

Referencia Rápida de Ruby

Objetos, bloques, iteradores, regex, E/S de archivos

Fundamentos

Hola Mundo

```
puts "Hello, World!"
print "no newline"
p [1, 2, 3] # inspect output: [1, 2, 3]
```

Ejecutar Ruby

```
ruby script.rb # run a file
ruby -e 'puts "hi"' # run inline
irb # interactive REPL
```

Variables

name	Variable local
@name	Variable de instancia
@@count	Variable de clase
\$debug	Variable global
MAX_SIZE	Constante (mayúsculas por convención)

Tipos

42.class	# Integer
3.14.class	# Float
"hello".class	# String
true.class	# TrueClass
nil.class	# NilClass
:symbol.class	# Symbol

Cadenas

Básicos de Cadenas

```
name = "World"
puts "Hello, #{name}!" # interpolation (double quotes)
puts 'No #{interpolation}' # literal (single quotes)
multi = <<-HEREDOC
  indented heredoc
HEREDOC
```

Métodos de Cadenas

.length / .size	Conteo de caracteres
.upcase / .downcase	Conversión de mayúsculas/minúsculas
.strip	Eliminar espacios al inicio y final
.split(' ', '')	Dividir en array
.gsub(/pat/, 'rep')	Sustitución global
.include?('sub')	Verificar si contiene subcadena
.start_with?('pre')	Verificar prefijo
.chars / .bytes	Array de caracteres / bytes
.to_i / .to_f	Convertir a entero / float
.freeze	Hacer la cadena inmutable

Arrays y Hashes

Arrays

```
arr = [1, "two", :three]
arr << 4 # push (append)
arr[0] # 1
arr[-1] # 4 (last element)
arr[1..2] # ["two", :three] (slice)
```

Métodos de Array

.push / .pop	Agregar/eliminar del final
.shift / .unshift	Eliminar/agregar al inicio
.flatten	Aplanar arrays anidados
.compact	Eliminar valores nil
.uniq	Eliminar duplicados
.sort / .reverse	Ordenar / invertir orden
.map { x x * 2 }	Transformar cada elemento
.select { x x > 0 }	Filtrar elementos
.reduce(0) { sum, x sum + x }	Acumular en un solo valor

Hashes

```
user = { name: "Alice", age: 30 } # symbol keys
old = { "key" => "value" } # string keys
user[:name] # "Alice"
user[:email] = "a@b.com" # add pair
user.fetch(:name, "default") # with default
```

Métodos de Hash

.keys / .values	Array de claves / valores
.each { k, v }	Iterar pares clave-valor
.merge(other)	Combinar dos hashes
.key?(k) / .value?(v)	Verificar existencia
.select { k, v }	Filtrar pares
.transform_values { v }	Transformar todos los valores

Flujo de Control

Condicionales

```
if score >= 90 then "A"
elsif score >= 80 then "B"
else "C"
end
puts "adult" if age >= 18 # inline if
puts "minor" unless age >= 18 # inline unless
```

Case / When

```
case status
when :ok then puts "success"
when :error then puts "failed"
when 400..499 then puts "client error"
else puts "unknown"
end
```

Bucles

```
5.times { |i| puts i }
(1..10).each { |n| puts n }
while condition do end
until condition do end
loop { break if done }
```

Ternario y Lógico

```
status = age >= 18 ? "adult" : "minor"
name = input || "default" # or-assign
name ||= "fallback" # same effect
```

Métodos

Definir Métodos

```
def greet(name, greeting = "Hello")
  "#{greeting}, #{name}!"
end
greet("Alice") # "Hello, Alice!"
greet("Bob", "Hi") # "Hi, Bob!"
```

Valores de Retorno

```
def add(a, b)
  a + b # last expression is implicit return
end
def divide(a, b)
  return nil if b == 0
  a.to_f / b
end
```

Argumentos Keyword y Splat

```
def connect(host:, port: 80, **opts)
  puts "#{host}:#{port} #{opts}"
end
def log(*messages)
  messages.each { |m| puts m }
end
```

Convenciones de Métodos

method?	Retorna booleano (predicado)
method!	Muta el receptor (método bang)
self.method	Definición de método de clase

Clases

Definición de Clase

```
class User
  attr_accessor :name, :email
  def initialize(name, email)
    @name = name
    @email = email
  end
end
```

Herencia

```
class Admin < User
  def initialize(name, email, level)
    super(name, email)
    @level = level
  end
end
```

Control de Acceso

public	Predeterminado; accesible desde cualquier lugar
private	Solo accesible dentro de la clase
protected	Accesible dentro de clase y subclases
attr_reader	Generar método getter
attr_writer	Generar método setter
attr_accessor	Generar getter y setter

Módulos

Mixins

```
module Greetable
  def greet
    "Hello, I'm #{name}"
  end
end
class User; include Greetable; end
```

Espacios de Nombres

```
module Payment
  class Processor
    def charge(amount) end
  end
end
p = Payment::Processor.new
```

Referencia Rápida de Ruby

Include vs Extend

include ModName	Agregar como métodos de instancia
extend ModName	Agregar como métodos de clase
prepend ModName	Insertar antes de la clase en la búsqueda de métodos

Bloques e Iteradores

Sintaxis de Bloques

```
[1, 2, 3].each { |n| puts n }      # single-line block
[1, 2, 3].each do |n|
  puts n                          # multi-line block
end
```

Yield

```
def with_logging
  puts "start"
  result = yield
  puts "end"
  result
end
with_logging { expensive_operation }
```

Procs y Lambdas

```
square = Proc.new { |x| x ** 2 }
square.call(5)          # 25
double = ->(x) { x * 2 } # lambda
double.call(3)         # 6
[1, 2, 3].map(&square)  # [1, 4, 9]
```

Iteradores Comunes

.each	Iterar sobre elementos
.map / .collect	Transformar cada elemento
.select / .filter	Mantener elementos coincidentes
.reject	Eliminar elementos coincidentes
.reduce / .inject	Acumular en un solo valor
.each_with_index	Iterar con índice
.flat_map	Mapear y aplanar un nivel
.any? / .all? / .none?	Verificaciones booleanas sobre la colección

Regex

Coincidencia

```
"hello 42" =~ /\d+/      # 6 (match position)
"hello" =~ /\d+/        # nil (no match)
"hello".match?(/ell/)   # true
md = "age: 30".match(/(\d+)/)
md[1]                   # "30"
```

Patrones Comunes

/^start/	Anclado al inicio
/end\$/	Anclado al final
/\d+/	Uno o más dígitos
/\w+/	Caracteres de palabra
/\s+/	Espacio en blanco
/[a-z]+/i	Sin distinción de mayúsculas
/(group)/	Grupo de captura

Sustitución

```
"hello world".sub(/world/, "Ruby") # first match
"aabba".gsub(/a/, "x")            # all matches: "xxbbx"
"foo bar".gsub(/(\w+)/) { $1.upcase } # "FOO BAR"
```

E/S de Archivos

Leer y Escribir

```
content = File.read("data.txt")
lines = File.readlines("data.txt", chomp: true)
File.write("out.txt", "hello\n")
File.open("log.txt", "a") { |f| f.puts "entry" }
```

Operaciones con Archivos

File.exist?(path)	Verificar si el archivo existe
File.directory?(path)	Verificar si la ruta es un directorio
File.basename(path)	Nombre de archivo sin directorio
File.extname(path)	Extensión del archivo
File.size(path)	Tamaño del archivo en bytes
File.delete(path)	Eliminar un archivo
Dir.glob('*.*rb')	Buscar archivos que coincidan con el patrón
FileUtils.mkdir_p(path)	Crear directorio recursivamente

CSV y JSON

```
require "json"
data = JSON.parse(File.read("data.json"))
File.write("out.json", JSON.pretty_generate(data))
require "csv"
CSV.foreach("data.csv", headers: true) { |row| puts row["name"] }
```