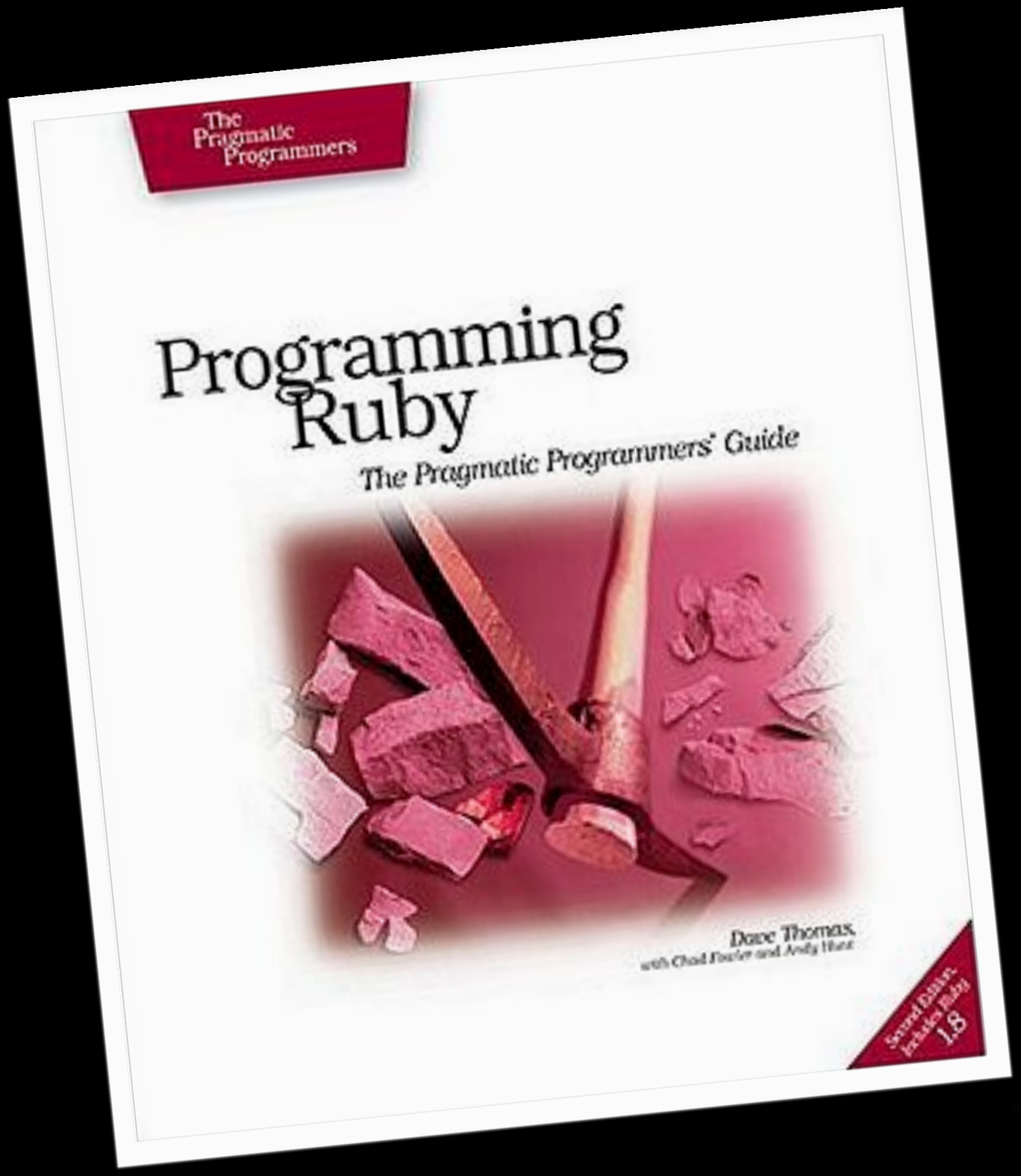
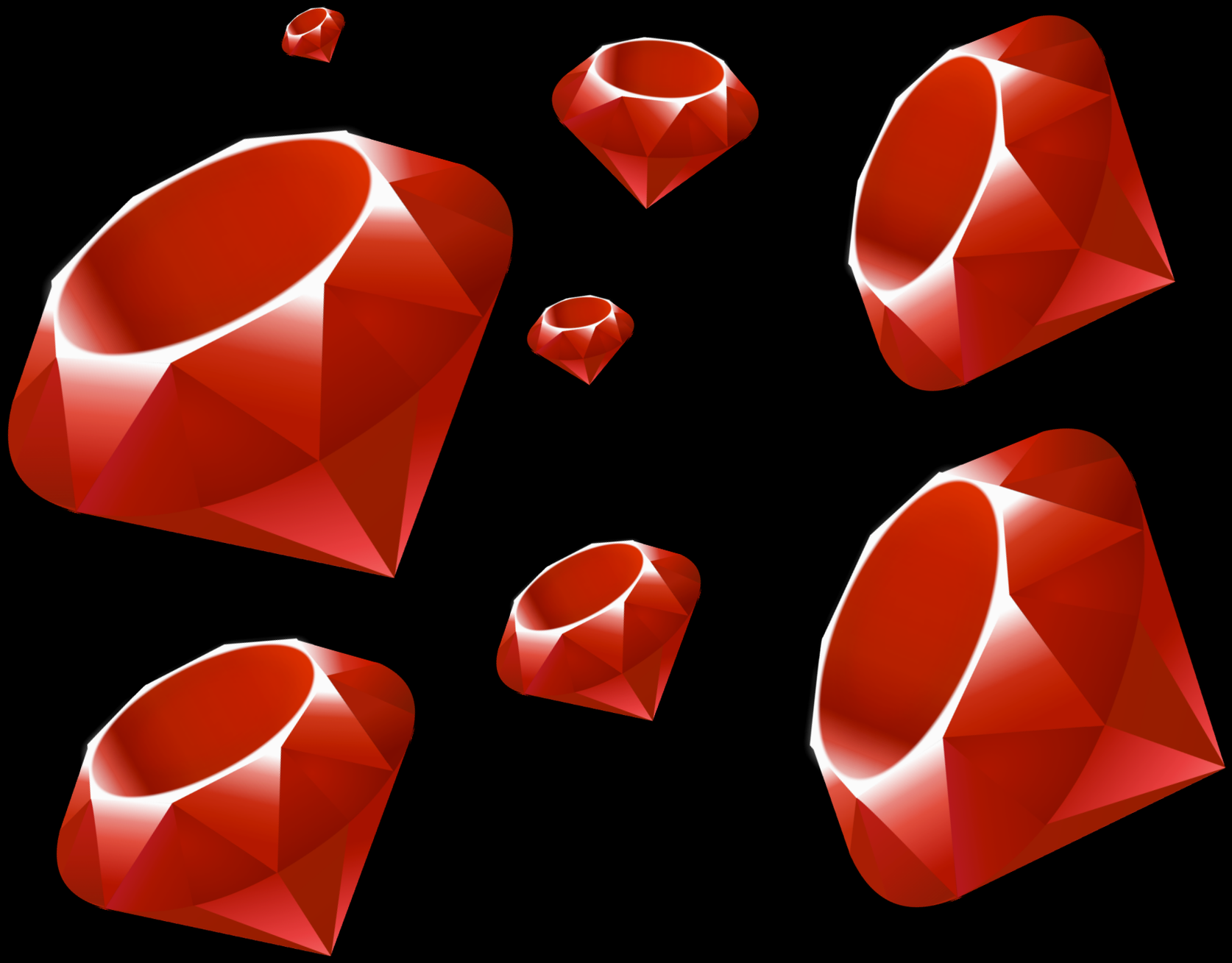

The Pragmatic Programmers
Programming Ruby
The Pragmatic Programmers' Guide
Dave Thomas
with Chad Fowler and Andy Hunt
Second Edition, Includes Ruby 1.8

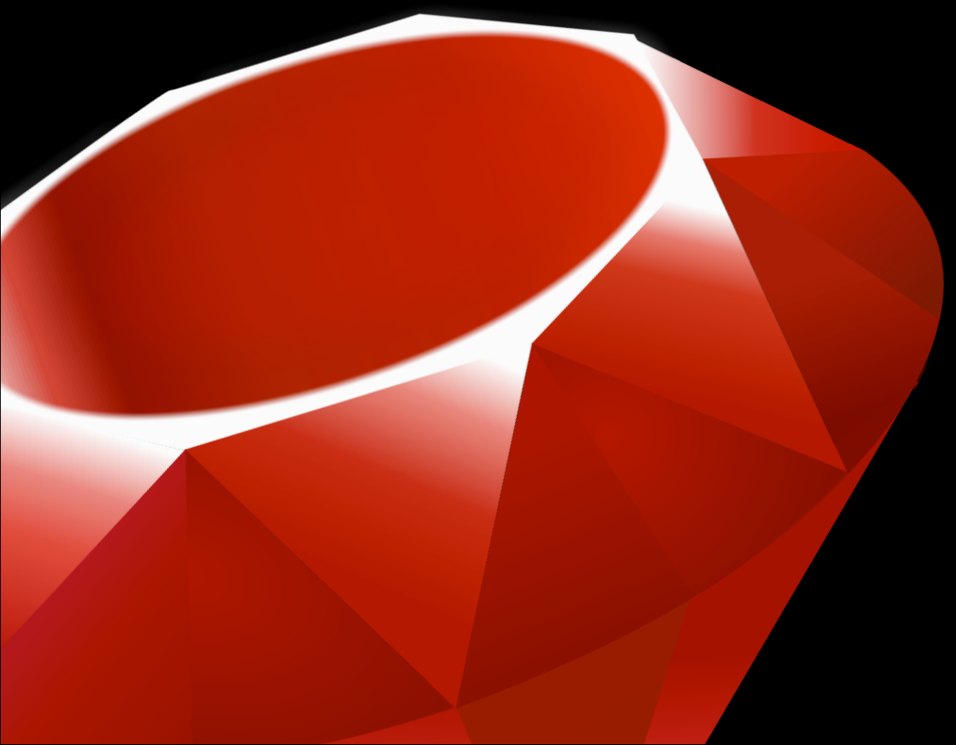
2000



(1.8.7)

MRI

Matz Reference Implementation
Matz Ruby Interpreter



(2.1)
YARV
Yet Another Ruby VM

JRuby

Ruby on Java

IronRuby

Ruby on .NET

Rubinius

Use Ruby™

Rubinius

Ruby on Ruby

Ejecutando

Ruby Scripts

hello_ruby.rb

hello_ruby.rb

```
print "Hagamos algo tipico. ¿Cómo te llamas? "
name = gets.chomp
puts "#{name}? Es un nombre interesante"
```

ruby hello_ruby.rb

```
[13:44][~/curso_rails(master)]$ ruby hello_ruby.rb
Hagamos algo tipico. ¿Cómo te llamas? Alvaro
Alvaro? Es un nombre interesante
```

Identificadores en Ruby

variables

minúsculas

locales, de instancia, de clase y globales

```
name = "Alvaro"

order_number = 23444

sales_result = 9_123
```

constantes

mayúsculas

un valor fijo

```
IGV = 19
WEATHER_STATES = ["cold", "hot"]

Country = "Peru"

class ProductOwner
end
```

keywords

propias del lenguaje

```
def class if
else end ...
```

métodos

mismas reglas que variables locales

```
def start
end

def add_this
end

def is_admin?
end

def update!
end
```

Cadenas en Ruby

Cadenas simples

```
puts 'Esta es una cadena simple'

puts 'Se pueden ' + 'sumar'

puts 'Y' + ' repetir y' * 5

puts 'Esta esa una comilla: \''
```

Cadenas dobles

```
puts "Esta es una cadena doble"
puts "Se ve así: \"   "



name = "Alvaro"
puts "Mira #{name}, es de noche"
```

Conversiones

```
name = "Alvaro"
puts "Mira #{name}, es de noche"

age = 24
puts "Tengo " + age.to_s + " años"
```

```ruby
#Primera versión
print "¿Qué edad tienes?"
user_age = gets.chomp

puts " ¿" + user_age + "? ¡Pareces de " + (user_age.to_i - 5).to_s + "!"
```

```ruby
#Segunda versión
print "¿Qué edad tienes?"
user_age = gets.chomp

sweet_age = user_age.to_i - 5
puts " ¿#{user_age}? ¡Pareces de #{sweet_age}!"
```

Números en Ruby

Number

Fixnum

BigInt

Number

Fixnum

BigInt

`+` `-` `*` `/` `%` `**`

`zero?`

Float

```
+ - * / %    **

zero?
```

Estructuras de Control en Ruby

condicionales

```
if condicion then resultado end
```

```
if condicion
     resultado
end

resultado  if condicion
```

condicionales

```
if condicion then resultado  else negación end
```

```
if condicion
      resultado
else
   negación
end
```

condicionales

```
if condicion
      resultado
elsif condición2
   resultado_alternativo
end
```

condicionales

```
if not condicion
     resultado
end

if !condicion
     resultado
end
```

condicionales

```
unless condicion then resultado
end
```

```
unless condicion
      resultado
end

resultado  unless condicion
```

condicionales

```
unless condicion
  resultado
elsif condición2
  resultado_alternativo
end
```

condicionales

```
variable = resultado  if condicion
```

```
puts "Dos + Dos = Cuatro. ¿Esto es Verdadero o Falso? (V/F)"
respuesta = gets.chomp
resultado = if respuesta == "V"
              "¡Correcto! Pero eso ya lo sabías"
            else
              "Falso. ¿Estás seguro que estás en el curso correcto?"
            end

puts resultado
```

condicionales

```
case sentencia
     when evaluación then
resultado
     .....
end
```

condicionales

```
variable = case sentencia
                    when
  evaluación then resultado
                    .....
                  end
```

```ruby
puts "Dos + Dos = Cuatro. ¿Esto es Verdadero o Falso? (V/F)"
respuesta = gets.chomp
resultado = case respuesta
              when "V" then "¡Correcto! Pero eso ya lo sabías"
              when "F" : "Falso. ¿Estás seguro que estás en el curso correcto?"
              else "¿Uh? No te entendí"
            end

puts resultado
```

iteradores

```
while expresión
      acciones
end

acciones while expresión
```

iteradores

```
until expresión
      acciones
end
```

```
acciones until expresión
```

métodos

como siempre.. en Ruby

envío de mensajes

```
def mi_metodo
  # código
end

mi_metodo();
```

envío de mensajes

```
def mi_metodo
  # código
end

mi_metodo
```

envío de mensajes

```
def mi_metodo(argumento)
  # código
end

mi_metodo(12);
```

envío de mensajes

```
def mi_metodo(argumento)
  # código
end

mi_metodo 12
```

envío de mensajes

```
class Order < ActiveRecord::Base

  has_many :order_details

end
```

envío de mensajes

```
class Order < ActiveRecord::Base

  has_many :order_details

end
```

has_many(:order_details)

envío de mensajes

```
person.do_work
```

envío de mensajes

```
person.do_work
```

envío de mensajes

```
person.do_work
```

método send

```
send(:metodo,args)
```

método send

```
puts "Hola".send(:upcase)
```