Keynote

Yukihiro "Matz" Matsumoto
まつもと ゆきひろ
@yukihiro_matz

Today's Menu

- History
- Future
- Diversity

History

10th Annual RubyConf

History

9 Keynotes by me

2001

Tampa, FL

2001

Human-Oriented Programming in Ruby

2002

Seattle, WA

2002

Be Minor, Be Cool

2003

Austin, TX

2003

Visions for the Future

How Ruby Sucks

2004

Chantilly, VA

2004

2004

2005

San Diego, CA

2005

Visions for the Future

Wild and Weird Ideas

2006

Denver, CO

2006

The Return of the Bikeshed or Nuclear Plant in the Backyard

2007

Charlotte, NC

2007

Language Matters

2008

Orlando, FL

2008

Reasons behind Ruby

2009

San Francisco, CA

2009

The 0.8 True Language (ZEPT)

2010

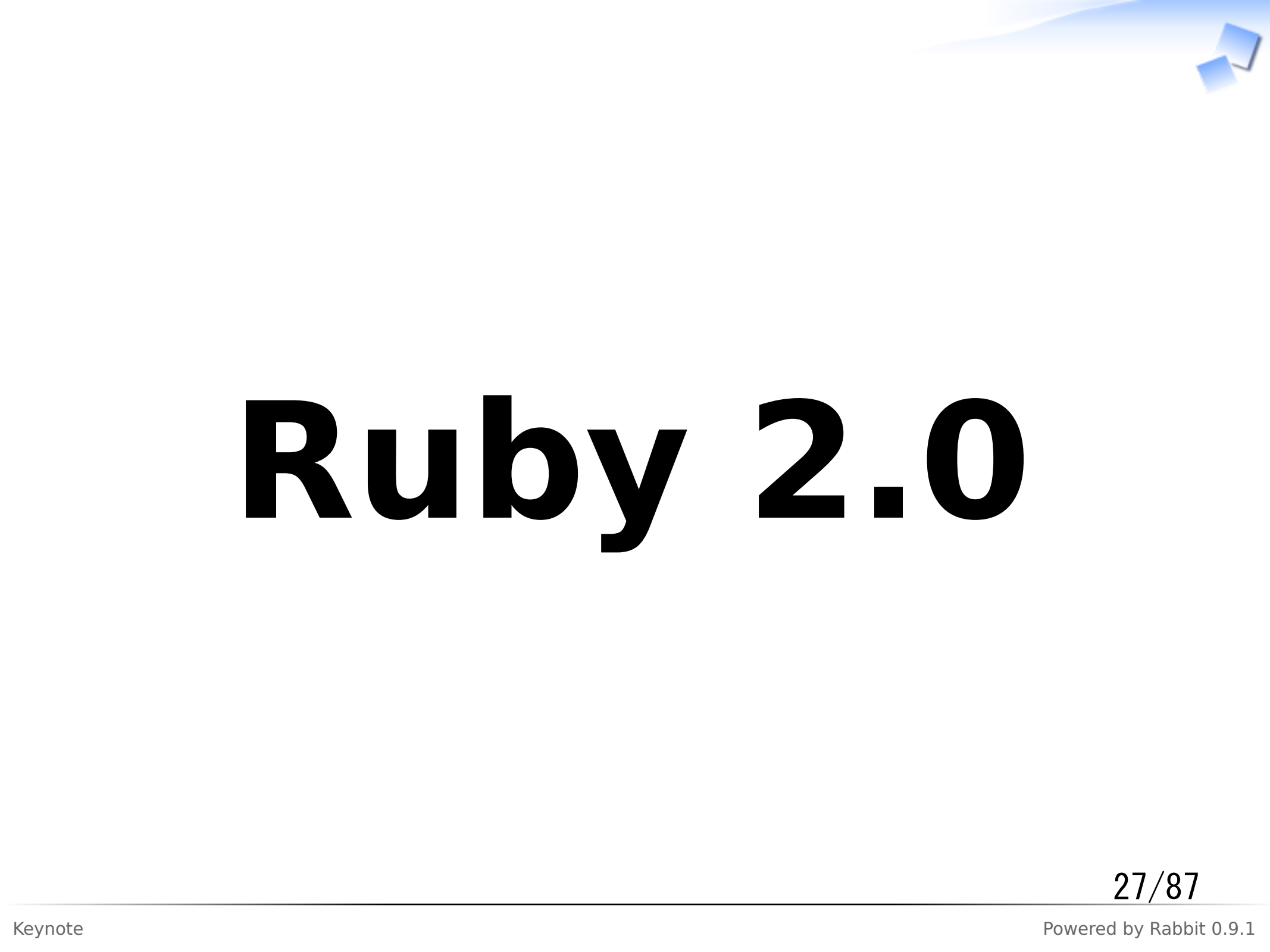
New Orleans, LA

2010

Future and Diversity

Future and Diversity

The Future


Ruby 2.0

Ruby 2.0

- Traits
- Method Combination
- Keyword arguments
- Namespaces
- a few other nifty features

Traits

trait

a trait is a collection of methods, used as a "simple conceptual model for structuring object oriented programs".

from Wikipedia (en)

What's wrong for Modules?

- Conflict detection
- Conflict resolution
- Tree modification
- No method combination

Conflict Detection

name conflict

- intentional (overriding)?
- or accidental?

Conflict Problem

```
module American
  attr_accessor :address
end
module Japanese
  attr_accessor :address
end
class JapaneseAmerican
  include American
  include Japanese
end
JapaneseAmerican.new.address
# which address?
p JapaneseAmerican.ancestors
# => [JapaneseAmerican, Japanese, American, Object, Kernel]
```

Solution

- We will introduce #mix
- #mix will replace #include
- #mix can detect and resolve conflict

Module#mix

- injects the current snapshot into other class/module.
- raises error when name conflict
- unless you resolve it explicitly

Conflict Problem

```
module American
  attr_accessor :address
end
module Japanese
  attr_accessor :address
end
class JapaneseAmerican
  # Japanese comes First
  include American
  include Japanese
end
```

Detecting Conflict

```
module American
  attr_accessor :address
end
module Japanese
  attr_accessor :address
end
class JapaneseAmerican
  mix American
  mix Japanese  # => address conflict!
end
```

Resolving Conflict

```
class JapaneseAmerican
  mix American, :address => :us_address
  mix Japanese, :address => :jp_address
end
```

Tree Modification

```
module M1; end
class C1; include M1; end

module M2; end
module M1; include M2; end

p C1.ancestors
# [C1, M1, Object, Kernel]
p M1.ancestors
# [M1, M2]
```

inconsistent

Tree Modification

- #mix copies attributes
- so tree modification afterward does not affect.
- consistent at leaset

alias_method_chain

- ugly
- fragile to multiple wrapping
- we want to wrap methods

Module#prepend

- We will introduce #prepend
- #prepend put the module before the current class/ module
- methods defined in the class will wrap methods of same names

Module#prepend

```
module Foo
  def foo
    p :before
    super
    p :after
  end
end
class Bar
  def foo
    p :foo
  end
  prepend Foo
end
Bar.new.foo # :before, :foo, :after
```

Keyword Arguments

calling

```
1.step(by: 2, to: 20) do |i|
  p i
end
```

Keyword Arguments

defining

```
def step(by: step, to: limit)
  ...
end
```

Keyword Arguments

- Mere expanded hash argument at the end
- Automatic decomposition

Namespaces

encapsulation of monkey patching

- monkey patching is global modification
- embodies freedom, but dangerous

Namespaces

encapsulation of monkey patching

- classsbox / selector namespace / refinement / whatever

What if

```
class Integer
  def /(other)
    return quo(other)
  end
end
p 1/2 # => (1/2)
```

Allow Refinement

```
module MathN
  refine Integer do
    def /(other)
      return quo(other)
    end
  end
  p 1/2 # => (1/2)
end
p 1/2   # => 0
```

Using Refinement

```
module Rationalize
  using MathN
  p 1/2 # => (1/2)
end
p 1/2 # => 0
```

Real Private Methods

```
class HasPrivate
  module Private
    def priv
    end
  end
  using Private
  def pub
    priv
  end
end
h = HasPrivate.new
h.priv  # => error
h.instance_eval {
  priv  # => error
}
```

FAQ

When will they be available?

Ruby 2.0

FAQ

When will Ruby2.0 be?

Christmas

on whatever year!

Diversity

I love Diversity

I dislike Diversity

The Ruby Language

- specification
- implementation

The Ruby Language

specification

- ✓ Standard Ruby (ISO)
- ✓ RubySpec

The Ruby Language

implementation

- ✓ CRuby
- ✓ JRuby
- ✓ Rubinius
- ✓ MagLev
- ✓ . . .

Alternative to fill the Niche

- JRuby for JVM
- MacRuby for Mac
- MagLev for GemStone
- Ruboto for Android

Yet another Niche

Embedding

Rite

- The New Comer
- Light weight implmentation
- of usable subset of the Ruby language

Target

Embedding

- Small devices
- Digital Appliances
- Applications (Game?)

and more

Embeddable Ruby

- think of Lua
- with better language

Principle

- Components
- Configurable

Components

the implementation will be combination of components

- parser
- virtual machine
- garbage collector
- debugger
- class libraries

Configurable

- to minimal set of features required for an application
- no universal behavior between platforms
  - e.g. no file I/O for small devices

Configurable

- use double or float
- use int, long or long long for fixnums
- ASCII or UTF-8

Requirement

- portable
  - minimal requirement: standard C (C99)
- should run on PC / RTOS / free standing
- less memory
- less latency

Implementaion Detail

- register-based virtual machine
- 32bit word-code
- floats are immediate
- (possibly generational) incremental mark-sweep GC

What can I do with Rite?

embedding

- ✓ application embedding
- ✓ small devices

  e.g. digital TV

What can I do with Rite?

concurrent

- ✓ assign virtual machine for each thread

the Ruby chip

- by Prof. Tanaka from Kyushu Institute of Technology
- MIPS-like FPGA CPU with a few instructions added
  - that help method look-up
  - and garbage collection marking

FAQ

When will Rite available?

I don't know, sorry.

But it's a part of Japanese government funded two year project (2010-2011)

FAQ

Will Rite be Open-Source?

Yes, probably under MIT license.

But we need business model to satisfy the government.

FAQ

Will Rite be Open-Source?

We might choose GPL plus commercial subscription model (a la MySQL).

FAQ

Will Rite replace MRI?

No, Rite will not be a full-featured, universal implementation.

FAQ

Will Rite replace MRI?

It is a Domain Specific Implementation, like Ruboto.

FAQ

How about C API?

Rite will have different C API from CRuby.

Currently we have no plan to provide compatibility layer.

FAQ

Will Rite support M17N?

No, you have to configure single character encoding from ASCII or UTF-8 in compile time.

FAQ

Will Rite support (native) threads?

No, to use threads you can use multiple VM per native threads. Rite may support fibers in the future.

FAQ

Does Rite run faster than YARV/JRuby/Rubinius, etc?

Probably Not, but maybe on some benchmarks due to float immediate values and other techniques.

FAQ

How can I contribute to Rite?

Wait until we make it open-source. We will open it on github.

FAQ

Rite sounds familier

Originally Rite was a code name for the first Ruby 2.0 virtual machine, which was replaced by YARV. It's coind from Ruby Lite.

FAQ

Do you resign from CRuby?

No, but I have spent less time on CRuby recently anyway.

FAQ

Do you resign from CRuby?

I will keep being a maintainer of CRuby. And bove all, I will keep being active as the creator of the language and, the leader of the community.

Thank you!