruby Language
David Thomas
Andrew Hunt

Ruby in early 2000s

Became a Language
for Geeks

In 2004

Ruby on Rails

Ruby on Rails

Web Application Framework

Web development that doesn’t hurt

Ruby on Rails

- 10x Productive than Java
- 15 Minutes to code Blog

Enterprise Ruby

Ruby started to be used in the Enterprise Environment

Enterprise Ruby

Concerns

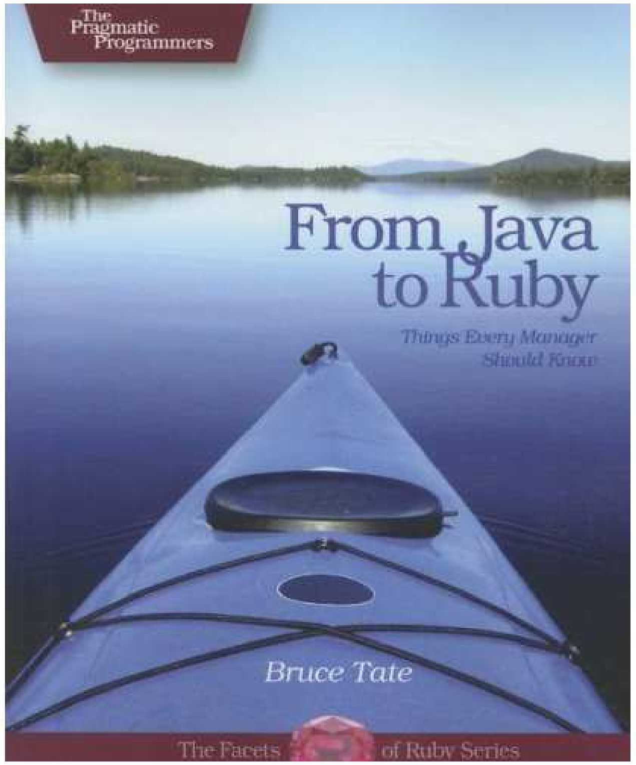
- Fast Enough?
- Scales?

Enterprise Ruby

Issues are Matters of Resource/Money We Put in.

Ruby’s Mindshare

From Java To Ruby


The Pragmatic Programmers
From Java to Ruby
Things Every Manager Should Know
Bruce Tate
The Facets of Ruby Series

What’s "Enterprise"?

What Major Players Recommend:

- Sun Microsystems
- Microsoft
- Apple
- Etc.

Why Ruby?

- Ruby is Productive
- Ruby is Motivating
- Ruby is Fun

A Message from Ruby

Enjoy Programming!