Code Conventions II
Разбор Nokogiri

План

- Code Conventions II
- Nokogiri

Классы и модули

Структура класса

```ruby
class Person
  extend SomeModule
  include AnotherModule

  CustomError = Class.new(StandardError)

  SOME_CONSTANT = 20

  attr_reader :name

  validates :name

  ...
```

Extend vs Include

```ruby
module A
  def hello
    puts 'hello'
  end
end
```

```ruby
class B
  include A
end

Bar.new.foo # hello
Bar.foo # NoMethodError

class C
  extend A
end

Baz.new.foo # NoMethodError
Baz.foo # hello
```

Структура класса

```
def self.some_method
  end

  def initialize
  end

  def some_method
  end

...
```

```
protected

  def some_protected_method
  end

  private

  def some_private_method
  end
end
```

Mixins

```ruby
# bad
class Person
  include Foo, Bar
end

# good
class Person
  # multiple mixins go in separate statements
  include Foo
  include Bar
end
```

Классы в одну строку

```ruby
# bad
class FooError < StandardError
end

# good
class FooError < StandardError; end

# okish
FooError = Class.new(StandardError)
```

Файлы с классами

```ruby
# foo.rb
class Foo
  # 30 methods inside
end

# foo/bar.rb
class Foo
  class Bar
  end
end

# foo/car.rb
class Foo
  class Car
  end
end
```

Namespaces

```ruby
module Utilities
  class Queue
  end
end
```

```ruby
# bad
class Utilities::Store
  Module.nesting # => [Utilities::Store]

  def initialize
    @queue = Queue.new
  end
end
```

```ruby
# good
module Utilities
  class WaitingList
    Module.nesting # =>
[Utilities::WaitingList,
Utilities]

    def initialize
      @queue = Queue.new
    end
  end
end
```

Модуль или класс?

SOLID

- Single responsibility principle

Класс имеет единственное предназначение

- Open-closed principle

Открыт для расширения, закрыт для модификации

- Liskov substitution principle

Объект можно заменить экземпляром наследника

- Interface segregation principle

Много интерфейсов лучше, чем один общий

- Dependency inversion principle

Зависимость от абстракции, а не от реализации

Что не забыть написать?

- `.to_s`
- `attr_reader, attr_writer, attr_accessor`
- `.name, .name=`

Структуры

```ruby
# good
class Person
  attr_accessor :first_name, :last_name

  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name = last_name
  end
end

# better
Person = Struct.new(:first_name, :last_name) do
end
```

Структуры

```ruby
# bad
class Person < Struct.new(:first_name, :last_name)
end

# good
Person = Struct.new(:first_name, :last_name)
```

Утиная типизация

```ruby
# bad
class Animal
  # abstract method
  def speak
  end
end

# extend superclass
class Duck < Animal
  def speak
    puts 'Quack! Quack'
  end
end

# extend superclass
class Dog < Animal
  def speak
    puts 'Bau! Bau!'
  end
end
```

Утиная типизация

```ruby
# good
class Duck
  def speak
    puts 'Quack! Quack'
  end
end

class Dog
  def speak
    puts 'Bau! Bau!'
  end
end
```

Без статических переменных

```ruby
class Parent
  @@class_var = 'parent'

  def self.print_class_var
    puts @@class_var
  end
end

class Child < Parent
  @@class_var = 'child'
end
```

```ruby
Parent.print_class_var
```

def self.method

```ruby
class TestClass
  # bad
  def TestClass.some_method
    # body omitted
  end

  # good
  def self.some_other_method
    # body omitted
  end
end
```

Alias – сложновато

```ruby
class Westerner
  def first_name
    @names.first
  end

  alias given_name first_name
end
```

```ruby
class Fugitive < Westerner
  def first_name
    'Nobody'
  end

  alias given_name first_name
end
```

Можно делать фабрики

```ruby
class Person
  def self.create(options_hash) #ha-ha
    # body omitted
  end
end
```

Комментарии

- Не надо писать комментарии

Если не получается объяснить *почему*

```
x = BuggyClass.something.dup

def compute_dependency_graph
...30 lines of recursive graph merging...
end

# good

# BuggyClass returns an internal object, so we have to dup it to modify it.
x = BuggyClass.something.dup

# This is algorithm 6.4(a) from Worf & Yar's _Amazing Graph Algorithms_ (2243).
def compute_dependency_graph
...30 lines of recursive graph merging...
end
```

Все-таки как комментировать?

- На английском
- Грамотно
- # с пробелом
- На отдельной строке

Ключевые слова

- TODO
- FIXME
- OPTIMIZE
- HACK
- REVIEW

- Остальные можно объявить в README

Волшебные комментарии

```
# encoding: big5
# Characters before encoding: ascii or after it
# encoding: utf-16le - will lead to error

# frozen_string_literal: true
p "".frozen? # => true

# warn_indent: true
def method_name
  end
```

Коллекции

Инициализация

```ruby
# bad
arr = Array.new
hash = Hash.new

# good
arr = []
arr = Array.new(10)
hash = {}
hash = Hash.new(0)
```

%w

```ruby
# bad
STATES = ['draft', 'open', 'closed']

# good
STATES = %w[draft open closed]
```

```
%i
```

```
# bad
STATES = [:draft, :open, :closed]

# good
STATES = %i[draft open closed]
```

.first, .last

- .first
- .second
- .third
- .forth
- .fifth
- .forty_two
- .last

Set вместо Array

```ruby
require 'set'

# Union
Set[1,2,3] | Set[3,4,5]
#=> #{1,2,3,4,5}

# Intersection
Set[1,2,3] & Set[2,3,4,5]
#=> #{2,3}

# Subtraction
Set[1,2,3,4] - Set[3,4,5]
#=> #{1,2}


Set[1,2,3] == Set[3,1,2]
```

Символы как ключи

```ruby
# bad
hash = { 'one' => 1, 'two' => 2, 'three' => 3 }

# good
hash = { one: 1, two: 2, three: 3 }


# bad
hash = { :one => 1, :two => 2, :three => 3 }

# good
hash = { one: 1, two: 2, three: 3 }
```

Hash#each

```ruby
# bad
hash.keys.each { |k| p k }
hash.values.each { |v| p v }
hash.each { |k, _v| p k }
hash.each { |_k, v| p v }

# good
hash.each_key { |k| p k }
hash.each_value { |v| p v }
```

Hash#fetch

```ruby
heroes = { batman: 'Bruce Wayne', superman: 'Clark Kent' }
# bad - if we make a mistake we might not spot it right away
heroes[:batman] # => 'Bruce Wayne'
heroes[:supermann] # => nil

# good - fetch raises a KeyError making the problem obvious
heroes.fetch(:supermann)


batman = { name: 'Bruce Wayne', is_evil: false }

batman[:is_evil] || true # => true

# good - fetch works correctly with falsy values
batman.fetch(:is_evil, true)
```

Hash + blocks

```ruby
batman = { name: 'Bruce Wayne' }

# bad
batman.fetch(:powers, obtain_batman_powers)

# good
batman.fetch(:powers) { obtain_batman_powers }
```

Hash#values_at

```ruby
# bad
email = data['email']
username = data['nickname']

# good
email, username = data.values_at('email', 'nickname')
```

Общие правила

- Не меняйте хэш во время прохода
- Не используйте мутабельные ключи (rehash!)
- С 1.9 хэши отсортированы

Немного эффективности

```ruby
# bad
array.reverse.each { ... }

# good
array.reverse_each { ... }
```

```ruby
# bad
some_hash.count

# good
some_hash.size
```

Underscore

```ruby
# bad
num = 1000000

# good
num = 1_000_000
```

Префиксы

```
# good - easier to separate digits from the prefix
num = 0o1234
num = 0x12AB
num = 0b10101
num = 1234
```

Случайные числа

```ruby
# bad
rand(6) + 1

# good
rand(1..6)
```

Строки

Интерполяция

```
# bad
email_with_name = user.name + ' <' + user.email + '>'

# good
email_with_name = "#{user.name} <#{user.email}>"

# good
email_with_name = format('%s <%s>', user.name, user.email)
```

+, <<, #{}

- << - быстрее всех для строк (0.12 vs 90)
- #{} – подходит для объектов

```ruby
start = Time.now
1000000.times { a << 'a' }

finish = Time.now

puts diff = finish - start
```

Не бросайте {}

```ruby
def to_s
  "#@first_name #@last_name"
end

# good
def to_s
  "#{@first_name} #{@last_name}"
end
```

Nokogiri

- Gem install nokogiri

- Парсинг XML
- Парсинг HTML
- Поддержка селекторов CSS
- Сборка XML и HTML

Что видит пользователь?

```ruby
require 'nokogiri'
require 'open-uri'

# Fetch and parse HTML document
doc = Nokogiri::HTML(open('https://nokogiri.org/tutorials/installing_nokogiri.html'))

puts "### Search for nodes by css"
doc.css('nav ul.menu li a', 'article h2').each do |link|
  puts link.content
end

puts "### Search for nodes by xpath"
doc.xpath('//nav//ul//li/a', '//article//h2').each do |link|
  puts link.content
end

puts "### Or mix and match."
doc.search('nav ul.menu li a', '//article//h2').each do |link|
  puts link.content
end
```

Базовые классы

Nokogiri::HTML::Document
Nokogiri::XML::Document

```
doc1 = Nokogiri(string_or_io)
doc2 = Nokogiri::HTML(string_or_io) # [, url, encoding, options, &block]
doc3 = Nokogiri::XML(string_or_io) # [, url, encoding, options, &block]
```

Добавляем элементы

```
doc.create_element "div" # <div></div>
doc.create_element "div", :class => "container"
doc.create_element "div", "contents"
doc.create_element "div", "contents", :class => "container"


doc.create_element "div"
                   { |node| node['class'] = "container" }
```

Как устроено внутри?

```
def create_element name, *args, &block
  elm = Nokogiri::XML::Element.new(name, self, &block)
…
```

```
if(rb_block_given_p()) { rb_yield(rb_node); }
```

Node, Fragment

```ruby
fragment = Nokogiri::XML.fragment(string)
fragment = Nokogiri::HTML.fragment(string, encoding = nil)
# Note: Searching a fragment relative to the
# document root with xpath
# will probably not return what you expect.
# You should search relative to
# the current context instead. e.g.
fragment.xpath('//*').size #=> 0
fragment.xpath('.//*').size #=> 229
```

Node

```ruby
node = document.create_element('name')
```

```ruby
node.read_only?
node.blank?

node.type
node.cdata?
node.comment?
node.element?
node.fragment?
node.html?
node.text?
node.xml?
```

```ruby
# ATTRIBUTE_DECL: Attribute declaration type
# ATTRIBUTE_NODE: Attribute node type
# DOCB_DOCUMENT_NODE: DOCB document node type
# DOCUMENT_TYPE_NODE: Document type node type
# DTD_NODE: DTD node type
# ELEMENT_DECL: Element declaration type
# ENTITY_DECL: Entity declaration type
# ENTITY_NODE: Entity node type
# ENTITY_REF_NODE: Entity reference node type
# NAMESPACE_DECL: Namespace declaration type
# NOTATION_NODE: Notation node type
# PI_NODE: PI node type
# XINCLUDE_END: XInclude end type
# XINCLUDE_START: XInclude start type
```

Доступ

```
node['src']
node['src'] = 'value'
node.key?('src')
node.keys
node.values
node.delete('src')
node.each { |attr_name, attr_value| }
```

Детали

```
node == another_node # compares pointer_id
node.clone # optional depth
```

```
node.to_html(options={})
node.to_xml(options={})
node.to_s

node.inspect
node.pretty_print(pp)
```

Операции

```
node.traverse { |node| }
node.remove
node.replace(node_or_tags)
node.swap(node_or_tags)

node.next
node.next=(node_or_tags)
node.add_next_sibling
```

```
node.after(node_or_tags)
node.next_element
node.previous
node.previous=(node_or_tags)
node.before(node_or_tags)
node.previous_element
```

Reader

```
reader = Nokogiri::XML::Reader(string_or_io)
```

- Итератор по узлам
- Каждый узел будет доступен только один раз

Какой код лучше?

```
provider.address.city
```

```
provider.adress_city
```

```
provider.city
```

Закон Деметры

Есть O.m(params), m не трогает ничего кроме:

- собственно самого O
- параметров m
- других объектов, созданных в рамках m
- прямых компонентных объектов O
- глобальных переменных, доступных O, в пределах M

Нарушение в Nokogiri

```
doc = Nokogiri::Slop <<-EOXML
<employees>
  <employee status="active">
    <fullname>Dean Martin</fullname>
  </employee>
  <employee status="inactive">
    <fullname>Jerry Lewis</fullname>
  </employee>
</employees>
EOXML
```

Разбираем через точку

```
# navigate!
doc.employees.employee.last.fullname.content # => "Jerry Lewis"

# access node attributes!
doc.employees.employee.first["status"] # => "active"

# use some xpath!
doc.employees.employee("[@status='active']").fullname.content # => "Dean Martin"
doc.employees.employee(:xpath => "@status='active'").fullname.content # => "Dean Martin"

# use some css!
doc.employees.employee("[status='active']").fullname.content # => "Dean Martin"
doc.employees.employee(:css => "[status='active']").fullname.content # => "Dean Martin"
```

Комментарии разработчиков

1. Don't use this.
2. This may or may not be a backhanded compliment.
3. No, really, don't use this. If you use it, don't report bugs.
4. You've been warned!

Пишем дет

1. Символьное дифференцирование полинома
2. Символьное дифференцирование тригонометрических функций
3. Символьное интегрирование полинома
4. Символьное интегрирование тригонометрических функций
5. Численное интегрирование по формуле трапеций
6. Численное интегрирование по формуле прямоугольников
7. Численное интегрирование по формуле Симпсона
8. Умножение векторов (скалярное и векторное)
9. Уравнение прямой, проходящей через 2 точки на плоскости, через 2 точки в пространстве
10. Расстояние между двумя точками, между точкой и прямой, между точкой и плоскостью
11. Площадь круга, прямоугольника, треугольника, объем шара, параллелепипеда, пирамиды
12. Численное решение дифференциальных уравнений
13. Теория вероятности.

Ссылки

- https://github.com/rubocop-hq/ruby-style-guide
- https://nokogiri.org/
- https://ruby-doc.org/core-2.6.3/Math.html