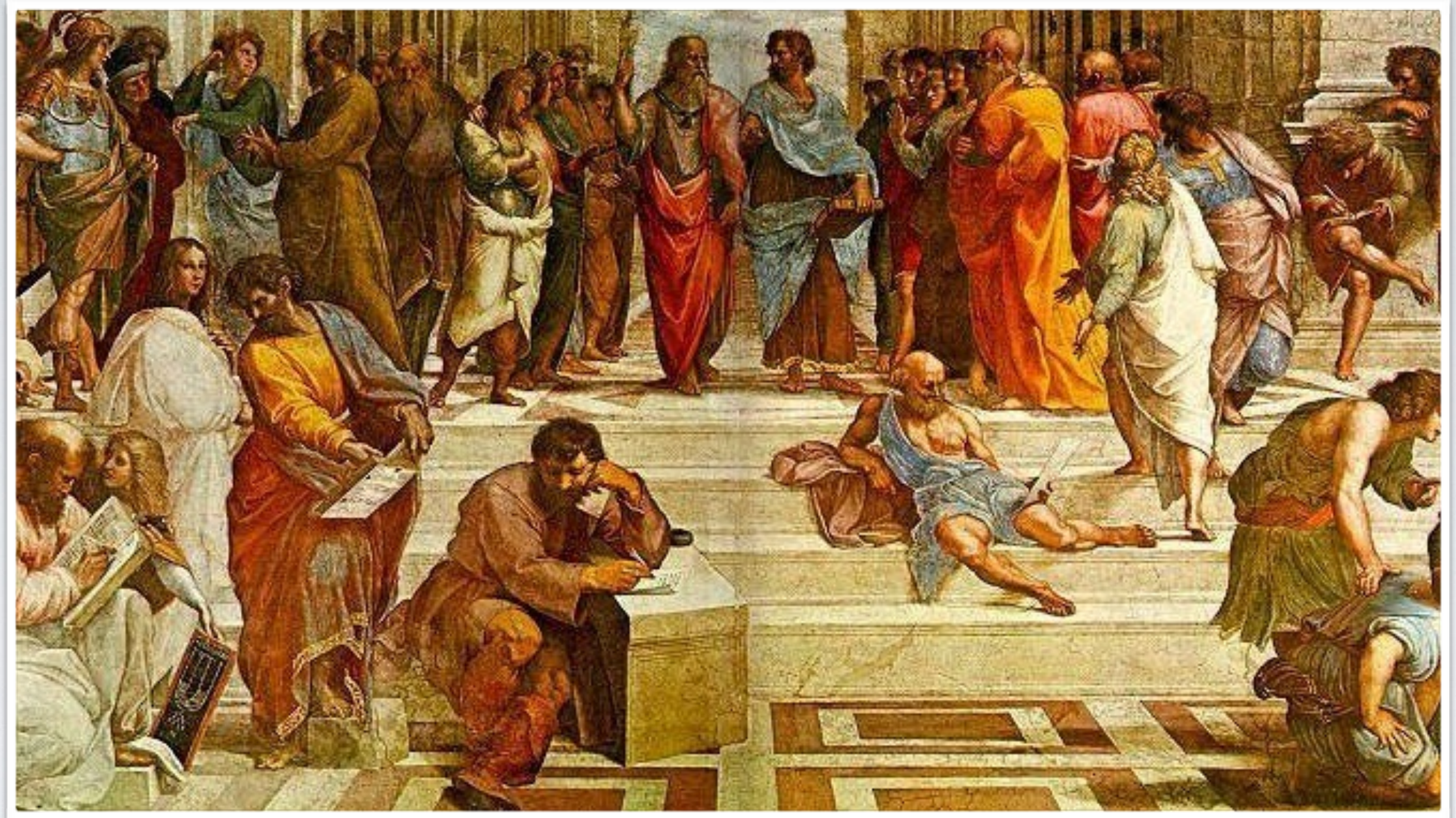
RUBY

Интерпретируемый, динамичный, человеколюбивый



ФИЛОСОФИЯ RUBY

твой путь — твои решения

- Принцип наименьшей неожиданности
- Минимальное время не исполнения программы, но разработки
- Множество возможных путей решения задачи
- Просто, но не примитивно
- Программы должны писаться так, чтобы их понимал человек, и лишь иногда - машина

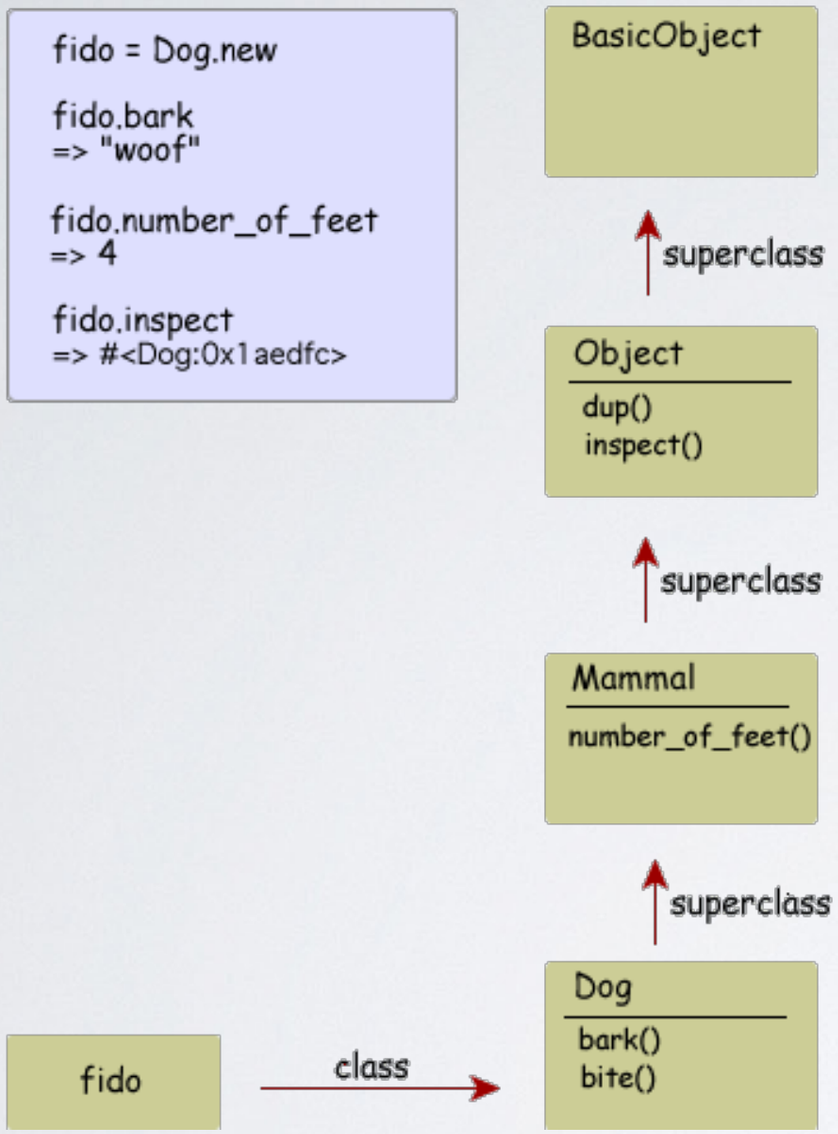
ОБЪЕКТНАЯ МОДЕЛЬ РУБИ

вместо Адама — BasicObject

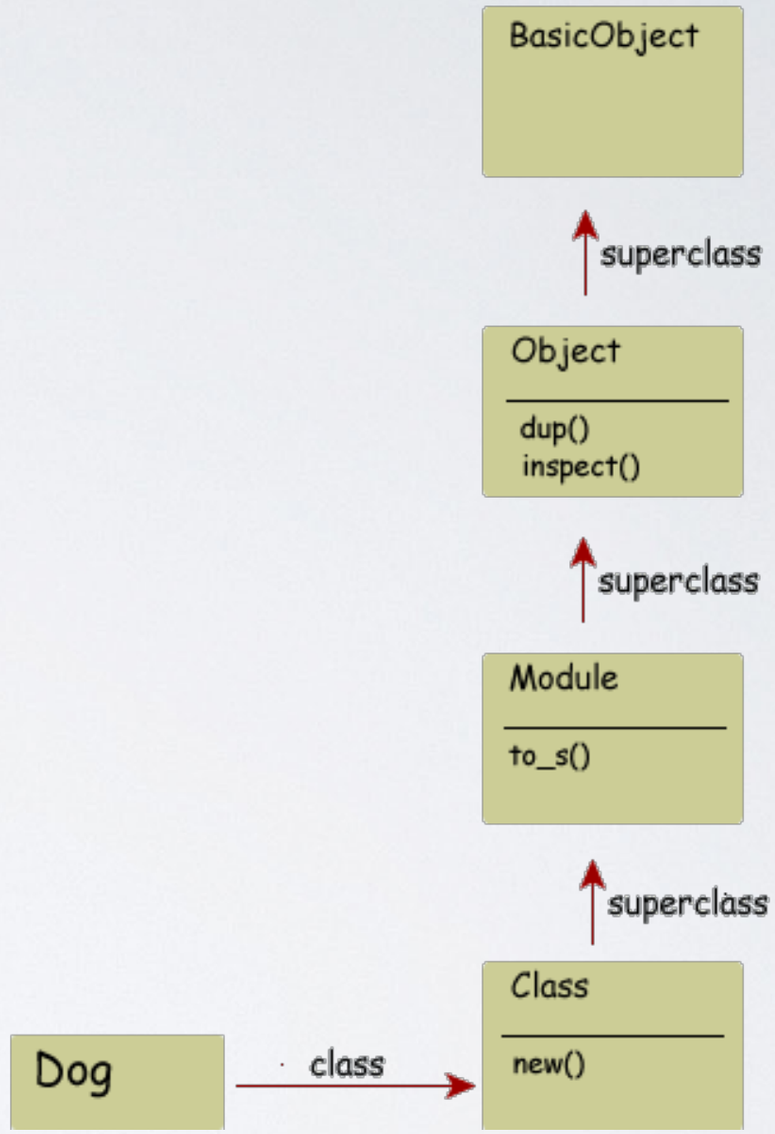
ВСЕ ОЧЕНЬ ХОРОШО!

- Нет примитивных типов
- Каждый объект - экземпляр класса
- Нет абстрактных классов
- Нет множественного наследования
- Mixins предпочтительней наследования
- Утиная типизация

Класс - это конструктор объектов (экземпляров класса, *instances of class*).



Но и сам класс - это объект в цепочке этих конструкторов!



Классы, выступая объектами, становятся производными от анонимных синглтон-классов (singleton class).

Модуль — это именованная группа, которая содержит методы, которые можно подмешивать к

Объекту

include:
добавляет методы модуля объекту.

```ruby
module MyModule
  def goal
    'Служить экземпляру класса!'
  end
end

class MyClass
  include MyModule
end

puts MyClass.new.goal
  # => 'Служить экземпляру класса!'
puts MyClass.goal
  # => [NoMethodError] undefined method
  # `goal' for MyClass:Class
```

Классу

extend:
вызывает **include** для синглтон-класса объекта.

```ruby
module MyModule
  def goal
    'Служить классу!'
  end
end

class MyClass
  extend MyModule
end

puts MyClass.goal
  # => 'Служить классу!'
puts MyClass.new.goal
  # => [NoMethodError] undefined method
  # `goal' for #<MyClass:0x007fa49a081768>
```

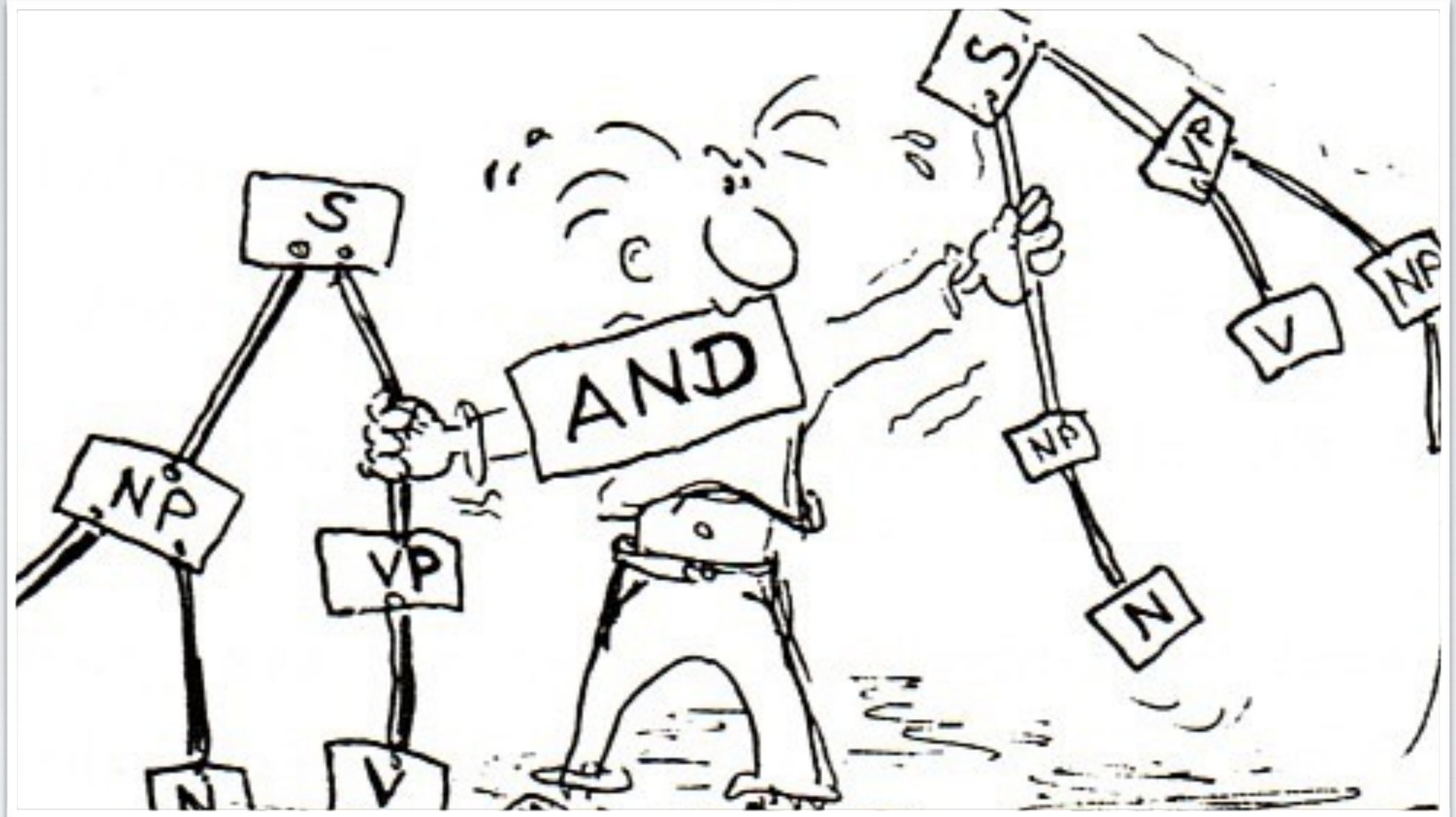
МОДУЛИ (КОНТЕЙНЕРЫ МЕТОДОВ)

```ruby
module Module1
  def method
    puts 'Module1#method'
  end

  def ancestor_method
    super
  end
end

module Module2
  def method
    puts 'Module2#method'
  end

  def ancestor_method
    super
  end
end

class Parent
  def method
    puts 'Parent#method'
  end
  alias :ancestor_method :method
end
```

```ruby
class Child < Parent
  include Module1
  include Module2

  def method
    puts 'Child#method'
    puts super
  end

  def ancestor_method
    super
  end
end

Child.new.method
  # => выведет "Child#method",
  # а потом "Module2#method"
Child.new.ancestor_method
  # => Parent#method
Child.ancestors
  # => [Child, Module2, Module1, Parent,
  #     Object, Kernel, BasicObject]
```

ПОЛЬЗОВАТЕЛЬСКИЕ КЛАССЫ

```ruby
# $x глобальная переменная
# @x - инстанс-переменная экземпляра класса
# @@x - переменная класса
# y - локальная переменная с ограниченной областью видимости

class MyClass
  attr_reader :x

  @@x = 'class variable'
  def initialize
    y = 'local variable'
    @x = 'instanse object variable'
  end

  def get_class_variable
    @@x
  end
end

object = MyClass.new
object.x #=> 'instanse object variable'
object.get_class_variable #=> 'class variable'
y # => [NameError] undefined local variable or method `y' for main:Object
```

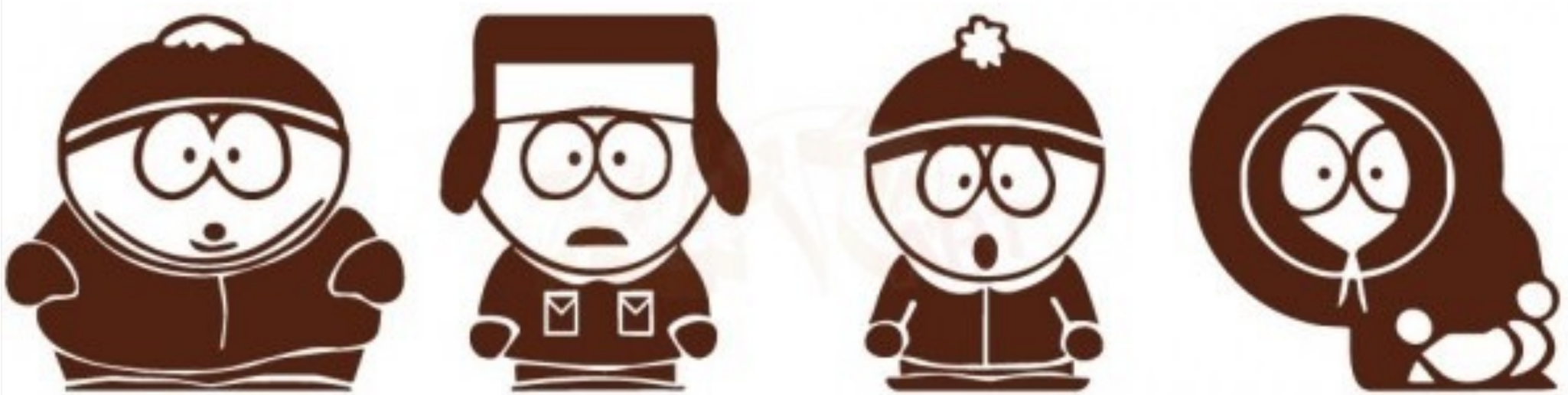


ОСОБЕННОСТИ СИНТАКСИСА

минимум управляющих конструкций

БАЗОВЫЕ ОПЕРАЦИИ

```ruby
# присвоение
x = 1
x = 'string'
x = [1, 2]

# сравнение
1 == 2  #=> false

# продвинутое сравнение
1 <=> 2 #=> -1

# условия
if 1 == 1
  true
else
  false
end  #=> true

'correct condition' if 1 == 1 #=> 'correct condition'

0 == 1 ? true : false         #=> false

# все операции в ruby - методы
1.==(2)  #=> false
1.<=>(2) #=> -1

# методы конвертации
1.to_s   #=> '1'
'1'.to_i #=> 1

# переменная - это укатель на объект
s1 = 'Hello World'
s2 = s1
p s1.object_id # => 70144764317700
p s2.object_id # => 70144764317700
```



БАЗОВЫЕ КЛАССЫ

String, Symbol, Numeric, Hash, Array

STRING

```ruby
# инициация
'hello world!'.class #=> String
"hello world!"
%{hello world!}
String.new('test').class #=> String

# экранирование
'it\'s test example' #=> it's test example
"symbol \"A\"" #=> symbol "A"

# конкатенация
s1 = 'my '
s2 = 'test'

s1 + s2 #=> my test
s1 # => my

s1 << s2 #=> my test
s1 # => my test

# подстановка в строку
my_age = 29
'my age: ' + my_age #=> [TypeError] no implicit conversion of Fixnum into String
"my age #{my_age}" #=> my age: 29
```

SYMBOL

```ruby
:symbol.class          #=> Symbol

# один и тот же символ всегда имеет один и тот же id
:symbol.object_id      #=> 384328
:symbol.object_id      #=> 384328
'string'.object_id     #=> 70273663768340
'string'.object_id     #=> 70273663768280

# может создаваться на базе строки
s = 'thing'
:"#{s}".object_id      #=> 405448
:thing.object_id       #=> 405448

# конвертация
:thing.to_s            #=> 'thing'
'thing'.to_sym         #=> :thing
```

NUMERIC

```ruby
(2**30-1).class  #=> Fixnum
(2**30).class  #=> Bignum
1.1.class #=> Float


4/3 #=> 1
(4/3).class #=> Fixnum

4.0/3 #=> 1.3333333333333333
(4/3.0).class #=> Float

rand(100) #=> random Fixnum value from 0 to 99
rand #=> random Float value (like 0.4876133333333333)

```

ARRAY

```ruby
# массив это упорядоченная по индексу коллекция объектов
array = Array.new  #=> []
Array.new(3)       #=> [nil, nil, nil]
Array.new(3, true) #=> [true, true, true]
[true, true, true] #=> [true, true, true]

# может быть смешенного типа
[1, 'test', false, nil] #=> [1, 'test', false, nil]

# многомерные массивы
empty_table = Array.new(3) { Array.new(3) }
#=> [[nil, nil, nil], [nil, nil, nil], [nil, nil, nil]]

```

HASH

```ruby
# хэш это неупорядоченная группа пар ключ-значение
array = Hash.new           #=> {}
{:k1 => 'a', :k2 => 'b'} #=> :k1=>"a", :k2=>"b"}
{k1: 'a', k2: 'b'}         #=> :k1=>"a", :k2=>"b"}

# определяем хэш со значением по умолчанию
h = Hash.new(5)
h[:any_key]                #=> 5

# ключами и значениями могут объекты любого класса
h = {:key => 'value'}
{h => :test, 1 => 2}       #=> {{:key=>"value"}=>:test, 1=>2}
```

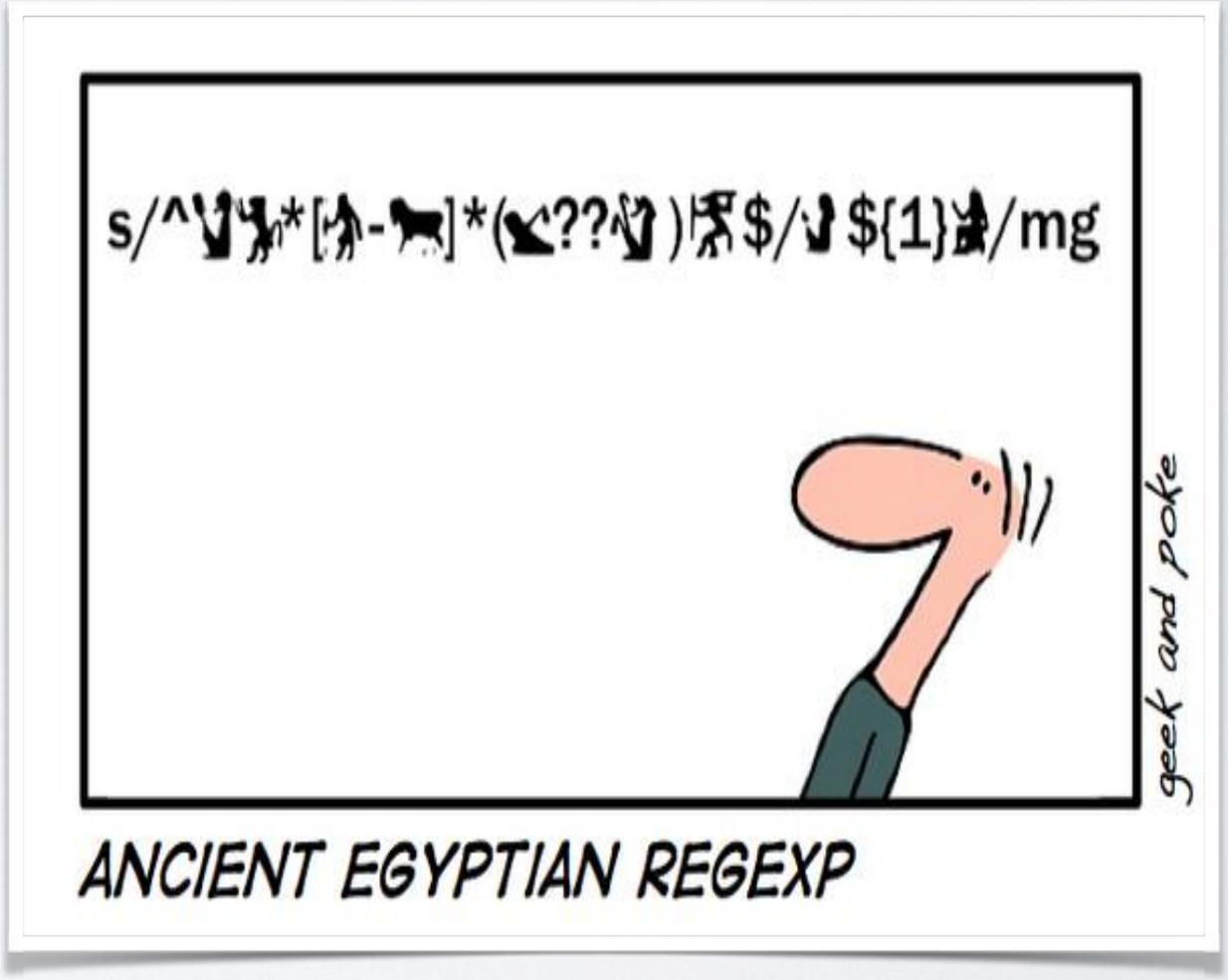
ENUMERATORS

```ruby
3.times do
  puts "Hello"
end

(0..3).each do |i|
  puts i*i
end

%w(январь февраль март).each do |m|
  puts m
end

0.upto(3) do |k|
  puts k
end

4.downto(1) do |t|
  puts t**3
end
```

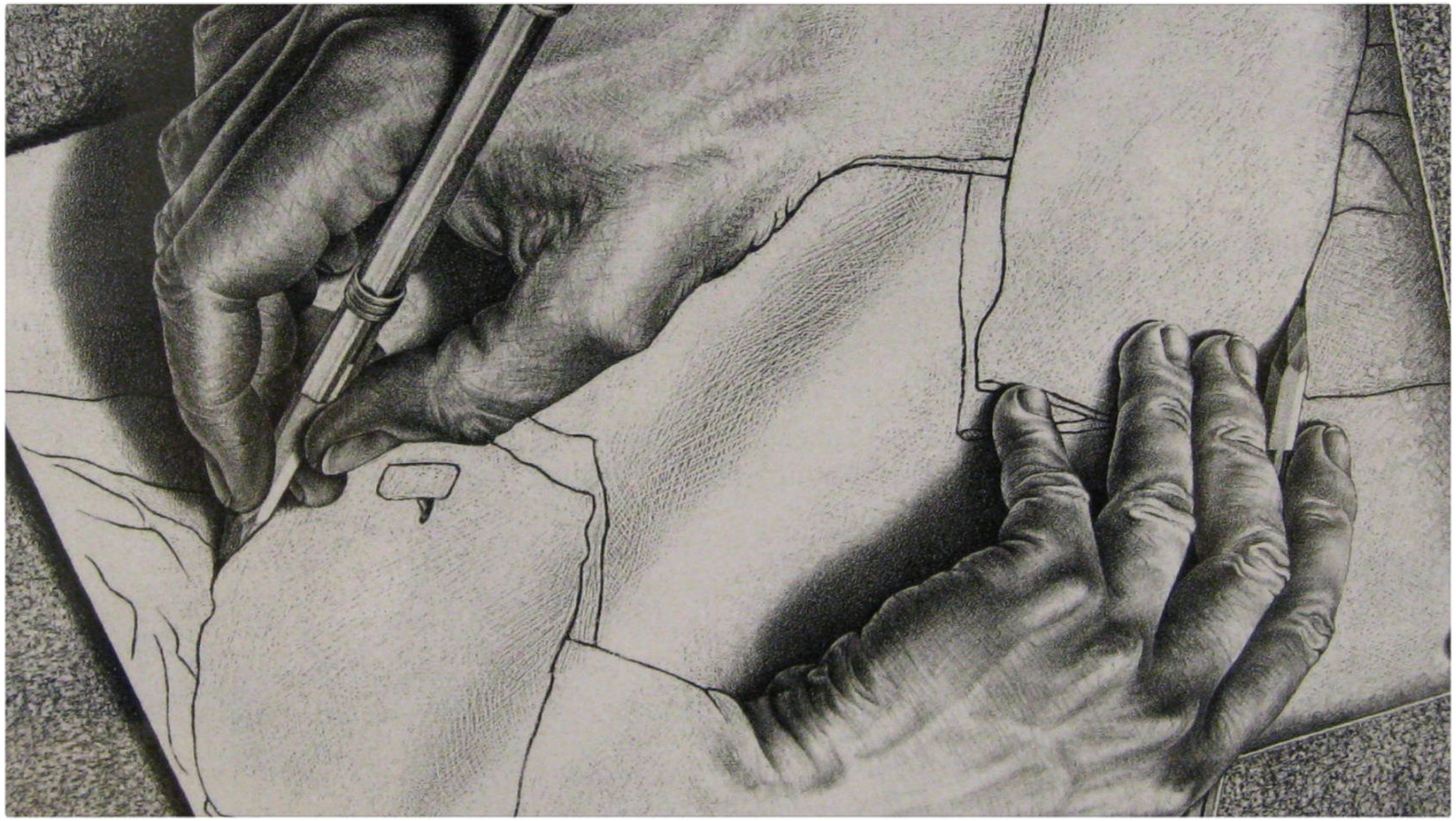
РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Регулярные выражения (англ. **regular expressions**) — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.

(via) wiki



http://rubular.com/



МЕТАПРОГРАММИРОВАНИЕ

Создаем магию вместе с Ruby DSL

БЛОКИ

```ruby
def open_file_and_do_smth
  f = File.open('test')
  yield(f.read)
  f.close
  inside_param = 'Какой блок? Ничего о нем не знаю и знать не хочу'
end

outside_param = 'А я с блоком старые кореша. '

open_file_and_do_smth do |text|
  puts text # => Я строка. Мой жизненный цикл - эта презентация.
            # Вы хоть сочувствуете мне?
  outside_param += 'Подтверждаю'
end

puts outside_param # => 'А я с блоком старые кореша. Подтверждаю'

open_file_and_do_smth do |text|
  puts inside_param # => [NameError] undefined local variable
                    # or method `inside_param' for main:Object
end

puts { :key1 => 'value1', :key2 => 'value2' } # syntax error
puts :key1 => 'value1', :key2 => 'value2' # так всё нормально
```

```ruby
class Person
  attr_accessor :name, :age,
                :position, :ideology

  def initialize
    yield(self)
  end
end

spartak = Person.new do |his|
  his.name = "Alex Mikitenko"
  his.age = 29
  his.position = 'qa'
  his.ideology = 'anarchist'
end
```

Proc и lambda:

блоки с отложенным выполнением

Proc

```
1  my_proc = Proc.new { puts "Это proc" }
2  puts my_proc
3  # => #<Proc:0x007fbe6b889f10@test.rb:1>
```

- Игнорирует лишние или недостающие аргументы
- **return** перемещает нас в окружение, где был определен **proc**, и выполняет **return** из него

lambda

```
1  my_lambda = lambda { puts "Это lambda" }
2  my_lambda = -> { puts "Это lambda" }
3  puts my_lambda
4  # => #<Proc:0x007f8b5c1c5658@test.rb:6 (lambda)>
```

- Педантично ожидает ровно того количества аргументов, которые было объявлено
- **return** прерывает выполнение блока и возвращает управление в окружение, где была вызвана **lambda**

DSL

Предметно-ориентированный язык (англ. Domain Specific language, DSL — «предметно-специфичный язык») — язык программирования, специализированный для конкретной области применения (в противоположность языку общего назначения[en], применимому к широкому спектру областей и не учитывающему особенности конкретных сфер знаний)

(via) wiki

```ruby
class LoginPage
  include Watirsome

  text_field :username, label: 'Username'
  text_field :password, label: 'Password'
  button :submit_login, text: 'Login'

  def login(username, password)
    self.username = username
    self.password = password
    submit_login
  end
end

browser = Watir::Browser.new
page = LoginPage.new(browser)
page.login('demo', 'demo')
```

https://github.com/p0deje/watirsome

НЕКОТОРЫЕ ДИНАМИЧЕСКИЕ ВКУСНОСТИ

method_missing:

```ruby
class MyClass
  def method_missing(name)
    puts "#{name} ? Здесь таких нет !"
  end
end

MyClass.new.undefined_method
  # => undefined_method ? Здесь таких нет !
```

const_set:

```ruby
def create_class(name)
  Object.const_set(name,
    Class.new do
      def hello_class
        "Привет из нового класса!"
      end
    end
  )
end

create_class 'MyClass'
MyClass.new.hello_class
# => Привет из нового класса!
```

define_method:

```ruby
class MyClass
  %i[a b c].each do |method|
    define_method(method) { puts method.to_s }
  end
end

object = MyClass.new
object.a # => a
object.b # => b
object.c # => c
```

EXCEPTIONS

- обработка исключений выполняется в рамках управляющих конструкций begin…end
- перехват исключения производит метод rescue или его алиас fail
- обязательная для выполнения часть кода идет после метода ensure

```ruby
begin
  puts 'value is ' + 5
    # => выведет в консоль
    # "[TypeError] no implicit conversion of Fixnum into String"
    # "А я выполнюсь в любом случае!"
rescue TypeError => e
  puts "[#{e.class}] #{e}"
ensure
  puts "А я выполнюсь в любом случае!"
end
```

БИБЛИОТЕКИ (GEMS)

В консоли:

- *gem install rails* => установить конкретный gem
- *gem list* => получить список всех gems
- …

В приложении:

- *require 'rails'* => подключить rails gem

RBENV, RMV, BUNDLER

- **rvm, rbenv** — это системы управления версиями ruby
- **bundler** — это система управления пакетами (gems) ruby.

Getting Started

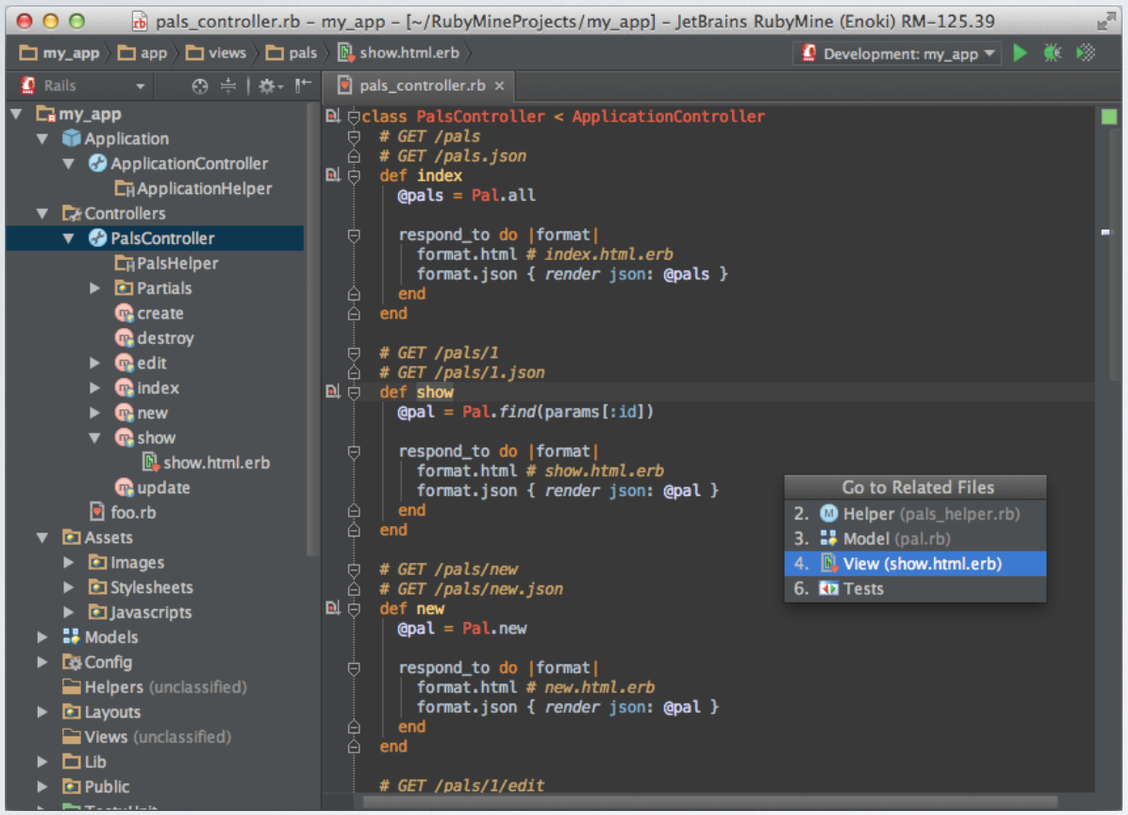
Getting started with bundler is easy! Open a terminal window and run this command:

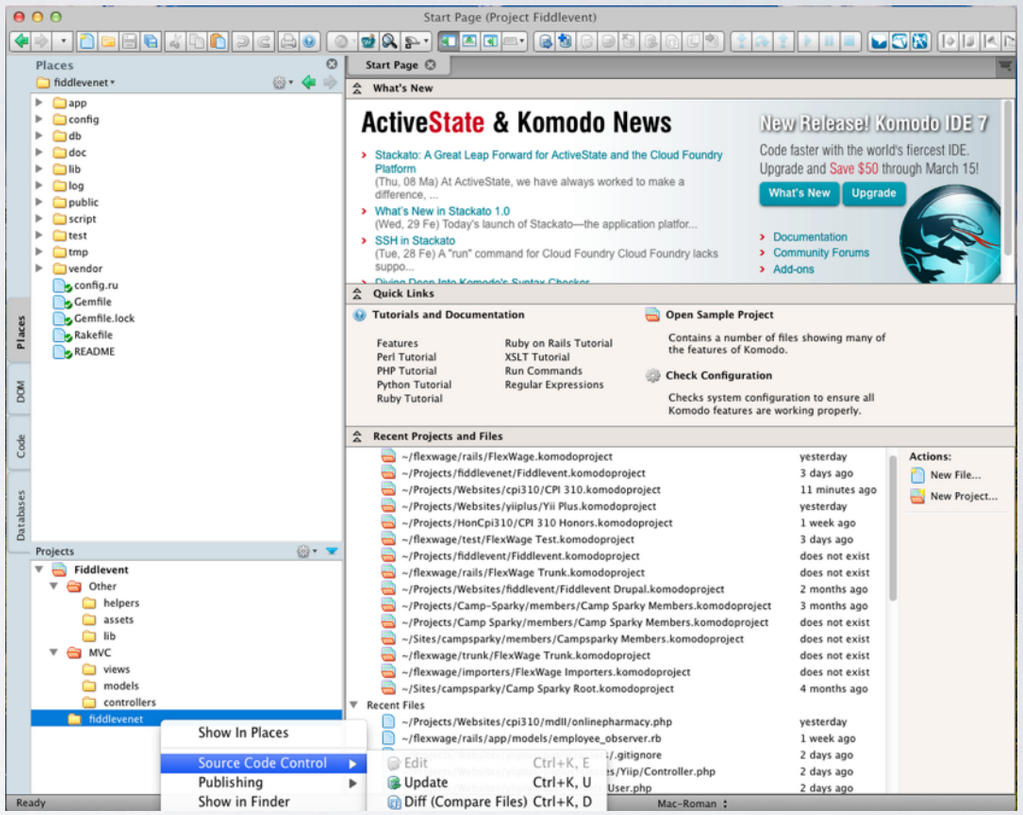
```
$ gem install bundler
```

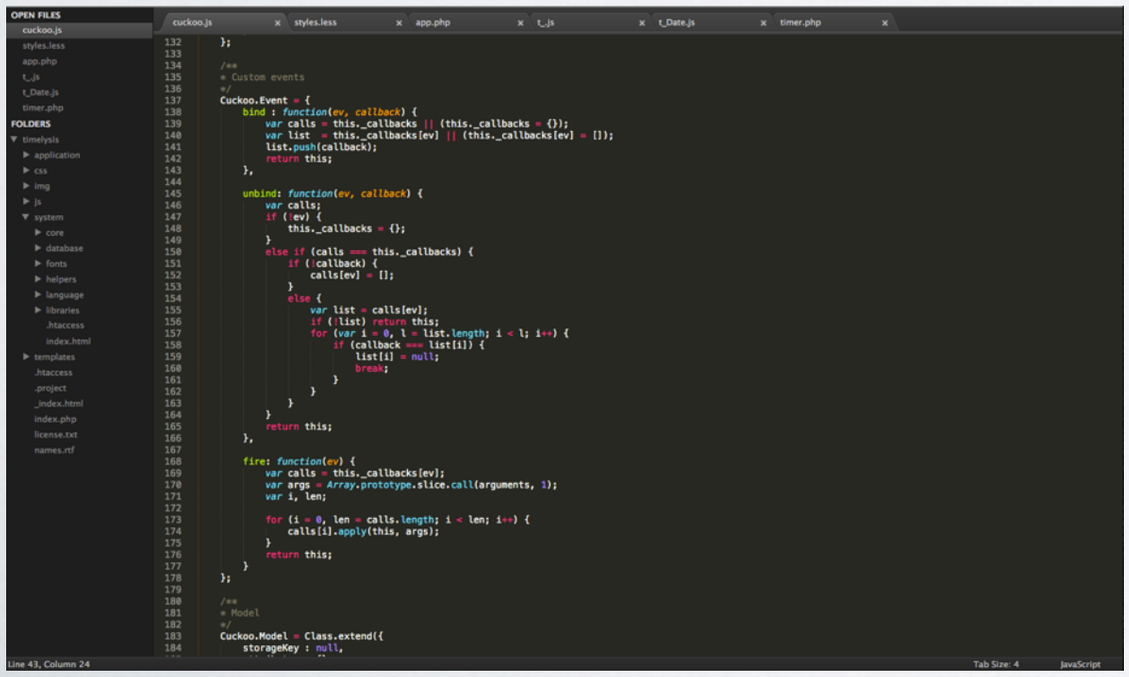
Specify your dependencies in a Gemfile in your project's root:

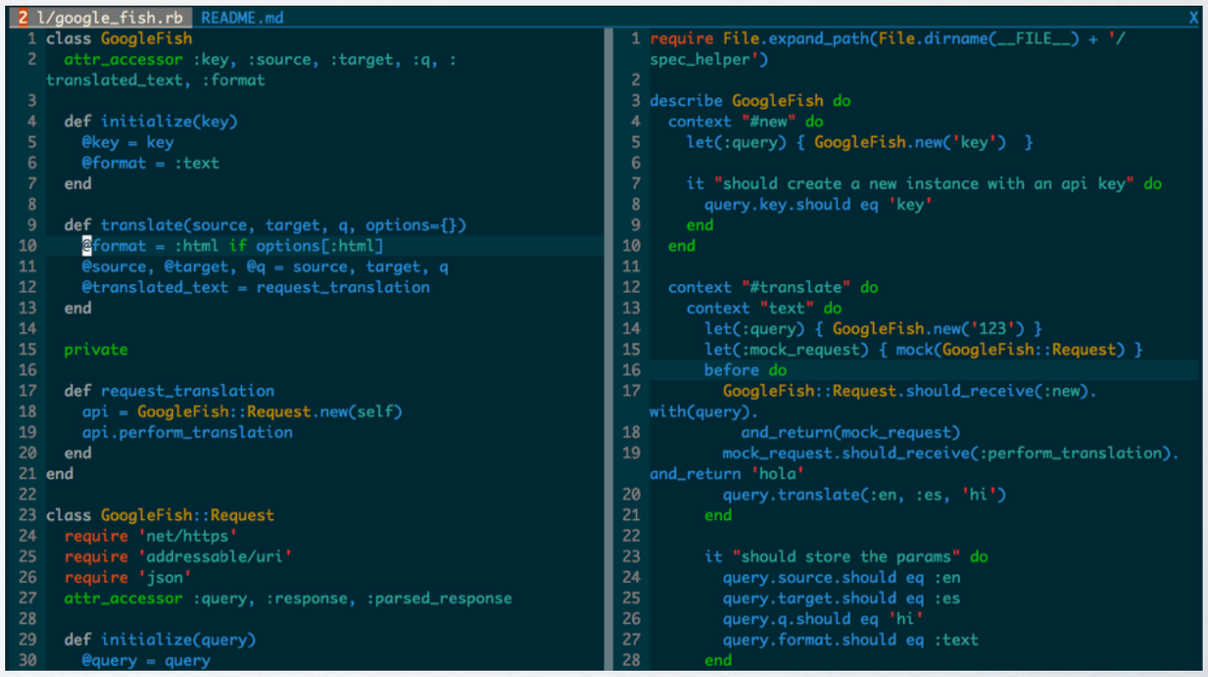
```
source 'https://rubygems.org'
gem 'nokogiri'
gem 'rack', '~>1.1'
gem 'rspec', :require => 'spec'
```

СРЕДА РАЗРАБОКИ









RUBY STYLE GUIDE

RuboCop

https://github.com/bbatsov/ruby-style-guide

ВО СЛАВУ RUBY!

ИЗВЕСНЫЕ ПРОЕКТЫ

- **Github**: без комментариев
- **Basecamp**, **Redmine**: онлайн-инструменты для управления проектами и задачами
- **Indiegogo**: краудфандинговая платформа

(via) http://designwebkit.com/

ИЗВЕСНЫЕ ГЕМЫ

- Rails
- HAML
- Nokogiri
- Cucumber
- Rspec
- Rake
- Bundler
- Capybara
- Watir

ИНКАРНАЦИИ RUBY

ДОМАШНЕЕ ЗАДАНИЕ

- Сгенерируйте массив со случайными числами и отсортируйте его, не используя метод :sort

- Создайте объект, у которого будет метод, отличительный от всех остальных объектов этого же класса

ПОЛЕЗНЫЕ МАТЕРИАЛЫ

- Изучаем Ruby, Фитцджеральд М.
- Язык программирования Ruby, Флэнаган Д., Мацумото Ю.
- Metaprogramming Ruby 2, Paolo Perrotta
- Введение в Ruby ООП: http://nashbridges.me/introducing-ruby-oop
- Модули ruby: http://habrahabr.ru/post/143990/