Ruby on Rails

Web Development that doesn‘t hurt

Dezember 2008

www.xing.com/profile/**Christian_Feser**
www.xing.com/profile/**Michael_Kram**
www.xing.com/profile/**Jakob_Schroeter**
www.**Marc-Seeger**.de

Ruby

A Programmer‘s Best Friend

agenda

- Design History of Ruby
- Agile Manifesto
- Language basics
- Exercise
- Typing
- Libraries & Gems
- Ruby VMs
- Good & bad things

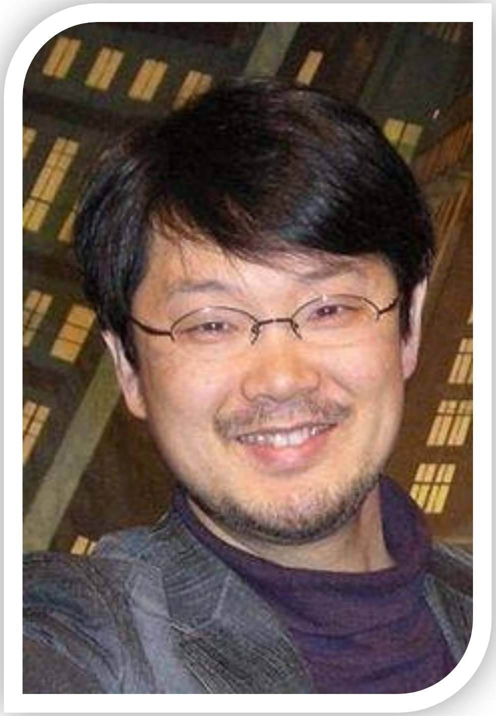
Design History of Ruby

Ruby *is* different from other languages, indeed.

--Matz.

History

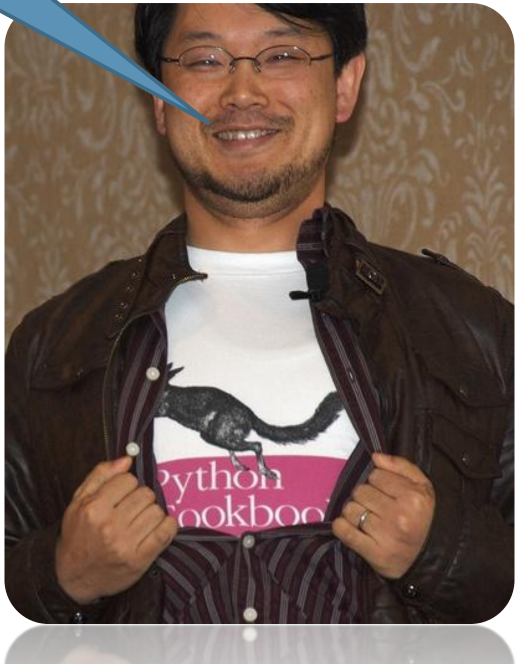
- Origin:
  - Yukihiro "Matz" Matsumoto
  - Japan 1993



- 1st english book: 2000
- Ruby on Rails: 2004

What Matz has to say

I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language



The Power of Ruby

… according to Yukihiro „Matz“ Matsumoto

Ruby is

Smalltalk
– „Unfamiliar Syntax“
+ Perl‘s scripting power
+ Pythons exceptions etc.
+ CLU‘s iterator

This helps making ruby…

- a Scripting Language
- a dynamic typing Language
- an Object Oriented Programming Language
- a good taste of Functional Programming
- a highly reflective Language
- a base for creating Domain Specific Languages

Can‘t I do all of that in Java/C?

Sapir-Whorf-Hypothesis:

- Language determines the way we think
  - Basic Programmers never use recursion
  - LISP programmers use macros for everything
  - FORTRAN programmers can write FORTRAN in any language

Ruby = human-oriented

- reduces the burden of programming.
- tries to push jobs back to machines.
- You can accomplish more tasks with less work
- … in smaller yet readable code.

principle of least surprise

I believe people want to express themselves when they program. They don't want to fight with the language.

Programming languages must feel natural to programmers. I tried to make people enjoy programming and concentrate on the fun and creative part of programming when they use Ruby.

Programming experience
(according to Dr. Jacob Nielson)

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

How Ruby helps you: Learnability

```
puts "Hallo HdM"
```

How Ruby helps you: Efficiency

- Not that fast to execute…
- BUT fast to programm
    - Pseudo-simplicity
    - Consistency
    - „Smartness“

How Ruby helps you: Memorability

- Conservativeness helps
- Easy to remember syntax
- Ruby is NOT a simple language …
- … BUT the complexity is:
  - Hidden
  - Natural
  - Consistent

How Ruby helps you: Errors

- You won‘t see that many because:
  - Consistent syntax rules
  - less code → less bugs

How Ruby helps you: Satisfaction

- Ruby is fun
- Ruby makes you feel smart ☺

Ruby in five E's

- **Everything is an object**
- **Elegant blocks give inline power**
- **Exploring with reflection**
- **Extending at runtime**
- **Extensive standard libray**

David Heinemeier Hansson

RUBY | OPENSSH | RUBY ON RAILS | OPENSSL | RUBY GEMS | SONAR
LINUX JOURN
UNDER-ICE Sonar Visualizations
Ruby
Everywhere, Including Rails
Getting Started with Ruby
Ruby the Enterprise Glue
Rails and More Rails
Ruby Has Many Gems
Ruby on Rails creator << David Heinemeier Hansson
PLUS
Creating a Searchable Site
Mashups
Game Rubies

Agile Manifesto

Painless Programming

Ruby – An Agile Language?

- the language design should focus on users
- the language should encourage readability
- the language should be expressive, and helps communication between developers
- the language should embrace changes, and be dynamic

(Matz@Rubyconf2006)

Language Basics

A Programmer‘s Best Friend

Hello World!

```
#include <stdio.h>
int main( int argc, char **argv ) {
        puts( "Hello, world!" );
        return( 0 );
}
```

Read the following aloud

```ruby
5.times { print "Hello HdM!" }


exit unless "text".include? "food"
```

Language Basics: comments

```
# this is a comment
# a = b - c
a = b + c # comment at the end

=begin
  def my_method
    ...
  end
=end
```

Language Basics

- ClassNames
- method_names and variable_names
- methods_asking_a_question?
- slightly_dangerous_methods!
- @instance_variables
- $global_variables
- SOME_CONSTANTS or OtherConstants

Language Basics

- Variable Declaration:

```
text = "Hallo Welt" <== String
zahl = 3.5 <== Float
bla = 3 <== Fixnum
blubb = 2325234586346536456456456 <== Bignum
```

- Function Declaration:

```
def do_something(text, number)
  puts text * 3
  puts number * 3
end
```

Language Basics: Strings

```ruby
str = "Hello" # Hallo
str = "Hello 'HdM' " # Hallo 'HdM'
str = 'Hello' # Hallo
str = 'Hello "HdM"' # Hallo "HdM"

%q{string in curly braces}
%q(string in parenthesis)
%Q$string in dollar symbols$
```

Language Basics: Strings

```
"slash: \\"           # slash: \
"shout: \"HdM!\""     # shout: "HdM!"
'Chris\'s ruby?'      # Chris's ruby?
"new\nline"           # New
                      # line
'New\nline'           # New\nline
```

Language Basics: Strings

```
s1 = "HdM"
s2 = "HdM"
s3 = "FUF"
s1 == s2        # => true
s1 == s3        # => false
s1.equal? s1 # => true
s1.equal? s2 # => false
s1.equal? s3 # => false
```

Language Basics: Strings

```
"Thomas und Ralf".delete("a")          # Thoms und Rlf
"Thomas und Ralf".delete("aou")        # Thms nd Rlf
"Thomas und Ralf".gsub("und", "oder")# Thomas oder Ralf
"Thomas und Ralf".gsub(/[aou]/, "$") # Th$m$s $nd R$lf
```

Language Basics: Hashes

```
h = { 'dog' => 'wuff', 'cat' => 'miau', 'donkey' => 'ihah' }
h.length                »3
h['dog']                »"wuff"
h['cow'] = 'muh'
h['cat'] = 7
h               »{"cow"=>"muh", "cat"=>7, "donkey"=>"ihah", "dog"=>"wuff"}
```

Language Basics: Arrays

```
a = [ 3.14159, "pie", 99, "Blubb" ]
a.type          »Array
a.length        »3
a[1]            »"pie"
a[4]            »nil
a[-1]           »"Blubb"
a[-2]           »99
a[1, 3]         »["pie", 99, "Blubb"]
a[0..2]         »[3.14159, "pie", 99]

b = Array.new
b.type          »Array
b.length        »0
b[0] = "second"
b[1] = "array"
b               »["second", "array"]
```

Give me some sugar: Array

```
people = Array.new
people << "Marc" << "Christian" << "Jakob" << "Michael"
people = ["Marc", "Christian", "Jakob", "Michael"]
people.push("Marc", "Christian", "Jakob", "Michael")
people = %w("Marc", "Christian", "Jakob", "Michael")
```

Control Structures

```
if expr [then]
  expr...
[elsif expr [then]
  expr...]...
[else
  expr...]
end
```

```
until expr [do]
         ...
end
```

```
for i in [1, 2, 3]
  puts i*2
end
```

```
puts "Error!" if $debug
```

```
puts "Error!" unless $production_mode
```

Language Basics: Classes

```ruby
class Project
  def initialize(name)
    @name = name
  end
end


project = Project.new("Learn Ruby")
```

Language Basics: Classes

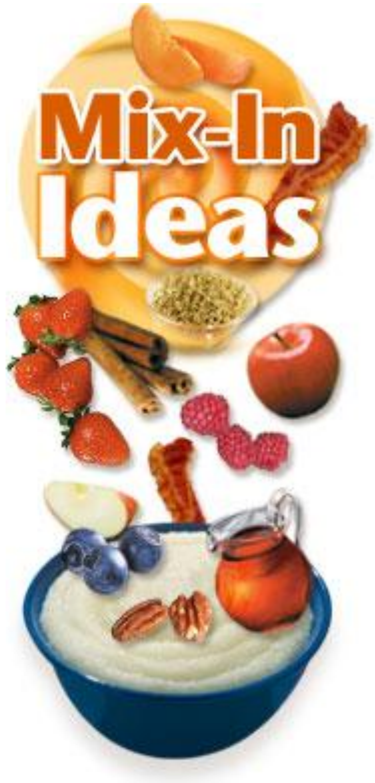
- Classes are always open (even built in classes)

```
class String
 def foo
 "foo"
 end
end

puts "hdm test".foo ==> "foo"
```

Another Example from Rails:
1.hour.from_now

Language Basics: Inheritance

- Single Inheritance
  - But mixins are available (= Interface with implemented methods)



Language Basics: Mixin Example

```
module BarModule
  def hello_world
    puts "Hello World"
  end
end

class BaseClass
  def class_method
    puts "In class method"
  end
end

class Foo < BaseClass
  include BarModule
end

f = Foo.new
f.class_method
f.hello_world
```

← This module implements the mixin

←A class that doesn‘t do that much

←inheriting
←and mixing!

←We inherited that one
←And mixed in that one

Blocks

„Blocks are unnamed functions“

Blocks

Define:

```
def foo &proc
  proc.call 2
  proc.call 4
  proc.call 6
end
```

```
def foo
  yield 2
  yield 4
  yield 6
end
```

```
--------------------------------
```

Call:

```
foo{|some_number|
  puts some_number * 3
}

--------------------------------
```

Result:

```
6
12
18
```

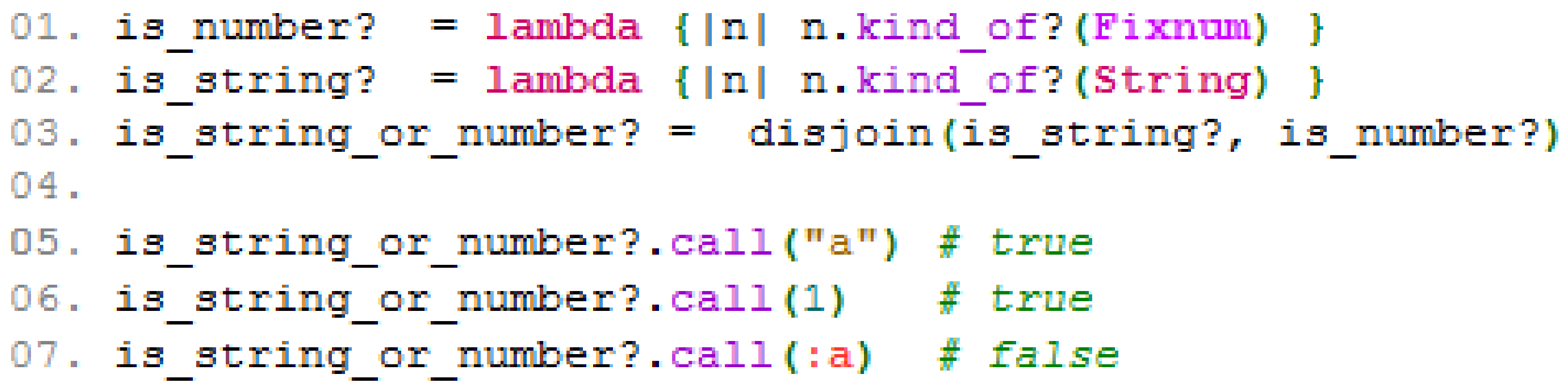
Blocks Example: Iterators

The current piece of the collection we are working with

What we are going to do with it

```
some_collection.each { |item| puts item }

some_collection.select { |item| item =~ /[xz]/ }
some_collection.reject { |item| item =~ /[xz]/ }
```

Closures



```
is_number?  = lambda {|n| n.kind_of?(Fixnum) }
is_string?  = lambda {|n| n.kind_of?(String) }
is_string_or_number? =  disjoin(is_string?, is_number?)

is_string_or_number?.call("a") # true
is_string_or_number?.call(1)   # true
is_string_or_number?.call(:a)  # false
```

Language Basics: IO

```ruby
file = File.open("config.cfg")
lines = file.readlines
file.close
lines.each do |line|
  puts line
end
```

Exercise

→ http://tryruby.hobix.com ←

Typing

…and why do you call Ruby „dynamic“?

Typing: strong / weak

- Strong typing
  - " 5 " / 2 ➔ „NoMethodError“
- Weak typing
  - " 16 " / 2 ➔ 8 (e.g. in Perl)

Ruby is strongly typed! (Java too)

Typing: explicit/implicit

- **Explicit:** int a = 5
- **Implicit:** a = 5

Ruby is implicitly typed! (Java explicitly)

Typing: static / dynamic

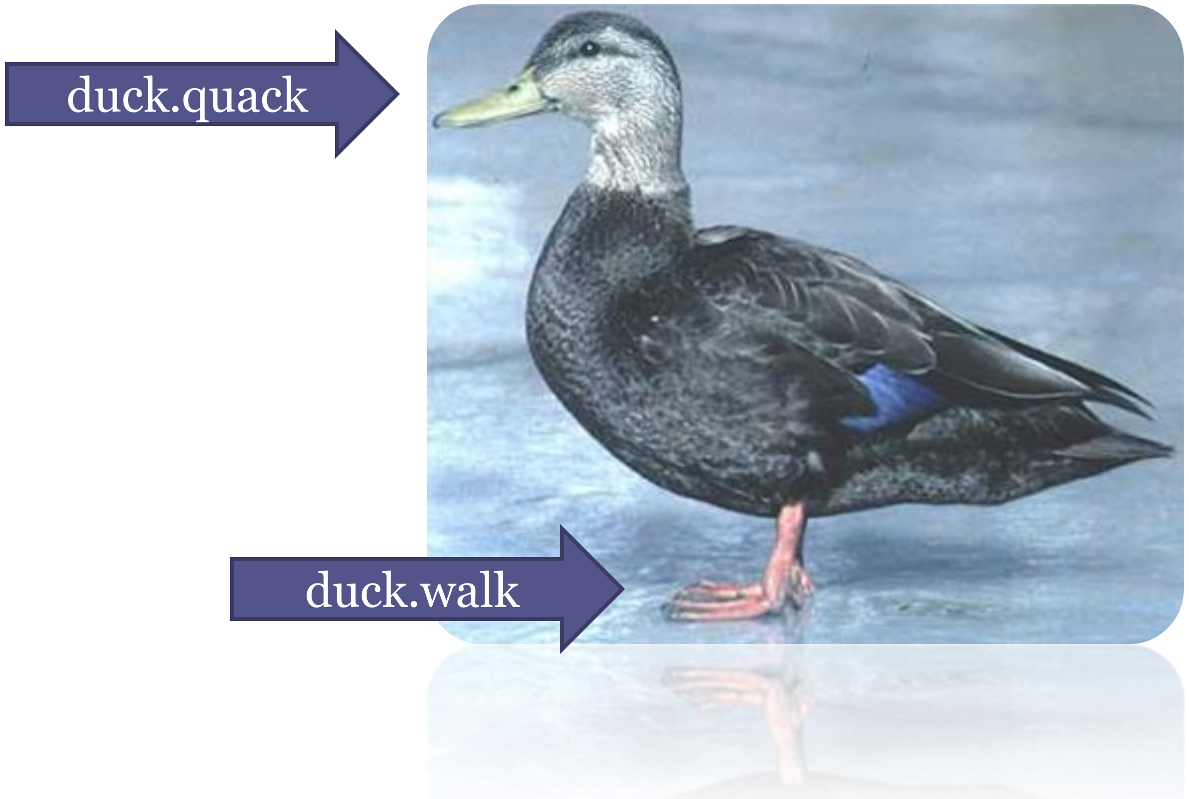
- Static typing
  - The compiler checks types during compilation
- Dynamic typing
  - The compiler doesn‘t check types during compilation

Ruby uses dynamic typing

Java uses static typing,
C# 4.0 will feature the concept of 'Dynamic lookup' ("foreach()" already uses it)
Visual Basic allows you to do both

Typing: duck?!

If it walks like a duck and quacks like a duck, I would call it a duck.



Some other languages supporting duck-typing:
- C# 4.0 will
- Groovy
- Javascript
- Perl
- Python
- Smalltalk (no types for variables)
- VB

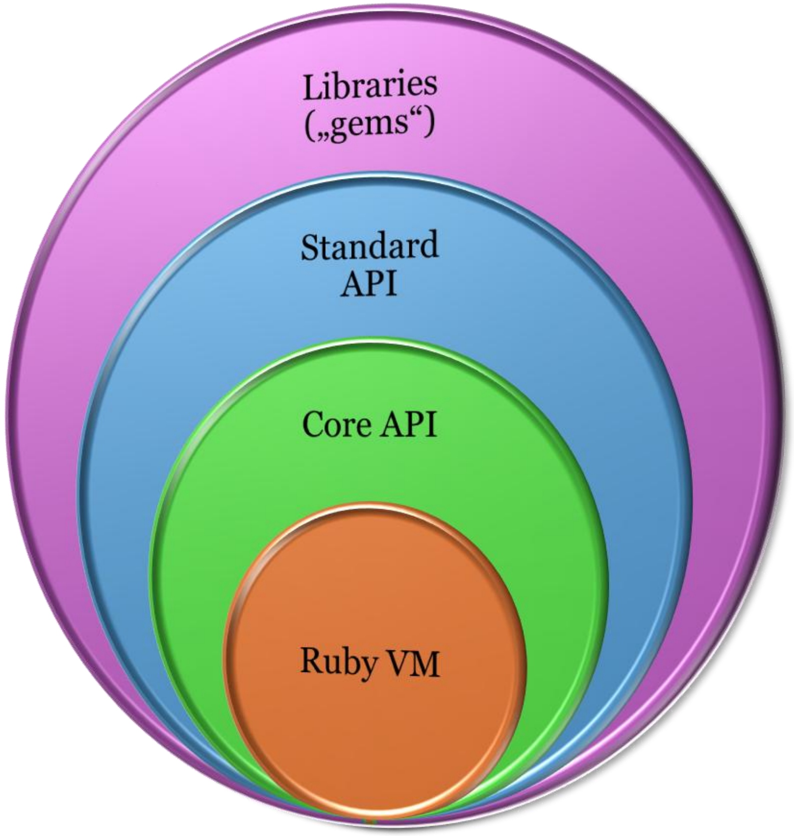
Typing: duck

- Duck typing allows an object to be passed in to a method that expects a certain type even if it doesn’t inherit from that type. All it has to do is support the methods and properties of the expected type *in use by the method*.

Libraries & Gems

Because you don‘t want to do all of the work yourself

The ruby world



Libraries

```
require "libs/my_model.rb"
```

Rubygems

Gems - search

```
> gem search -l xml

*** LOCAL GEMS ***

libxml-ruby (0.9.2, 0.9.0, 0.8.3)
xml-mapping (0.8.1)
xml-object (0.9.8)
xml-simple (1.0.11)
```

Gems - search

```
> gem search -r xml

*** REMOTE GEMS ***

axml (0.0.2)
diff2xml (0.0.2)
eimxml (0.0.2)
faster_xml_simple (0.5.0)
fastxml (0.1.92)
gccxml_gem (0.9.1)
hashtoxml (0.0.5)
jrexml (0.5.3)
libxml-feed (0.0.1)
libxml-ruby (0.9.4)
libxml-xmlrpc (0.1.5)
[...]
```

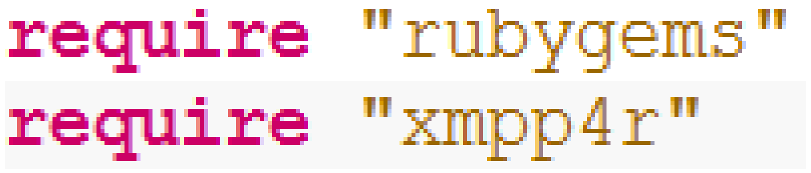
Gems - installing

```
> gem install textgraph
Successfully installed textgraph-0.1.0
1 gem installed
Installing ri documentation for textgraph-0.1.0...
Installing RDoc documentation for textgraph-
  0.1.0...
```

Gems - Updates

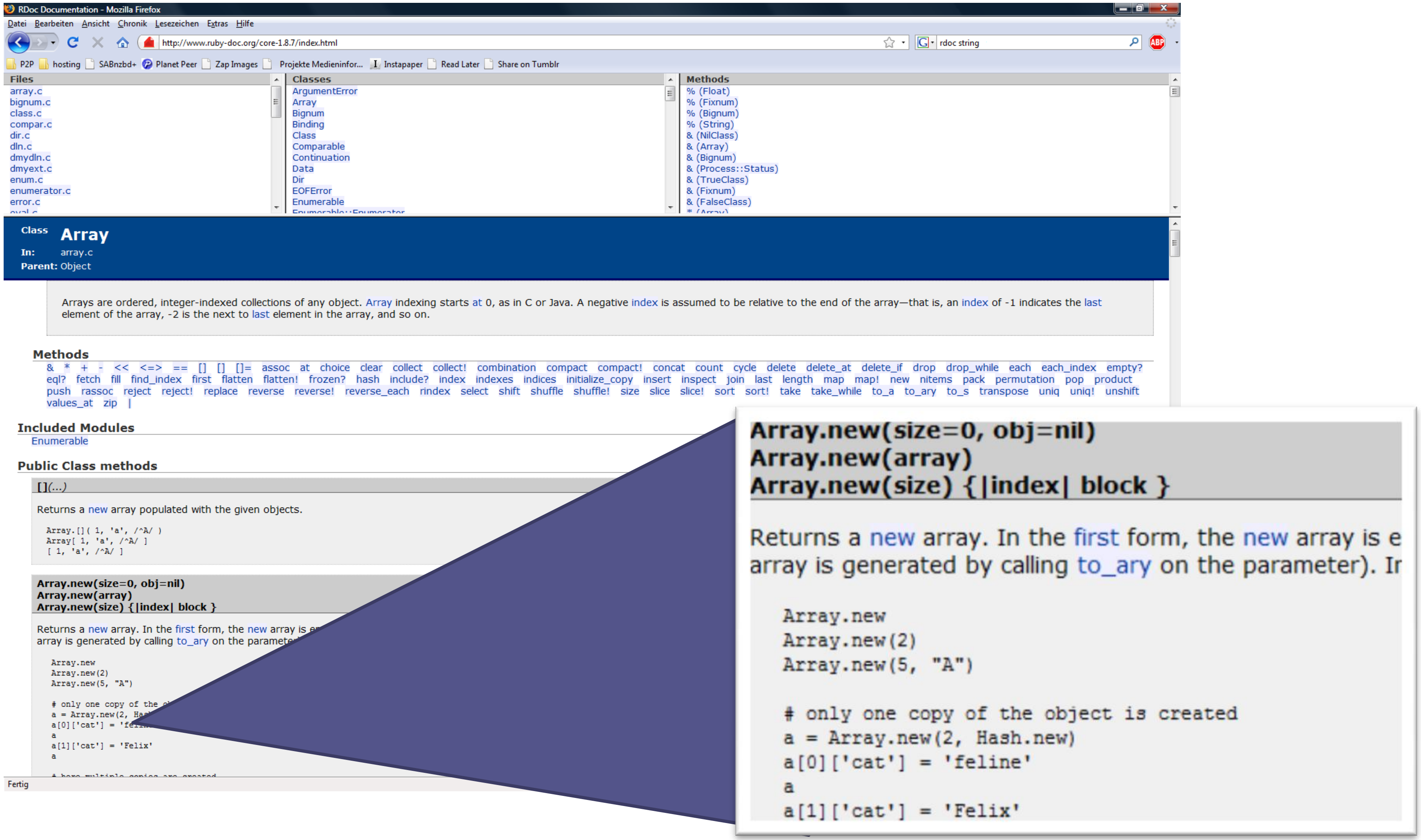
```
> gem update

Updating installed gems
Updating haml
Successfully installed haml-2.0.5
Updating libxml-ruby
Building native extensions.  This could take a while...
Successfully installed libxml-ruby-0.9.4-x86-
  mswin32-60
Updating ruby-debug-ide
Successfully installed ruby-debug-ide-0.4.2
[...]
```

Gems - Usage



```ruby
require "rubygems"
require "xmpp4r"
```

RDoc



RDoc Documentation - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

http://www.ruby-doc.org/core-1.8.7/index.html

rdoc string

P2P hosting SABnzbd+ Planet Peer Zap Images Projekte Medieninfor... Instapaper Read Later Share on Tumblr

Files	Classes	Methods
array.c	ArgumentError	% (Float)
bignum.c	Array	% (Fixnum)
class.c	Bignum	% (Bignum)
compar.c	Binding	% (String)
dir.c	Class	& (NilClass)
dln.c	Comparable	& (Array)
dmydln.c	Continuation	& (Bignum)
dmyext.c	Data	& (Process::Status)
enum.c	Dir	& (TrueClass)
enumerator.c	EOFError	& (Fixnum)
error.c	Enumerable	& (FalseClass)

Class **Array**
In: array.c
Parent: Object

Arrays are ordered, integer-indexed collections of any object. Array indexing starts at 0, as in C or Java. A negative index is assumed to be relative to the end of the array—that is, an index of -1 indicates the last element of the array, -2 is the next to last element in the array, and so on.

Methods

& * + - << <=> == [] [] []= assoc at choice clear collect collect! combination compact compact! concat count cycle delete delete_at delete_if drop drop_while each each_index empty? eql? fetch fill find_index first flatten flatten! frozen? hash include? index indexes indices initialize_copy insert inspect join last length map map! new nitems pack permutation pop product push rassoc reject reject! replace reverse reverse! reverse_each rindex select shift shuffle shuffle! size slice slice! sort sort! take take_while to_a to_ary to_s transpose uniq uniq! unshift values_at zip |

Included Modules

Enumerable

Public Class methods

[]*(...)*

Returns a new array populated with the given objects.

```
Array.[]( 1, 'a', /^A/ )
Array[ 1, 'a', /^A/ ]
[ 1, 'a', /^A/ ]
```

Array.new(size=0, obj=nil)
Array.new(array)
Array.new(size) {|index| block }

Returns a new array. In the first form, the new array is e
array is generated by calling to_ary on the parameter). Ir

```
  Array.new
  Array.new(2)
  Array.new(5, "A")

  # only one copy of the object is created
  a = Array.new(2, Hash.new)
  a[0]['cat'] = 'feline'
  a
  a[1]['cat'] = 'Felix'
  a
```

Fertig

Ruby

VMs

Ruby VMs

- Ruby 1.8 („MRI“-Matz's Ruby Interpreter)
- Ruby 1.9 („YARV“)
- JRuby
- Rubinius
- (IronRuby)

Speed

Rubinius < Ruby 1.8 < JRuby < Ruby 1.9

Ruby 1.8

- **Matz's Ruby Interpreter** or **Ruby MRI**

- Performance: Code -> Syntax Tree -> Run
- Threading: Green
- Unicode: no (utf8 though)

Ruby 1.9

- **YARV** (Yet another Ruby VM)

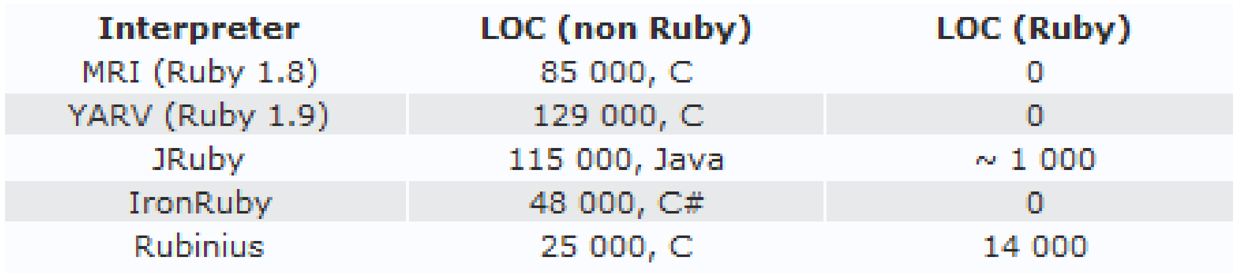
- Performance: Code -> Syntax Tree -> ByteCode
- Unicode: yes
- Threading: Native (but “global interpreter lock“)
- small syntax changes
- no more continuations

JRuby

- Performance: Hotspot!
- Threading: Native
- Unicode: no (utf8 or java unicode)
- Compatibility: 1.8 and 1.9!
- Addon:
  - Deployable on Tomcat/Glassfish/…
  - Access Java from Ruby or Ruby from Java
- Problems: C Libraries

Rubinius

- the ultimate level of “dogfooding”



Interpreter	LOC (non Ruby)	LOC (Ruby)
MRI (Ruby 1.8)	85 000, C	0
YARV (Ruby 1.9)	129 000, C	0
JRuby	115 000, Java	~ 1 000
IronRuby	48 000, C#	0
Rubinius	25 000, C	14 000

Good and bad things

A Programmer‘s Best Friend

Good Things

- Sweet Language
- Ruby on Rails
- Big community
  "ruby people are nice" - Martin Fowler
- Open Source

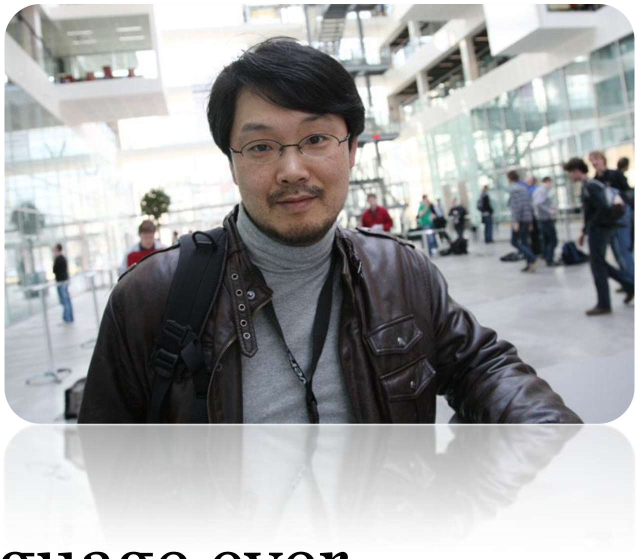
Bad Things

- Ruby 2 is vaporware
- „slow“
- IDE

Ruby 2

Future Prospects

Future Prospects



- gather wild & weird ideas
- try to make ruby the best language ever
- shed light to undefined corners of Ruby
- finally (if possible), document Ruby specification

Questions & Answers

Quellen

- confreaks.com – RubyConf 2006/2007/2008
- ruby-lang.org
- tryruby.hobbix.com
- „Ruby Grundlagen – PDF zum Buch Rapid Web Development mit Ruby on Rails“ – besimple.de
- OReilly - Ruby Cookbook
- poignantguide.net/ruby/