Ruby Tips & Quirks #2

Michał Łomnicki

www.starware.com.pl

Zmienne lokalne

```
puts local_variables.inspect

y = 4

puts local_variables.inspect
```

Zmienne lokalne

```
puts local_variables.inspect # => ["y"]

y = 4

puts local_variables.inspect # => ["y"]
```

Zmienne lokalne

Ruby 1.8

```ruby
puts local_variables.inspect # => ["y"]

y = 4

puts local_variables.inspect # => ["y"]
```

Zmienne lokalne

Ruby 1.9

```ruby
puts local_variables.inspect # => [:y]

y = 4

puts local_variables.inspect # => [:y]
```

String vs Symbol

```
class String
    unless instance_methods.include?("camelize")
        def camelize
            gsub(/\/(.?)/) { "::#{$1.upcase}" }
            .gsub(/(?:^|_)(.)/) { $1.upcase }
        end
    end
end
```

String vs Symbol

```
class String
    unless instance_methods.any? { |m| m.to_s == "camelize
        def camelize
            gsub(/\/(.?)/) { "::#{$1.upcase}" }
            .gsub(/(?:^|_)(.)/) { $1.upcase }
        end
    end
end
```

String vs Symbol

```ruby
class String
    unless method_defined?(:camelize)
        def camelize
            gsub(/\/(.?)/) { "::#{$1.upcase}" }
            .gsub(/(?:^|_)(.)/) { $1.upcase }
        end
    end
end
```

module functions

```
module Security
    def generate_password
        ('a'..'z').sample(8)
    end
end

class User
    include Security
end

User.new.generate_password
```

module functions

```
module Security
    def generate_password
        ('a'..'z').sample(8)
    end
end
```

module functions

```
module Security
    extend self
    def generate_password
        ('a'..'z').sample(8)
    end
end
```

module functions

```
module Security
    extend self
    def generate_password
        ('a'..'z').sample(8)
    end
end

Security.generate_password
```

module functions

```
module Security
    module_function
    def generate_password
        ('a'..'z').sample(8)
    end
end

Security.generate_password
```

respond_to?(:super)

```ruby
module Sanitizer
    def save
        puts "sanitized"
        super
    end
end

module Persistance
    def save
        puts "saved"
        super
    end
end

class User
    include Persistance
    include Sanitizer
end
```

respond_to?(:super)

```ruby
module Sanitizer
    def save
        puts "sanitized"
        super
    end
end

module Persistance
    def save
        puts "saved"
        super
    end
end

class User
    include Persistance
    include Sanitizer
end
```

respond_to?(:super)

```
 > User.new.save
=> sanitized
=> saved
=> save: super: no superclass method save (No
```

respond_to?(:super)

```ruby
module Sanitizer
    def save
        puts "sanitized"
        super if respond_to?(:super)
    end
end

module Persistance
    def save
        puts "saved"
        super if respond_to?(:super)
    end
end

class User
    include Persistance
    include Sanitizer
end
```

respond_to?(:super)

```
module Sanitizer
    def save
        puts "sanitized"
        super if respond_to?(:super)
    end
end

module Persistance
    def save
        puts "saved"
        super if respond_to?(:super)
    end
end

class User
    include Persistance
    include Sanitizer
end
```

respond_to?(:super)

```
> User.new.save
=> sanitized
```

respond_to?(:super)

```
module Sanitizer
    def save
        puts "sanitized"
        super if defined?(super)
    end
end

module Persistance
    def save
        puts "saved"
        super if defined?(super)
    end
end

class User
    include Persistance
    include Sanitizer
end
```

respond_to?(:super)

```
 > User.new.save

=> sanitized

=> saved
```

Class.include

```
module Sanitizer
    def save
        puts "sanitized"
        super if defined(super)
    end
end

module Persistance
    def save
        puts "saved"
        super if defined(super)
    end
end

class User
    include Persistance
    include Sanitizer
end
```

Class.include

```
module Sanitizer
    def save
        puts "sanitized"
        super if defined(super)
    end
end

module Persistance
    def save
        puts "saved"
        super if defined(super)
    end
end

class User
    include Persistance, Sanitizer
end
```

Class.include

```
 > User.new.save
=> saved
=> sanitized
```

Class.include

```
class User
    include Persistance, Sanitizer
end
```

Class.include

```
class User
    include Sanitizer, Persistance
end
```

Class.include

```
 > User.new.save
=> sanitized
=> saved
```

Block comments

```
=begin
  Objects don't specify their attributes
  directly, but rather infer them from the table definitic
  with which they're linked. Adding, removing, and changir
  attributes and their type is done directly in the databa
=end
```

Ruby1.9 - each_with_object

```
 > (1..5).inject({}) do |i, hsh|
          hsh[i] = i*2
          hsh
    end

=> {1=>2, 2=>4, 3=>6, 4=>8, 5=>10}
```

VS

```
> (1..5).each_with_object({}) do |i, hsh|
    hsh[i] = i*2
   end

=> {1=>2, 2=>4, 3=>6, 4=>8, 5=>10}
```

Ruby1.9 - public_send

```
class User
    protected
    def destroy
        puts "destroyed"
    end
end
User.new.public_send(:destroy)

NoMethodError: protected method destroy calle
```

Ruby1.9 - ObjectSpace.count_objects

```
ObjectSpace.count_objects
{
    :TOTAL=>76928,
    :FREE=>549,
    :T_OBJECT=>1363,
    :T_CLASS=>1008,
    :T_MODULE=>38,
    :T_FLOAT=>7,
    :T_STRING=>50339,
    :T_REGEXP=>234,
    :T_ARRAY=>7259,
    :T_HASH=>558,
    :T_FILE=>16,
    :T_DATA=>1695,
}
```

Ruby1.9 define_finalizer

```
str = "ruby1.9"

ObjectSpace.define_finalizer(str) do |object_id|
    puts "string was destroyed id: #{object_id}"
end

str = nil
GC.start

=> string was destroyed id: -607935038
```

Ruby1.9 call proc

```
prc = proc { puts "proc called" }
```

1) prc.call(1) # 1.8

2) prc[2] # 1.8

3) prc.(3) # new

4) prc.===(4) # new

Ruby1.9 call proc

```ruby
sleep_time = proc do |time|
    case time.hour
    when 0..6 then true
    else false
    end
end

case Time.now
when sleep_time
  puts "go to bed. now!"
else
  puts "work harder"
end
```

Ruby1.9 call proc

```
sleep_time = proc do |time|
    case time.hour
    when 0..6 then true
    else false
    end
end

case Time.now
when sleep_time
  puts "go to bed. now!"
else
  puts "work harder"
end

sleep_time.===(Time.now)
```

Pytania?