Day One

thomas@rubykurs.no
peter@rubykurs.no
ole.morten@rubykurs.no

Intro

Meet n' Greet

The Next Three Days

Today

- Basic syntax
- *Lunch*
- I/O and Environment
- OOP
- 3rd Party Code
- *End 17:00*

Setup

Ruby Intro

Background

- Appeared in 1995
- Perl, Smalltalk, Lisp
- Multiple impls
- 1.8.7 vs 1.9.2

Strengths

- String and file manipulation
- Dynamic language features
- Creating DSLs
- Large ecosystem, active community
- Focus on clean code and readability

Weaknesses

- Performance
- Easy to make a mess
- Small local community

Usage Outside Rails

- Scripting & automation
- Tools
- Project “glue”
- Polyglot architectures

Executing Code

```
ruby hello_world.rb
ruby -e "puts 'hello world'"
irb
```

Basic Syntax

Variables

```
simple_variable = "I am a string"
another_variable = 42
CANNOT_BE_CHANGED = "Constant string"
initialized_empty_var = nil
```

Objects

```ruby
puts "A String".methods # => ["upcase!", "count", "downcase" ...]

42.methods # => ["%", "+", odd?", ... ]

# Everything is an object.
# Objects are easy to inspect.
```

Functions

```
def strict_greeting(name)
  return ("Hello, "+name);
end

def loose_greeting name
  "Hello, "+name
end

puts(strict_greeting("Mr Smith"));  # => “Hello, Mr Smith”
puts loose_greeting "Mr Jones"      # => “Hello, Mr Jones”
```

Conditionals

```ruby
if true
  puts "always prints"
elsif false
  puts "never prints"
else
  puts "never prints"
end

puts "always prints" if true
puts "never prints" unless true
```

Loops

```ruby
while true
  puts "endless loop"
end

count = 0
until count == 5
  puts "climbing"
  count += 1
end
```

Exercise 01

exercises/01

Blocks

```ruby
3.times do
  puts "Nice for-loop!"
  puts "Multiline form"
end

3.times { puts "One-liner form" }

[1,2,3].each { |n| puts "Param to this block run is #{n}!" }

[1,2,3].map { |n| n * 2 }  # => [2, 4, 6]

closure_var = "dead people"
3.times { puts "I see #{closure_var}" }
```

Exercise 02

exercises/02

Array and Range

```ruby
an_array = [1, "two", 3] # can have any types

an_array[0]         # => 1
an_array.first      # => 1
an_array.last       # => “four”
an_array[1..2]      # => [“two”, 3]

(1..5).to_a         # => [1, 2, 3, 4, 5]
(1...5).to_a        # => [1, 2, 3, 4]

numbers_range = 0..9
numbers.class                    # Range
numbers_range.max                # => 9
numbers_range.min                # => 0
numbers_range.include? 5         # => true
```

Exercise 03

exercises/03

String

```ruby
holiday = "Christmas"
puts "Merry #{holiday}"    # => Merry Christmas

long_greeting = <<HERE_STRING
This is a long unquoted string
which includes line breaks, formatting, etc
We can also interpolate: #{greeting}
HERE_STRING

long_greeting.lines.count # => 3
```

Exercise 04

exercises/04

Symbol

```ruby
color_of_sky = :blue

# evalutes to itself - has no other value
# think of it as a unique label
# a "lonely enum value"
# handy for ids, like flags, hashmap keys etc
# can be transformed to and from String
```

Hash

```ruby
person = { :name => "Tony Soprano",
           :job => "Waste management" }

person[:name] # => "Tony Soprano"
person[:job]  # => "Waste management"

# Value can be anything
# Keys can be anything but must respond to .hash
```

Exercise 05

exercises/05

Exception Handling

```ruby
raise TypeError, "Param shouldn't be a String"
raise "A runtime exception, the easy way"

begin
  raise "kaboom"
rescue
  puts "rescuing from exception!"
end

begin
  raise TypeError.new("Inspectable exception")
rescue TypeError => te
  puts te.message
ensure
  puts "always runs in the end"
end
```

Exercise 06

exercises/06

Regular Expression

```ruby
literal_regex = /[A-za-z0-9]/
explicit_regex = Regexp.new("[A-za-z0-9]")
"Alfanumeric123" =~ literal_regex # => true

# Pragmatic but messy way
"This is -James Bond- calling" =~ /-(.*)-/ # => true
puts $1 # => James Bond
# $1 through $9 are thread-global variables
# Each match overwrites these variables

# Slightly cleaner and object-oriented
re = Regexp.new("-(.*)-")
match_data = re.match("This is -James Bond- calling")
puts match_data[0] # James Bond
# Can have several matches saved at the same time
```

Exercise 07

exercises/07

Lunch!

Environment and I/O

Shelling Out

```ruby
# Three ways of running shell commands
`ls`
%x[ ls ]
Kernel.system "ls"

current_dir = `pwd` # Store result
puts `pwd`          # Print result directly

file_path = "./.bash_profile"
file_body = `less #{file_path}` # String interpolation works fine

# Remember: this is Unix-specific code, breaks on Windows!
```

Exercise 08

exercises/08

File and Dir

```ruby
file = File.new("test1.txt", "w")
file.syswrite "wow" # do stuff to file
file.close

#better way, use a block:
File.open("test2.txt", "w") do |file_body|
  file_body.puts "Writing this to file"
end # File closed automagically

Dir["*"] # => ["bin", "Desktop", "Documents", ....]

# File and Dir are low level Ruby APIs
# Use FileUtils API for methods that map directly to cmd line operations
# cd(dir, options)
# pwd()
# mkdir(dir, options)
# chmod(mode, list, options)
# touch(list, options)
# ...etc

require 'fileutils'
FileUtils.mv('/tmp/your_file', '/opt/new/location/your_file')
```

Exercise 09

exercises/09

ARG, ENV, Paths

```
ENV # Hash of system environment variables
ENV # => {"SHELL"=>"/bin/bash", "HOME"=>"/Users/thomas", ...}

ARG # Array of the options send to called ruby program
ruby myscript.rb "one" 2 "three" # ARGV becomes ["one", 2, "three"]

__FILE__ # When executed in a script, bound to name of script file

$: # Paths where Ruby looks for libraries and files that we load

# Can add extra folders to load path during startup:
ruby -I lib:test hello.rb
# Can now run/require all files in ./lib and ./test
```

Exercise 10

exercises/10

OOP

Overview

- Classes and objects
- Modules
- Metaprogramming

Classes and Objects

Class Definition

```ruby
class Vehicle

  def initialize(name) # Constructor method
    @name = name    # Instance variable has @
suffix
  end

  def name  # Regular instance method
    @name
  end

end


v = Vehicle.new("Corolla")
puts v.name
```

Attributes

```ruby
class Vehicle
  def speed # Attribute reader
    @speed
  end
  def speed=(new_speed) # Attribute writer
    @speed = new_speed
  end
end

v = Vehicle.new("Mazda 6")
v.speed = 80
puts v.speed # => 80

# Shorter alternative form
class Vehicle
  attr :speed
end

v = Vehicle.new("Ford Fiesta")
v.speed = 90
puts v.speed # => 90
```

Subclassing, Self, Super

```ruby
class Car < Vehicle
  attr :manufacturer

  def initialize(manufacturer, name)
    super(name)
    @manufacturer = manufacturer
  end

  def full_title
    "#{self.manufacturer} #{self.name}"
  end

end

c = Car.new("Porsche", "911")
puts c.full_title #=> "Porsche 911"
```

Class Variables, Class Methods

```ruby
class Vehicle

  def Vehicle.set_max_speed(speed) # Class method
    @@max_speed = speed # Class variable has @@ prefix
  end

  def max_speed
    @@max_speed
  end

end

Vehicle.set_max_speed("299,792,458 metres per second")
c = Car.new("Porsche", "911")
puts c.max_speed # => 299,792,458 metres per second
```

Access Control

```ruby
class Vehicle

  def method1 # Default public visibility
    # ...
  end

  protected # Subsequent method will be visible to others of same class

  def method2
    # ...
  end

  private # Subsequent method only visible to same class objects

  def method3
    # ...
  end

  public # Subsequent methods will be public again

  # ...
end
```

Exercise 11

exercises/11

Modules

Namespacing

```ruby
module Ikea # Declares module
  OWNER = "Ingvar Kamprad" # Module-local constant

  class Factory # Module-local class def

  end
end

# Use :: separator to reach inside module from outside
puts Ikea::OWNER # => Ingvar Kamprad
factory1 = Ikea::Factory.new
factory2 = Factory.new # => NameError: uninitialized constant..

include Ikea # Make available to current context/object
factory3 = Factory.new
# Including the module mixes it into self
```

Mixins

```ruby
@materials = "55 planks"
build_porch # => NameError: undefined local variable or method
`build_porch' for main:Object

module Carpenter
  def build_porch
    puts "Built porch using #{@materials}!"
  end
end

include Carpenter # Mixing Carpenter into current object
build_porch # => "Built porch using 55 planks!"

# KEY POINT: module code interacts with code in object it's included in!

# Better example: The Enumerable module
# "The Enumerable mixin provides collection classes with several
# traversal and searching methods, and with the ability to sort. The
# class must provide a method each, which yields successive members of
# the collection [...]"
```

Exercise 12

exercises/12

Metaprogramming

Open Classes

```ruby
class Project # Create class
  def start
    puts "project starts"
  end
end

class Project # Reopen and add to the class
  def end
    puts "project ends"
  end
end

p = Project.new
p.start # => project starts
p.end   # => project ends

class String  # Can also monkeypatch core APIs
  def first_three
    self[0..2]
  end
end

puts "lorem ipsum".first_three # => "lor"
```

Dynamic Dispatch

```ruby
42 + 2              # => 44
42.+(2)             # => 44
42.send(:+, 2)      # => 44

# Can determine at runtime which method to call/
delegate to

# Can also determine if specific function call is safe
42.respond_to?(:+) # => true
42.respond_to?(:magical_operator) # => false
```

method_missing

```ruby
# We can respond to, and decide what happens, when non-existant
# methods are called
class Dummy
  def method_missing(method_name, *args, &block)
    puts "There's no method called #{method_name} here!"
  end
end

d = Dummy.new
d.mashed_potatoes # => "There's no method called mashed_potatoes here!"
```

Self-Modifying Classes

```ruby
# Pulling it all together:
# Classes are open, methods can be called dynamically
# and missing methods can be detected.
# So one thing we can do is create classes that modify themselves
# at runtime to respond to new scenarios...
class SmartFactory
  def method_missing(method_name, *args, &block)
    if method_name.to_s =~ /new_(.*)/
      expected_type = $1
      eval "#{expected_type}.new"
    else
      "SmartFactory doesn't respond to #{method_name}!"
    end
  end
end

sf = SmartFactory.new
sf.new_String   # => ""
sf.new_Array    # => []
sf.new_Hash     # => {}

# Real example: ActiveRecord responds to arbitrary find_by_X_and_Y()
```

Exercise 13

exercises/13

RubyGems

Using 3rd Party Code

```
# We package, distribute and use Ruby code as 'gems'.
# A gem can be a standalone tool, a code library or somewhere inbetween.

gem install Vagrant  # Installs the Vagrant tool
rake -h  # Used it as normal command line tool

gem install nokogiri # Installs Nokogiri xml library
# Can now use within your ruby code:
# require 'nokogiri'
# doc = Nokogiri::HTML(open('http://www.google.com/search?q=tenderlove'))
# doc.xpath('//h3/a[@class="l"]').each do |link|
```

Gem (Project) Internals

```
# Create skeleton for a new gem/project:
bundle gem my_gem

# Gem structure:
#.
#├── Gemfile
#├── Rakefile
#├── lib
#|   ├── my_gem
#|   |   └── version.rb
#|   └── my_gem.rb
#└── my_gem.gemspec
```

Wrapping Up

You Are Rubyists!

Learning More

Learn Ruby The Hard Way

Other Books

- Learn Python
- Learn Ruby
- Learn C
- Learn Regex
- Learn SQL
- CLI Crash Course

More Stuff

- About
- Blog

Learn Ruby The Hard Way

Do you really want to learn programming but have no skill? Are you a system administrator who wants to learn Puppet or Chef? Are you a designer who wants to build your own websites? Are you a Ruby on Rails programmer who's ashamed that you don't really know Ruby? Then you should read this book. It assumes absolutely **no prior programming knowledge** and will guide you carefully and slowly through the learning process.

Learn Ruby The Hard Way is a translation of the original "Learn Python The Hard Way" to teaching Ruby, with the translation done by Rob Sobers. "Learn Python The Hard Way" has taught hundreds of thousands worldwide how to code in Python, and this book uses the same proven method for Ruby. When you are done with this book you will have the skill to move on to other books about Ruby and be ready to understand them.

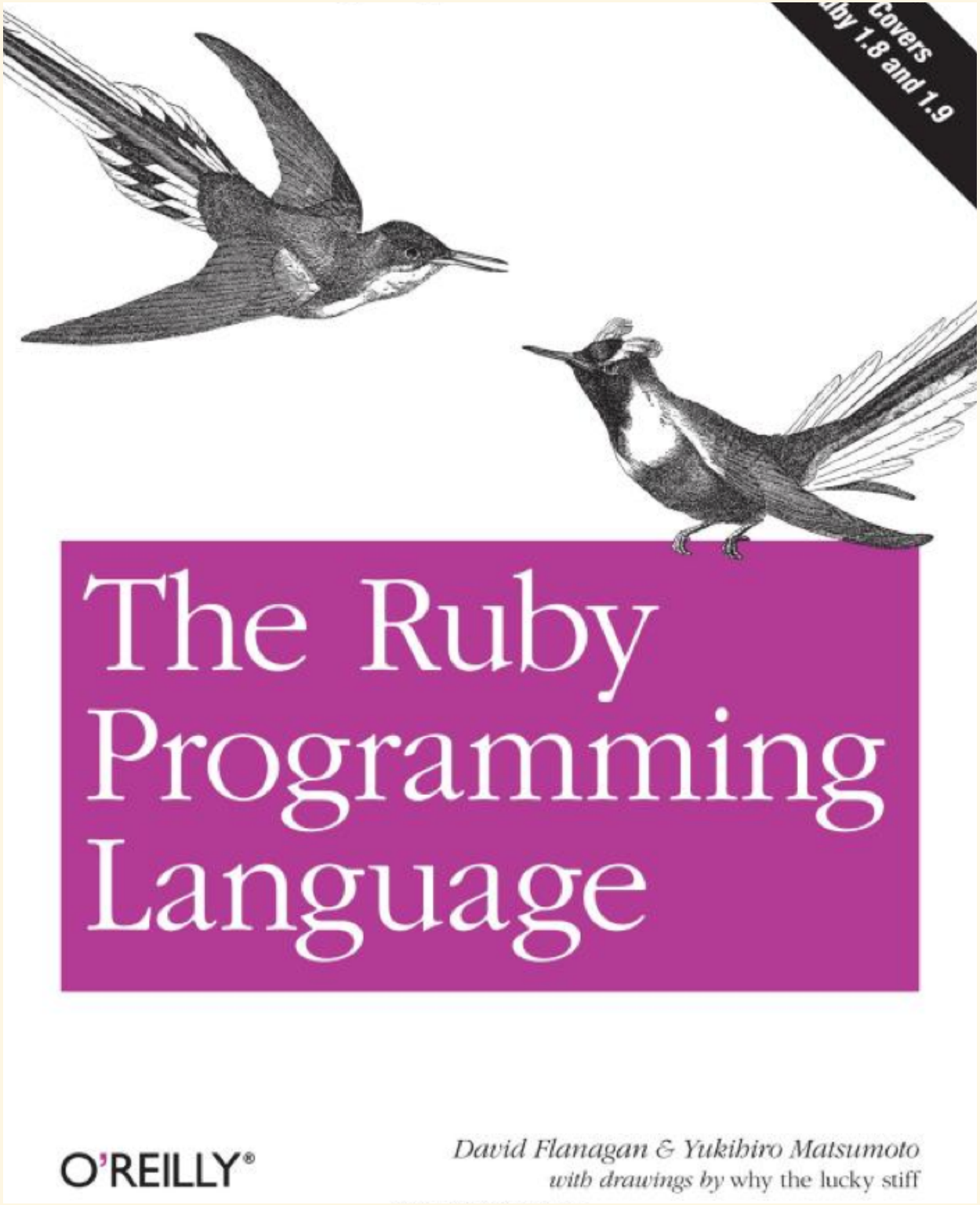
Get The Book for $2.99

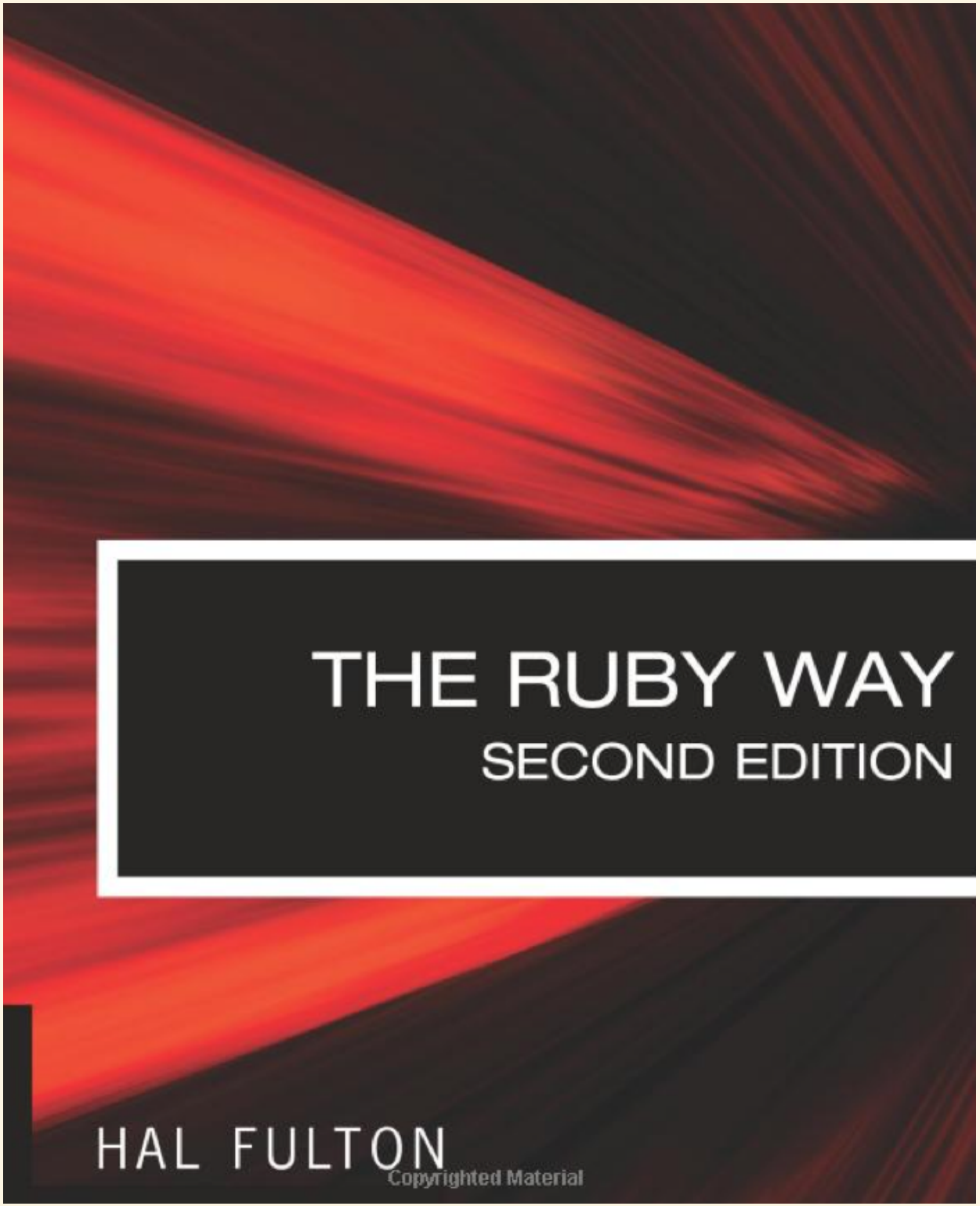
Buying the book keeps the HTML version free and funds the writing of other books.

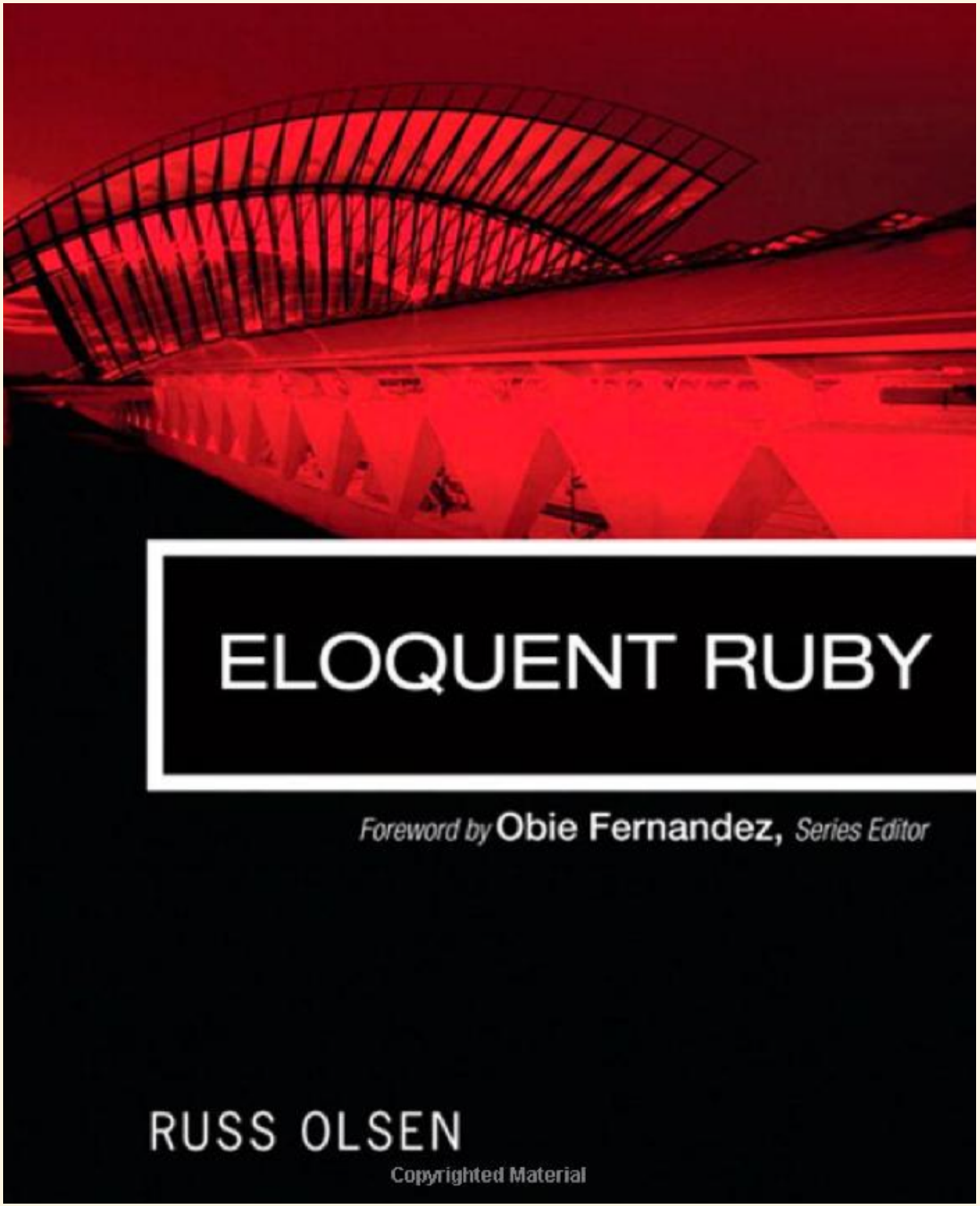
Buy PDF - $2.99
Buy ePub - $2.99
Buy Both - $5

Read The Free HTML

The book is free to read online because we want to create as many programmers as possible. We'd appreciate if you bought a copy to support writing more books, but feel free to read the read the **free** book online in HTML format.


Covers
by 1.8 and 1.9
The Ruby Programming Language
O'REILLY®
David Flanagan & Yukihiro Matsumoto
with drawings by why the lucky stiff


THE RUBY WAY
SECOND EDITION
HAL FULTON
Copyrighted Material


ELOQUENT RUBY
Foreword by Obie Fernandez, Series Editor
RUSS OLSEN
Copyrighted Material

HOME | PICKS | REQUEST A TOPIC

037 RR Versioning and Releases

by CHARLES MAX WOOD *on* JANUARY 12, 2012

00:00 / 00:00

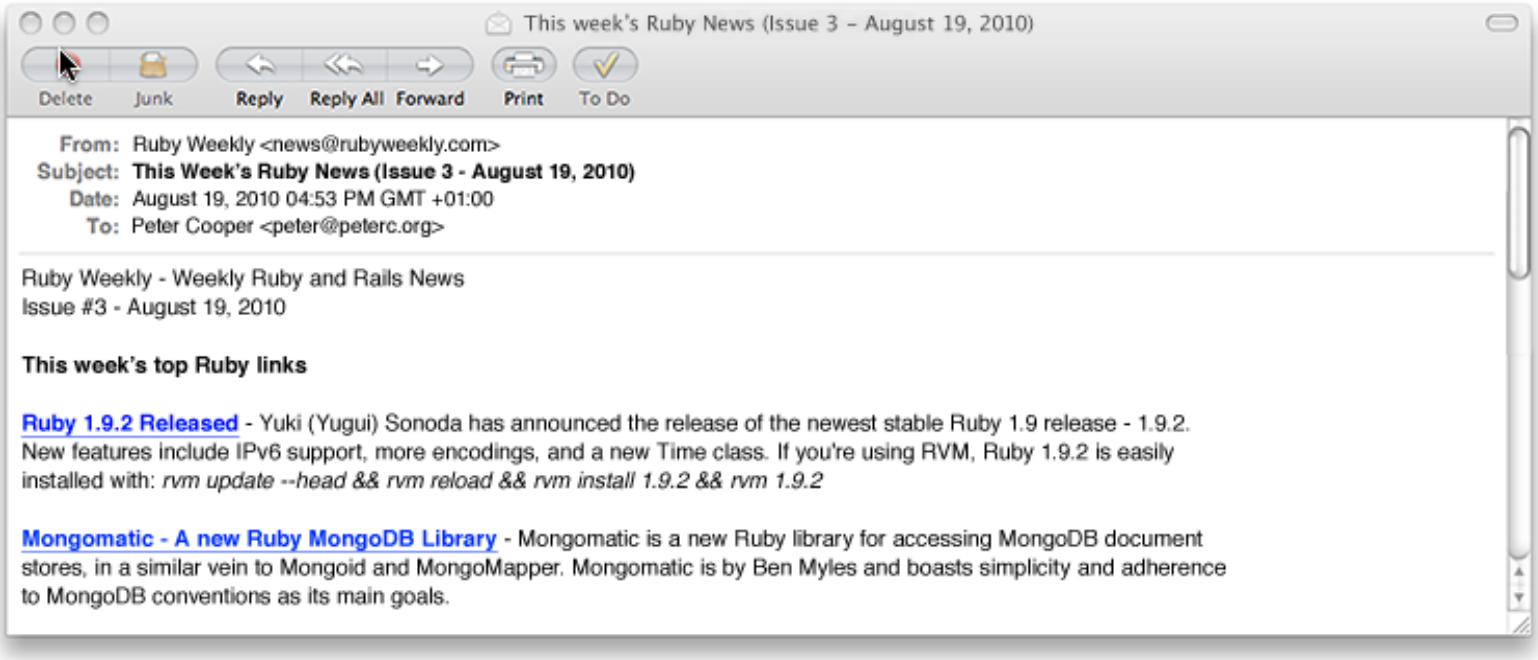
Podcast: Play in new window | Download (31.1MB)

Panel

- Avdi Grimm (twitter github blog book)
- David Brady (blog twitter github ADDcasts)
- James Edward Gray (blog twitter github)
- Josh Susser (twitter github blog)

Ruby Weekly

A free, once–weekly e-mail round-up of Ruby news and articles.



Enter your e-mail here: and click to subscribe :-)

One e-mail each Thursday. Easy to unsubscribe. No spam — your e-mail address is safe.
Curated by Peter Cooper (and not some big ole megacorp)

Want to see what you'll get? Check out issues 43, 42, and 41 or even our complete archive.

Feedback!

Questions?

thomas@rubykurs.no
peter@rubykurs.no
ole.morten@rubykurs.no