
THE
GOOD


THE
BAD


THE
UGLY


NO HATERS
ALLOWED

A BRIEF HISTORY OF PRETTY MUCH EVERYTHING

9,95

Array Integer Kernel

Class Struct Fixnum

Float Regexp Time

Object Bignum String

Nil Proc Dir IO

Module File Numeric

Data Range Hash

```
line = ''
print "ruby> "
while TRUE
  l = gets
  if l
    line = line + l
    continue if l =~ /,\s*$/
  end
  begin
    print eval(line).inspect, "\n"
  rescue
    $! = 'exception raised' if not $!
    print "ERR: ", $!, "\n"
  end
  break if not l
  line = ''
  print "ruby> "
end
```

1.0

1.2

14:4

onsdag den 8 juni 2011

1.6

18:0

etc

```
from_to_by = iter(from,to,by:int) yields(int)
    i : int := from
    if by> 0 then
        while i <= to do
            yield i
            i +:= by
        end
    else
        while i >= to do
            yield i
            i +:= by
        end
    end

for i in from_to_by(first,last,step) do
    ...
end
```

```
def from_to_by(from, to, by)
   i = from
   if by > 0 then
      while i <= to
         yield i
         i += by
      end
   else
      while i >= to
         yield i
         i += by
      end
   end
end

from_to_by(1, 10, 2) do lil
   ...
end
```

```
Class Bird;
  Begin
    Virtual: Procedure fly Is Procedure fly;
      OutText("I am flying");
  End;

Penguin Class Bird;
  Begin
    Procedure fly;
     OutText("Can't fly. =(");
end;
```

```ruby
class Bird
  def fly
    puts "I am flying"
  end
end

class Penguin < Bird
  def fly
    puts "Can't fly. =("
  end
end
```

```
(defflavor flux-capacitor ((energy-level 0.0)
                           plutonium-store)
  ()
  :gettable-instance-variables
  :settable-instance-variables)

(defflavor deLorean ()
  (flux-capacitor))

(defflavor TimeTrain (steam-level)
  (flux-capacitor))

(defmethod (flux-capacitor :load-energy) ()
  (setq energy-level 10.1))

(send train (make-instance 'TimeTrain))
(send train :load-energy)
```

```ruby
module FluxCapacitor
  def load_energy
    @energy_level = 10.0
  end
end

class DeLorean
  include FluxCapacitor
end
```

```ruby
class TimeTrain
  include FluxCapacitor
end

train = TimeTrain.new
train.load_energy
```

etc


THE
BAD

Flip flop

```
ARGF.each_line do |line|
  if (/^#ifdef/ =~ line)..(/^#endif/ =~ line)
    puts line
  end
end
```

śa

$\ $-l $-a $-d $-l

$/ $. $>

$-K $< $&

$` $” $+

$-p $’

$1 $! $,

$~ $:

$? $;

$* $= $0

$$ $_ $@

warn

binding

```ruby
def foo
  x = 1
  binding
end
```

```ruby
def b
  binding
end

def foo2
  x = 2
  b
end
```

```
ʎ jruby -e 'alias b binding'
-e:1 warning: `binding' should not be aliased
```

```
ʎ jruby -e 'alias evil eval'
-e:1 warning: `eval' should not be aliased
```

defined?

```ruby
p defined?(42)  #=> expression
p defined?(puts) #=> method
p defined?($')   #=> global-variable
@x = true
p defined?(@x)   #=> instance-variable
p defined?(Object.new.inspect.lenght) #=> nil
p defined?(a_method_we_dont_know) #=> nil
p defined?(@some_ivar) #=> nil
p defined?(a.chain.of.method.calls) #=> nil
```


THE
UGLY

Lexical scoping

```ruby
x = 1
class Foo
  puts x
end

module Bar
  puts x
end

def something
  puts x
end
```

```
def common_mistake
  var = 42
  def foo
    var + 13
  end
  puts foo
end
```

Return values

```
m = def foo
      42
    end
```

Speed

Blocks

```
foo proc { |x| x + 1} do |f|
  puts "hello"
end
```

Exceptions

Keyword

arguments

```ruby
def something(options = {})
  options.assert_valid_keys([:one, :two, :three])

  puts options[:one]
  puts options[:three] || "Default value"
end

something :hello => "foo", :three => false
```

Module

inclusion

```ruby
module A end
module C def bar; 25 end end

class B
  include A
end

module A
  include C
  def foo; 42 end
end

B.new.foo
B.new.bar
```

```ruby
module FlatMap
  def flat_map
    result = []
    each { |v| result.concat yield v }
    result
  end
end

module Enumerable
  include FlatMap
end

[1,2,3].flat_map { |x| [x] }
```


THE
GOOD

Monkey

patching

Categories

```
def foo
  [1,2,3].each do |x|
    puts x + 1
  end
end

module Bar
  category_on Array do
    def each
      super do |x|
        yield x + 1
      end
    end
  end
end
```

```
[1,2,3].each do |x|
  puts "Hello: #{x}"
end

using Bar do
  [1,2,3].each do |x|
    puts "Hello: #{x}"
  end
end

using Bar do
  foo
end
```

Refinements

```ruby
def foo
  [1,2,3].each do |x|
    puts x + 1
  end
end

module Bar
  refine Array do
    def each
      super do |x|
        yield x + 1
      end
    end
  end
end
```

```ruby
[1,2,3].each do |x|
  puts "Hello: #{x}"
end

using Bar
[1,2,3].each do |x|
  puts "Hello: #{x}"
end


foo
```

Namespaces

```ruby
def foo
  [1,2,3].each do |x|
    puts x + 1
  end
end

module Bar
  namespace Bar
  class ::Array
    def each
      default:each do |x|
        yield x + 1
      end
    end
  end
end
```

```ruby
[1,2,3].each do |x|
  puts "Hello: #{x}"
end

[1,2,3].Bar:each do |x|
  puts "Hello: #{x}"
end

using Bar
[1,2,3].each do |x|
  puts "Hello: #{x}"
end

foo
```

Meta-

programming

Syntax

%~quux~

%(hello (bar))

%s<foo>

Regexps

Taint

```
$SAFE = 1
x =  "fire_rockets"
x.taint
eval(x)
```


Evolution

Gradual typing

Concurrency

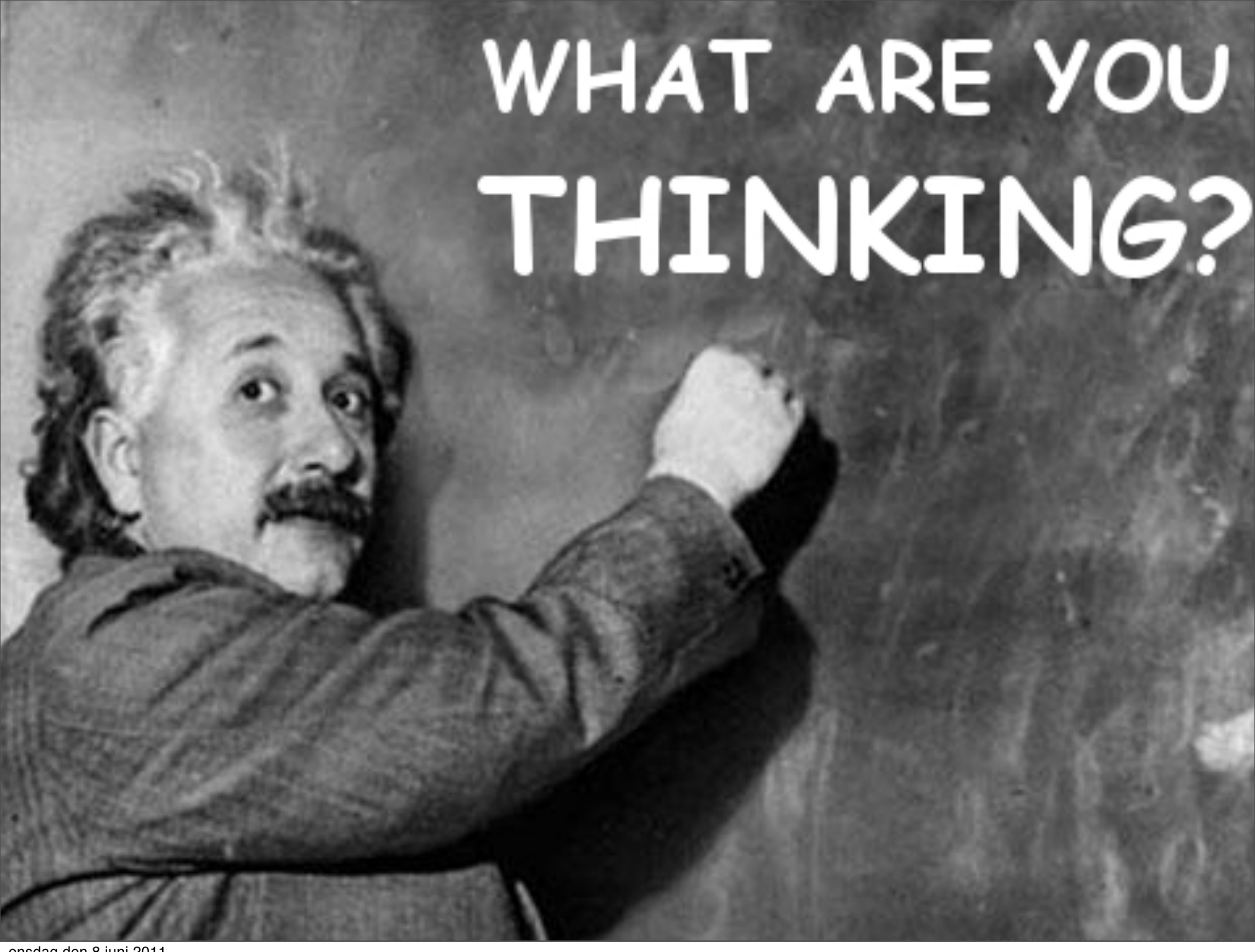
Memory model

Agents

Thread.new

Ops in VM

onsdag den 8 juni 2011


WHAT ARE YOU
THINKING?

Q?

Ola Bini

ThoughtWorks®

http://olabini.com