Metaprogramação na prática

Guilherme Carreiro

TDC Florianópolis - 16.05.2015

Guilherme Carreiro
Campinas, SP

Ruby, JavaScript, Java

dextra/projects
karreiro/picuture_from
discourse/OneBox

PictureFrom

PictureFrom is the most efficient library for getting profile pictures.

Installation

```
gem install picture_from
```

Usage

Using PictureFrom is pretty simple! First, make sure the library is required:

```
require 'picture_from'
```

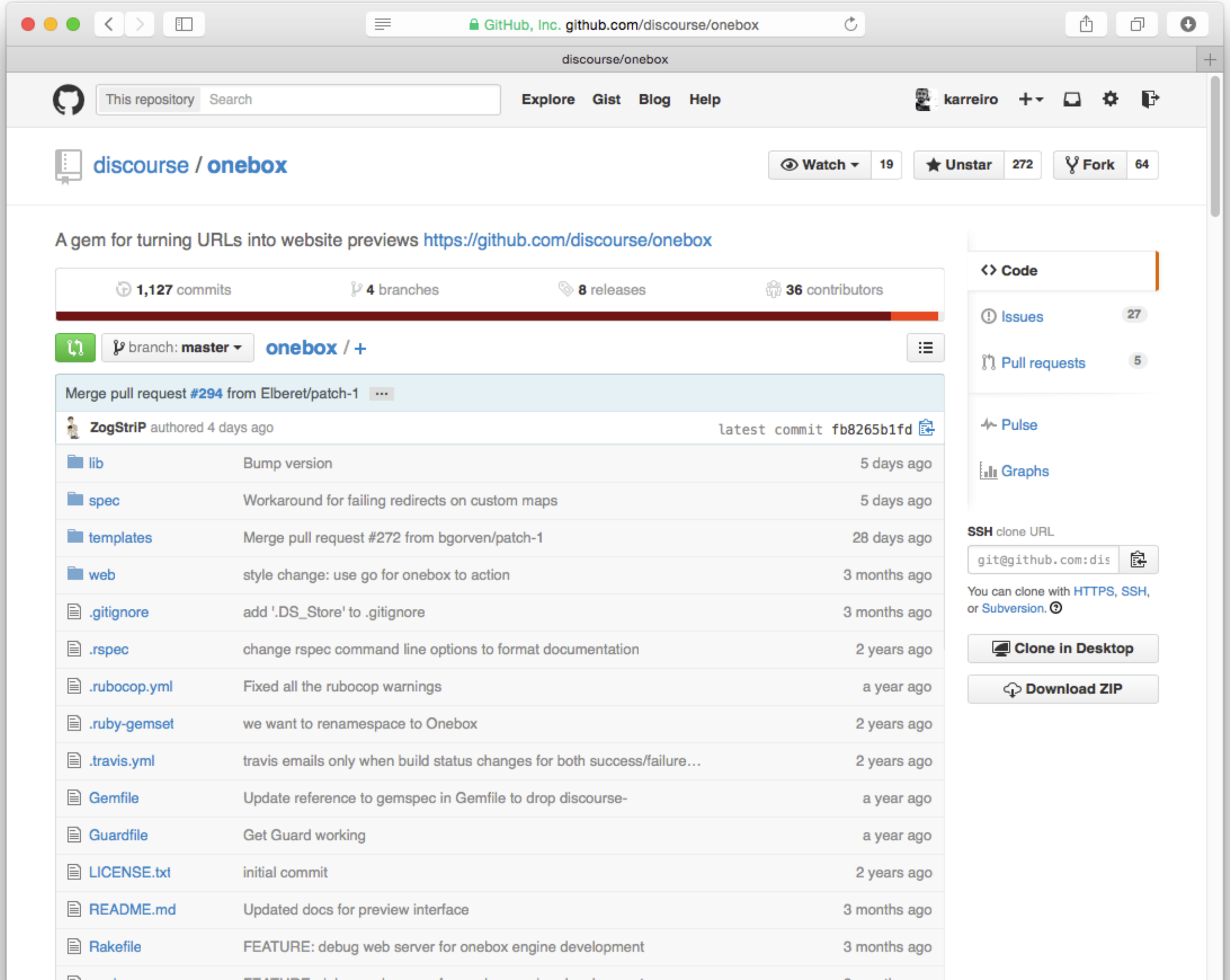
Pass an user information to the library's interface:

```
PictureFrom.url('zuck')
# => http://graph.facebook.com/zuck/picture

PictureFrom.url('Mark Zuckerberg')
# => http://graph.facebook.com/zuck/picture

PictureFrom.url('@unclebobmartin')
# => https://pbs.twimg.com/profile_images/1102364992/clean_code_72_color_400x400.png
```

Guilherme Carreiro
Campinas, SP

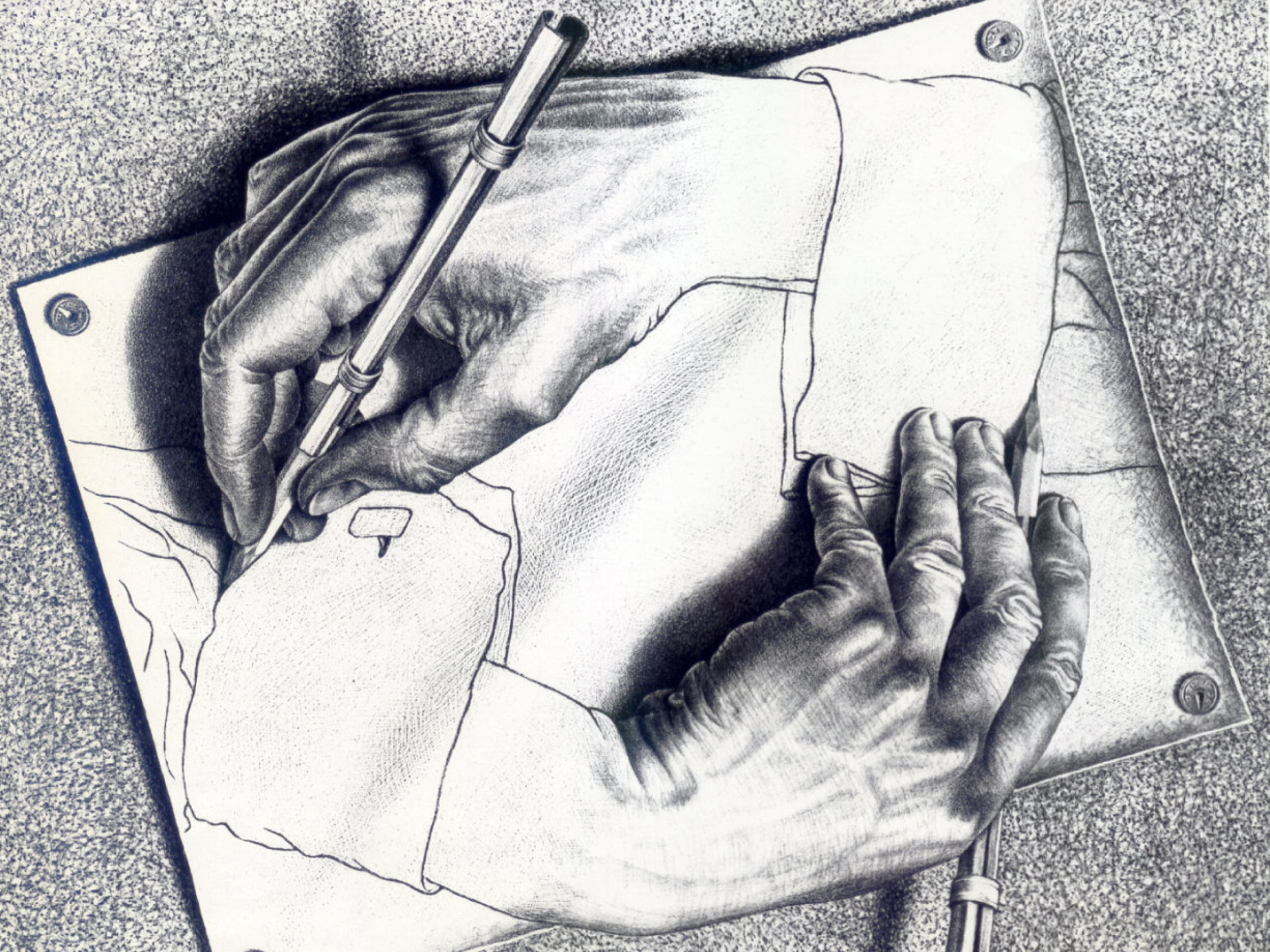
Ruby, JavaScript, Java

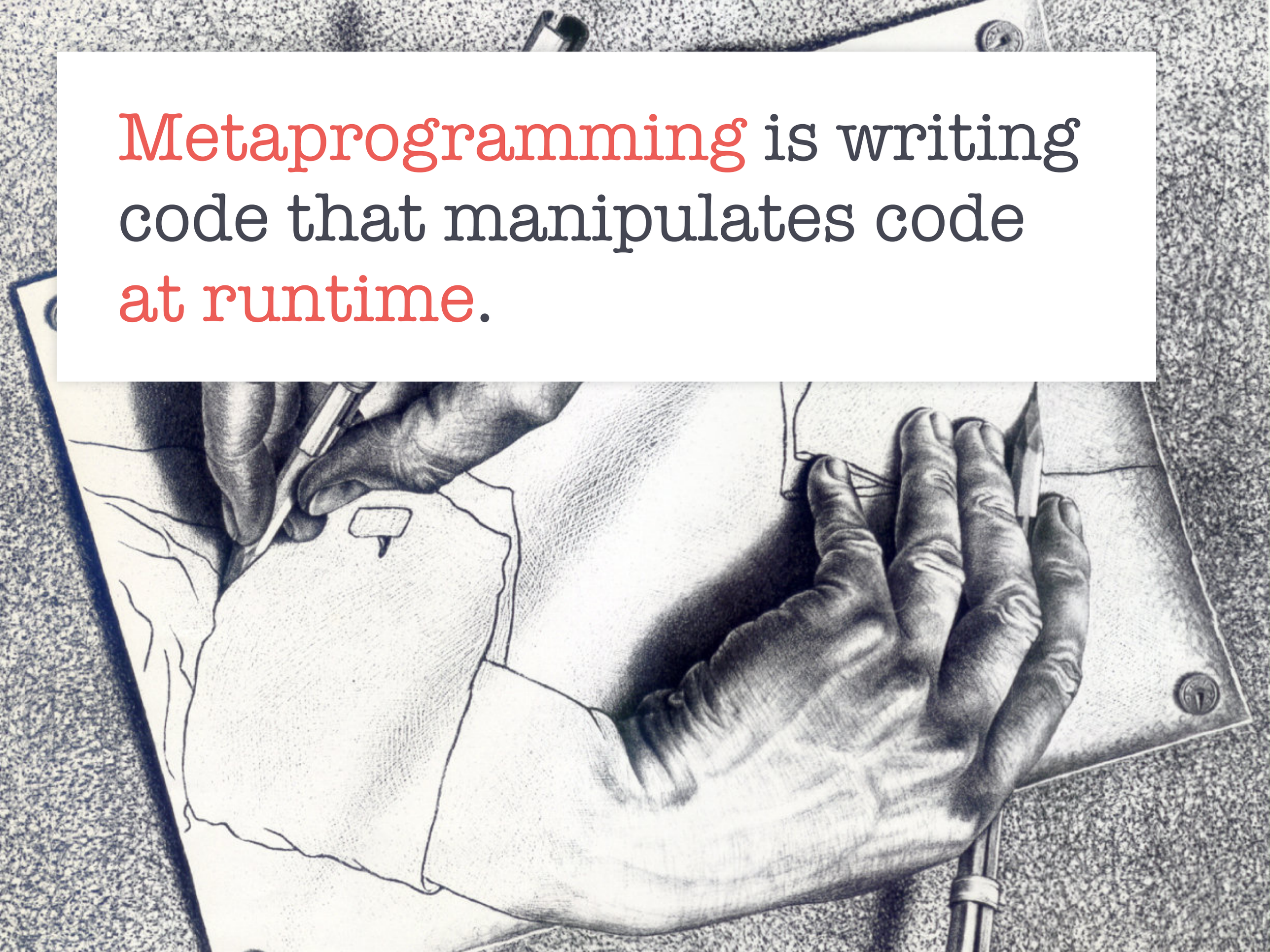
dextra/projects
karreiro/picuture_from
discourse/OneBox


GitHub, Inc. github.com/discourse/onebox
discourse/onebox
This repository
Search
Explore
Gist
Blog
Help
karreiro
discourse / onebox
Watch 19
Unstar 272
Fork 64
A gem for turning URLs into website previews https://github.com/discourse/onebox
1,127 commits
4 branches
8 releases
36 contributors
branch: master
onebox / +
Merge pull request #294 from Elberet/patch-1
ZogStriP authored 4 days ago
latest commit fb8265b1fd
lib Bump version 5 days ago
spec Workaround for failing redirects on custom maps 5 days ago
templates Merge pull request #272 from bgorven/patch-1 28 days ago
web style change: use go for onebox to action 3 months ago
.gitignore add '.DS_Store' to .gitignore 3 months ago
.rspec change rspec command line options to format documentation 2 years ago
.rubocop.yml Fixed all the rubocop warnings a year ago
.ruby-gemset we want to renamespace to Onebox 2 years ago
.travis.yml travis emails only when build status changes for both success/failure… 2 years ago
Gemfile Update reference to gemspec in Gemfile to drop discourse- a year ago
Guardfile Get Guard working a year ago
LICENSE.txt initial commit 2 years ago
README.md Updated docs for preview interface 3 months ago
Rakefile FEATURE: debug web server for onebox engine development 3 months ago
Code
Issues 27
Pull requests 5
Pulse
Graphs
SSH clone URL
git@github.com:dis
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP

```
eval(`rm -rf /`)
```

```ruby
class User < ActiveRecord::Base
  # empty
end

user = User.new(name: 'Guilherme', active: true)

user.active?
# => true
```





Metaprogramming is writing code that manipulates code at runtime.

```ruby
class User < ActiveRecord::Base
  # empty
end

user = User.new(name: 'Guilherme', active: true)

user.active?
# => true
```

```ruby
class User
  attr_accessor :name, :active
end

user = User.new.tap do |u|
  u.name = 'Guilherme'
  u.active = true
end

user.active?
# => NoMethodError: undefined method `active?' for
      #<User:0x007fe5d189a2b0 @name="Guilherme", @active=true>
```

#eval

```ruby
class User
  def self.define_methods
    attributes = [:name?, :active?]

    attributes.each do |attribute|
      class_eval <<-EOMETHOD
        def #{attribute}
          true
        end
      EOMETHOD
    end
  end
end

user = User.new
user.name?
# => NoMethodError: undefined method `name?' for #<User:0x007f...>

User.define_methods

user.name?
# => true
```

```ruby
class User
  def self.define_methods
    attributes = [:name?, :active?]

    attributes.each do |attribute|
      class_eval <<-EOMETHOD
        def #{attribute}
          true
        end
      EOMETHOD
    end
  end
```

```ruby
user = User.new
user.name?
# => NoMethodError: undefined method `name?' for #<User:0x007f...>
```

```ruby
User.define_methods

user.name?
# => true
```

```ruby
class User
  def self.define_methods
    attributes = [:name?, :active?]

    attributes.each do |attribute|
      class_eval <<-EOMETHOD
        def #{attribute}
          true
        end
      EOMETHOD
    end
  end
```

```ruby
user = User.new
user.name?
# => NoMethodError: undefined method `name?' for #<User:0x007f...>

User.define_methods
```

```ruby
user.name?
# => true
```

```ruby
class User
  def self.define_methods
    attributes = [:name?, :active?]

    attributes.each do |attribute|
      class_eval <<-EOMETHOD
        def #{attribute}
          true
        end
      EOMETHOD
    end
  end

user = User.new
user.name?
# => NoMethodError: undefined method `name?' for #<User:0x007f...>

User.define_methods

user.name?
# => true
```

```ruby
class User
  def self.define_methods
    attributes = [:name?, :active?]

    attributes.each do |attribute|
      class_eval <<-EOMETHOD
        def #{attribute}
          true
        end
      EOMETHOD
    end
  end
end

user = User.new
user.name?
# => NoMethodError: undefined method `name?' for #<User:0x007f...>

User.define_methods

user.name?
# => true
```

class_eval vs. instance_eval

```ruby
class User
  class_eval do
    def self.class_name
      '~ User ~'
    end
  end
end

User.class_name
# => "~ User ~”
```

```ruby
user = User.new

user.instance_eval do
  def self.instance_name
      '~ user ~'
  end
end

user.instance_name
# => "~ user ~”
```

Prefira const_get,
instance_variable_get, ou...

#define method

```ruby
class User < ActiveRecord::Base
  enum status: { active: 0, archived: 1 }
end

user = User.new

user.status
# => 'archived'

user.active?
# => false

user.active!
# => true

user.active?
# => true
```

```ruby
module ActiveRecord
  module Enum

    # (...)

    # Dado que value: 'active', name: 'status' e i: 0

    # def active?() status == 0 end
    define_method("#{value}?") { self[name] == i }

    # def active!() update! status: :active end
    define_method("#{value}!") { update! name => value }

    # (...)

  end
end
```

```ruby
class User
  attr_accessor :name, :status

  def active?
    self.status == 'active'
  end

  def archived?
    self.status == 'archived'
  end

  def deleted?
    self.status == 'deleted'
  end
end

User.new.tap { |u| u.status = 'active' }.active?
# => true
User.new.tap { |u| u.status = 'active' }.archived?
# => false
User.new.tap { |u| u.status = 'active' }.deleted?
# => false
```

```ruby
class User
  attr_accessor :name, :status

  STATUSES = [:active, :archived, :deleted]

  STATUSES.each do |status|
    define_method "#{status}?" do
      self.status == status.to_s
    end
  end
end

User.new.tap { |u| u.status = 'active' }.active?
# => true
User.new.tap { |u| u.status = 'active' }.archived?
# => false
User.new.tap { |u| u.status = 'active' }.deleted?
# => false
```

#method_missing

```ruby
class User
  attr_accessor :name, :active
end

user = User.new.tap do |u|
  u.name = 'Guilherme'
  u.active = true
end
```

```ruby
class CustomRecord
  def method_missing(name, *args, &block)
    name.to_s.end_with?('?') || super
    !!instance_variable_get("@#{name}".gsub /\?$/, '')
  end
end

class User < CustomRecord
  attr_accessor :name, :active
end
```

```ruby
user = User.new.tap do |u|
  u.name = 'Guilherme'
  u.active = true
end

user.active?
# => true
```

```ruby
class CustomRecord
  def method_missing(name, *args, &block)
    name.to_s.end_with?('?') || super
    !!instance_variable_get("@#{name}".gsub /\?$/, '')
  end
end

class User < CustomRecord
  attr_accessor :name, :active
end
```

```ruby
user = User.new.tap do |u|
  u.name = 'Guilherme'
  u.active = true
end

user.active?
# => true

user.respond_to? :active?
# => false
```

```ruby
class CustomRecord
  def method_missing(name, *args, &block)
    name.to_s.end_with?('?') || super
    !!instance_variable_get("@#{name}".gsub /\?$/, '')
  end

  def respond_to_missing?(name, include_all)
    name.to_s.end_with?('?') || super
  end
end

class User < CustomRecord
  attr_accessor :name, :active
end

user = User.new.tap do |u|
  u.name = 'Guilherme'
  u.active = true
end

user.respond_to? :active?
# => true
```

```ruby
class CustomRecord
  def method_missing(name, *args, &block)

    name.to_s.end_with?('?') || super

    self.class.send :define_method, name do
      !!instance_variable_get("@#{name}".gsub /\?$/, '')
    end

    send(name)
  end

  def respond_to_missing?(name, include_all)
    name.to_s.end_with?('?') || super
  end
end

user = User.new.tap { |u| u.name = 'Guilherme'; u.active = true }
```

```ruby
class CustomRecord
  def method_missing(name, *args, &block)
    'TDC' * 1_000_000 # complex stuff
    name.to_s.end_with?('?') || super

    self.class.send :define_method, name do
      !!instance_variable_get("@#{name}".gsub /\?$/, '')
    end

    send(name)
  end

  def respond_to_missing?(name, include_all)
    name.to_s.end_with?('?') || super
  end
end

user = User.new.tap { |u| u.name = 'Guilherme'; u.active = true }

puts Benchmark.measure { user.active? }
# => 0.000000   0.000000   0.000000 (  0.001539)
puts Benchmark.measure { user.active? }
# => 0.000000   0.000000   0.000000 (  0.000021)
```

```ruby
class User
  def initialize(friends)
    @friends = friends
  end
end

guilherme = User.new [
  'Danilo', 'Eder', 'Luan', 'Paulo', 'Sabrina', 'Tiago'
]

guilherme.number_of_friends
# => 6
```

attr_*

```ruby
class User
  attr_accessor :name, :active
end

user = User.new.tap do |u|
  u.name = 'Guilherme'
  u.active = true
end

user.name
# => "Guilherme"
```

```c
static VALUE
rb_mod_attr_accessor(int argc, VALUE *argv, VALUE klass)
{
    int i;
    for (i=0; i<argc; i++) {
        rb_attr(klass, rb_to_id(argv[i]), TRUE, TRUE, TRUE);
    }
    return Qnil;
}

void
rb_attr(VALUE klass, ID id, int read, int write, int ex)
{
  // (...) algumas coisas

  if (read) {
    rb_add_method(klass, id, VM_METHOD_TYPE_IVAR, (void *)attriv…
  }
  if (write) {
    rb_add_method(klass, rb_id_attrset(id), VM_METHOD_TYPE_ATTRSET…
  }
}
```

Qual é o melhor?

Qual é o melhor?

eval, method_missing, define_method

http://tenderlovemaking.com/2013/03/03/dynamic_method_definitions.html

#send

```ruby
def search_articles_by_user(user)
  if logged_user.admin?
    Article.admin_scope.search_by(user)
  elsif logged_user.customer_admin?
    Article.customer_admin_scope.search_by(user)
  elsif logged_user.customer?
    Article.customer_scope.search_by(user)
  elsif logged_user.user?
    Article.user_scope.search_by(user)
  end
end
```

```ruby
def search_articles_by_user(user)
  user_levels = [:admin, :customer_admin, :customer, :user]

  user_levels.each do |level|
    if logged_user.send "#{level}?"
      Article.send("#{level}_scope").search_by(user)
    end
  end
end
```

Hooks

Hooks

included, extended, prepended, inherited

```ruby
class User
  include ImageTrick
  attr_accessor :name, :image
  has_attached_file :image
end

user = User.new.tap { |u| u.image = '123456.png' }
user.image_url
# => "https://imagetrick.com/123456.png"
```

```ruby
module ImageTrick
  def self.included(klass)
    klass.extend(ClassMethods)
  end
  module ClassMethods
    def has_attached_file(name)
      define_method "#{name}_url" do
        image_name = instance_variable_get("@#{name}")
        "https://imagetrick.com/#{image_name}"
      end
    end
  end
end

class User
  include ImageTrick
  attr_accessor :name, :image
  has_attached_file :image
end

user = User.new.tap { |u| u.image = '123456.png' }
user.image_url
# => "https://imagetrick.com/123456.png"
```

Monkey Patch
(Open classes)

```ruby
module ImageTrick
  def self.included(klass)
    klass.extend(ClassMethods)
  end
  module ClassMethods
    def has_attached_file(name)
      define_method "#{name}_url" do
        image_name = instance_variable_get("@#{name}")
        "https://imagetrick.com/#{image_name}"
      end
    end
  end
end

class User
  include ImageTrick
  attr_accessor :name, :image
  has_attached_file :image
end

user = User.new.tap { |u| u.image = '123456.png' }
user.image_url
# => "https://imagetrick.com/123456.png"
```

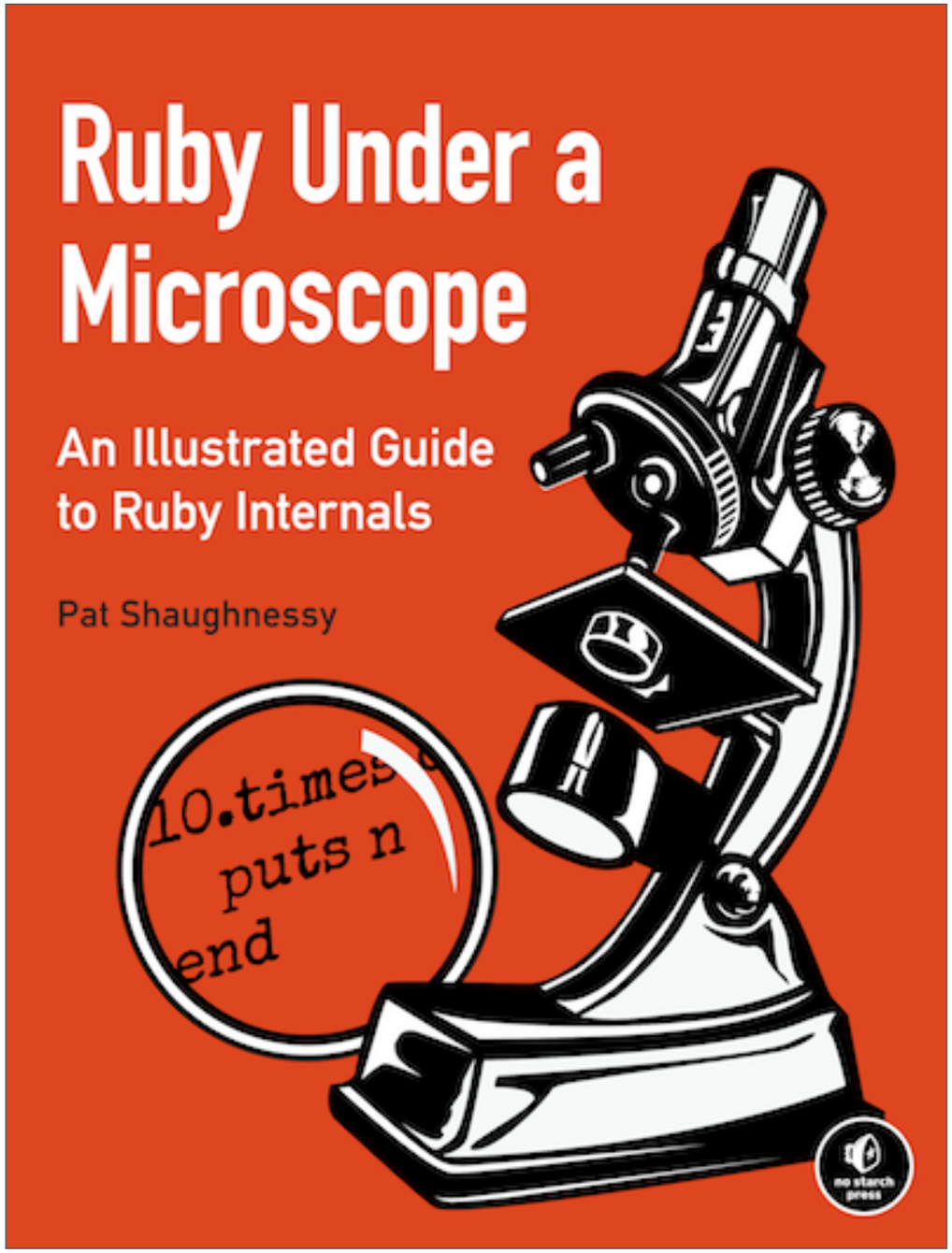
```ruby
module ImageTrick
  def self.included(klass)
    klass.extend(ClassMethods)
  end
  module ClassMethods
    def has_attached_file(name)
      define_method "#{name}_url" do
        image_name = instance_variable_get("@#{name}")
        "https://imagetrick.com/#{image_name}"
      end
    end
  end
end
```

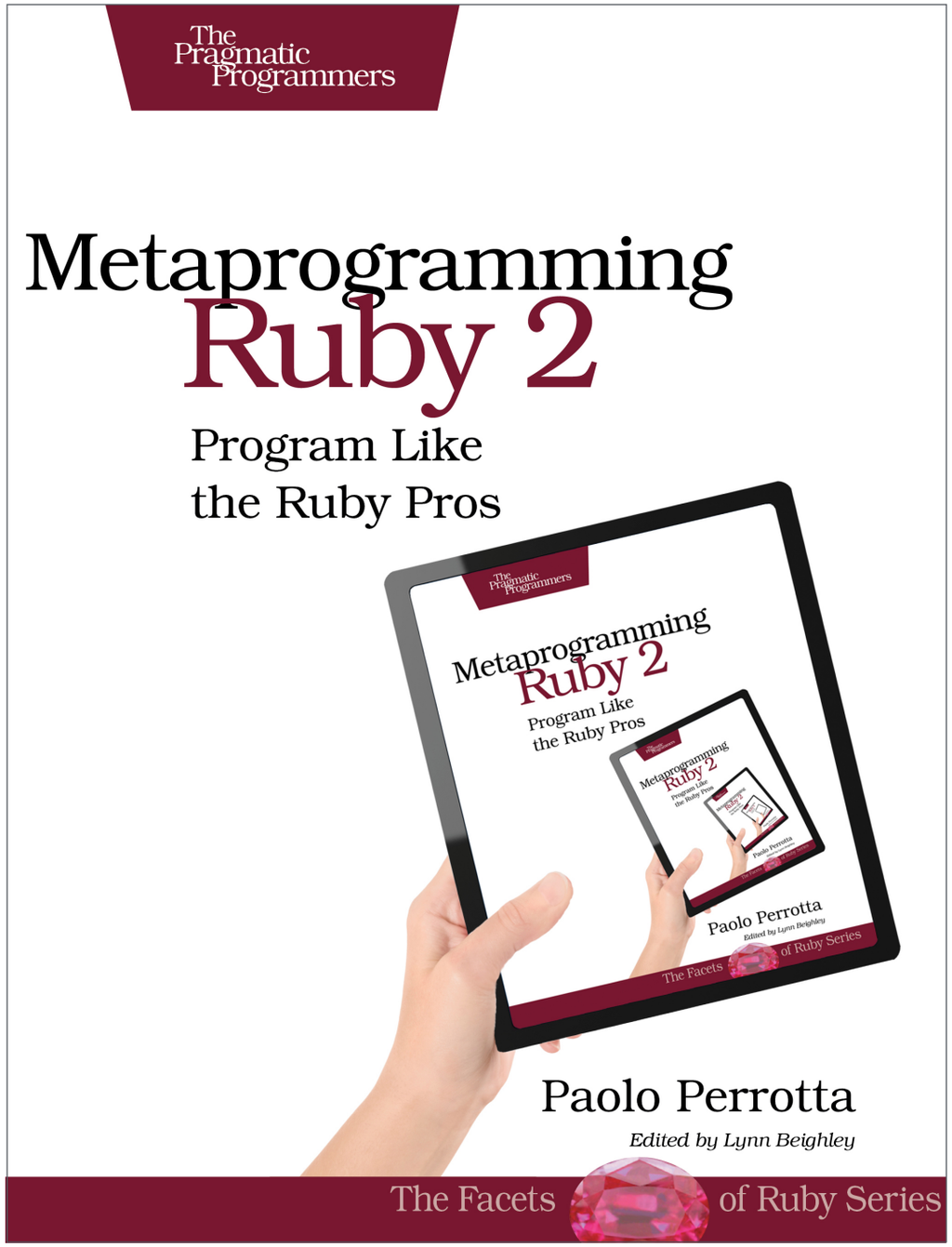
```ruby
module ImageTrick
  def self.included(klass)
    klass.extend(ClassMethods)
  end
  module ClassMethods
    def has_attached_file(name)
      define_method "#{name}_url" do
        image_name = instance_variable_get("@#{name}")
        "https://imagetrick.com/#{image_name}"
      end
    end
  end
end
```

```ruby
class String
  def serialized
    self.gsub(/[^a-zA-Z0-9\.\:\/\-_]+/, '-').downcase
  end
end

module ImageTrick
  def self.included(klass)
    klass.extend(ClassMethods)
  end
  module ClassMethods
    def has_attached_file(name)
      define_method "#{name}_url" do
        image_name = instance_variable_get("@#{name}")
        "https://imagetrick.com/#{image_name}".serialized
      end
    end
  end
end
```

```ruby
class String
  def serialized
    self.gsub(/[^a-zA-Z0-9\.\:\/\-_]+/, '-').downcase
  end
end

module ImageTrick
  def self.included(klass)
    klass.extend(ClassMethods)
  end
  module ClassMethods
    def has_attached_file(name)
      define_method "#{name}_url" do
        image_name = instance_variable_get("@#{name}")
        "https://imagetrick.com/#{image_name}".serialized
      end
    end
  end
end

"Some string".serialized
# => "some-string"
```

```ruby
module ImageTrick
  module RefinedString
    refine String do
      def serialized
        self.gsub(/[^a-zA-Z0-9\.\:\/\-_]+/, '-').downcase
      end
    end
  end
  def self.included(klass)
    klass.extend(ClassMethods)
  end
  module ClassMethods
    using RefinedString
    def has_attached_file(name)
      define_method "#{name}_url" do
        image_name = instance_variable_get("@#{name}")
        "https://imagetrick.com/#{image_name}".serialized
      end
    end
  end
end
"Some string".serialized
# => undefined method `serialized' for "Some string” (NoMethodError)
```


Ruby Under a Microscope
An Illustrated Guide to Ruby Internals
Pat Shaughnessy
10.times {
puts n
end
no starch press


The Pragmatic Programmers
Metaprogramming Ruby 2
Program Like the Ruby Pros
Paolo Perrotta
Edited by Lynn Beighley
The Facets of Ruby Series

Obrigado.

Guilherme Carreiro
GitHub: @karreiro, Twitter: @karreiro_