

## Лекция 5. Особенности использования языка Ruby

- Функциональный стиль программирования
- Примеры компактного кода
- Области применения языка



# Функциональное программирование

---



29.09.2020

- Функция в основе всего
- Функция может быть аргументом другой функции
- Хранится не состояние программы, а функции и аргументы
- Значения неизменны.  
Новое значение – новое имя
- Функции должны быть чистыми

# Достоинства функционального программирования

---



29.09.2020

- Компактный код
- Упрощение модульного тестирования
- Отложенные вычисления
- Распараллеливание без участия программиста
- Lisp, Scheme, Erlang, Haskell, Clojure...
- **Ruby не является языком функционального программирования!**

# Предпосылки функционального стиля.


## Блоки



29.09.2020

- Метод с блоком:

```
def test_func (x)
  x.each_index do |i|
    yield i+1
  end
end
test_func(5) { |n| s="#{n}:"; n.times {s += '*'}; puts s; }
```



- Результат выполнения

```
1.*
2.**
...
```

- Эквивалентное преобразование кода:

```
def test_func (x)
  x.each_index do |i|
    n=i+1; s="#{n}:"; n.times {s += '*'}; puts s;
  end
end
test_func(5)
```

# Предпосылки функционального стиля. Блоки



29.09.2020

```
class Array
  # универсальный метод поиска
  def find
    for i in 0...size
      # получаем текущий элемент массива
      value = self[i]
      # вызываем код блока для проверки условия. Если true – выходим.
      return value if yield(value)
    end
    # ничего не нашли. Выходим.
    return nil
  end
end
# задаём массив и сложное условие поиска v*v > 30
[1, 3, 5, 7, 9].find { |v| v * v > 30 } # => 7
```

- D. Thomas, C.Fowler, A. Hunt. Programming Ruby 1.9. The Pragmatic Programmers` Guide. - Texas.Dallas: The Pragmatic Programmers, 2010

# Блоки

## Определение формата вызова

---



29.09.2020

```
class File
  def self.my_open(*args)
    result = file = File.new(*args)
    # Если блок есть, вызвать его код и закрыть файл.
    if block_given?
      result = yield file
      file.close
    end
    # result содержит либо объект File, либо то, что вернул блок
    return result
  end
end
```

```
f = File.my_open("file.txt"); f.puts 123; f.close
File.my_open("file.txt") { |f| f.puts 123 }
```

- D. Thomas, C.Fowler, A. Hunt. Programming Ruby 1.9. The Pragmatic Programmers` Guide. - Texas.Dallas: The Pragmatic Programmers, 2010

# Класс Proc



29.09.2020

- Объект как код для выполнения

```
a = Proc.new { |name| puts "Hello, #{name}!" }  
a.call 'world' # => Hello, world!
```

- Результат выполнения = результат последней операции

```
b = Proc.new { |x| x * x }  
c = b.call 4  
puts c # 16
```

# Proc:lambda



29.09.2020

- Объект как функция

- Компактная форма записи

р a = ->(x) { x \* x }

р a.call 2 # получим 4

- Традиционная форма записи

р b = lambda { |x| x + x }

р b.call 3 # получим 6





29.09.2020

# lambda vs Proc

---

- Proc – контейнер кода

```
def proc_test
  Proc.new { puts 'proc'; return }.call
  puts 'proc_test end' # сюда не попадаем никогда!
end
proc_test # => proc
```

- Код внутри lambda изолирован

```
def lambda_test
  ->{ puts 'lambda'; return }.call
  puts 'lambda_test end'
end
lambda_test # => lambda
            #      lambda_test end
```

# lambda для функции высшего порядка



29.09.2020

- Применение lambda для функции высшего порядка

```
some_func = ->(x) { x + 3 }  
def g(func, x); func.call( x ) * func.call( x ) end  
puts g(some_func, 7) # 100
```

```
a = ->(x) { x * x }  
p (1..9).map(&a) # [1, 4, 9, 16, 25, 36, 49, 64, 81]
```

# Функции высшего порядка через блоки



29.09.2020

```
def g(x); (yield x) * (yield x) end
```

- Анонимная функция в форме блока

```
puts g(7) { |x| x + 3 } # 100
```

- Передача функции как блока

```
f1 = ->(x) { x + 3 }  
puts g(7, &f1) # 100
```

- Преобразование method -> lambda

```
def f2(x); x+3 end  
puts (f2_l = method(:f2).to_proc).lambda? # true  
puts g(7, &f2_l) # 100
```

# Функциональный стиль

---



29.09.2020

- Нет изменений. Только создание новых объектов.
- В Ruby чистый функциональный стиль приводит к избыточному хранению данных!
- Нельзя слепо следовать функциональному стилю!
- Примеры взяты из:  
<https://github.com/tokland/tokland/wiki/RubyFunctionalProgramming>

# Функциональный стиль

## Массивы

---



29.09.2020

- Императивный стиль

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

```
# меняем массив
```

```
indexes << 4
```

```
indexes # [1, 2, 3, 4]
```

- Функциональный стиль

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

```
# создаем новый массив
```

```
all_indexes = indexes + [4] # [1, 2, 3, 4]
```

# Функциональный стиль

## Хэш

---



29.09.2020

- Императивный стиль

```
hash = {:a => 1, :b => 2}
```

```
# меняем объект hash
```

```
hash[:c] = 3
```

- Функциональный стиль

```
hash = {:a => 1, :b => 2}
```

```
# создаём новый объект hash
```

```
new_hash = hash.merge(:c => 3)
```

# Функциональный стиль

## Строка

---



29.09.2020

- Императивный стиль

```
string = 'hello'
```

```
# меняем строку
```

```
string.gsub(/l/, 'z')
```

```
string # "hezzo"
```

- Функциональный стиль

```
string = 'hello'
```

```
# создаём новую строку
```

```
new_string = string.gsub(/l/, 'z') # "hezzo"
```

# Функциональный стиль

## Добавление по условию

---



29.09.2020

- Императивный стиль

```
output = []
```

```
output << 1
```

```
output << 2 if i_have_to_add_two
```

```
output << 3
```

- Функциональный стиль

```
# создаем массив, затем убираем пустые, создав  
# НОВЫЙ ОБЪЕКТ
```

```
output = [1, (2 if i_have_to_add_two), 3].compact
```



# Использование переменных

---



29.09.2020

- **Неправильно:**

# переопределяем переменную

```
number = gets
```

```
number = number.to_i
```

- Правильно:

# **новый тип значения** — новая переменная

```
number_string = gets
```

```
number = number_string.to_i
```

# Пустой массив + each + вставка значения → map



29.09.2020

- **Неправильно:**

```
dogs = []  
['milu', 'rantanplan'].each do |name|  
  dogs << name.upcase  
end  
dogs # => ["MILU", "RANTANPLAN"]
```

- **Правильно:**

```
dogs = ['milu', 'rantanplan'].map do |name|  
  name.upcase  
end # => ["MILU", "RANTANPLAN"]
```

```
dogs = ['milu', 'rantanplan'].map { |name| name.upcase } # => ["MILU", "RANTANPLAN"]  
dogs = ['milu', 'rantanplan'].map(&:upcase) # => ["MILU", "RANTANPLAN"]
```

# Пустой массив + each + добав. по условию → select/reject



29.09.2020

- **Неправильно:**

```
dogs = []  
['milu', 'rantanplan'].each do |name|  
  if name.size == 4  
    dogs << name  
  end  
end  
dogs # => ["milu"]
```

- **Правильно:**

```
dogs = ['milu', 'rantanplan'].select do |name|  
  name.size == 4  
end # => ["milu"]
```

# Инициализация + each + накопление результата → inject

---



29.09.2020

- **Неправильно:**

```
length = 0
['milu', 'rantanplan'].each do |dog_name|
  length += dog_name.length
end
length # => 14
```

- Правильно (accumulator обеспечивает накопление):

```
length = ['milu', 'rantanplan'].inject(0) do |accumulator, dog_name|
  accumulator + dog_name.length
end # => 14
```

# Инициализация + [присвоение по условию + ]\* ...

---



29.09.2020

- **Неправильно**

```
name = obj1.name
```

```
name = obj2.name if !name
```

```
name = ask_name if !name
```

- **Правильно**

```
name = obj1.name || obj2.name || ask_name
```

# Любая операция возвращает результат

---



29.09.2020

- Императивный стиль

```
if found_dog == our_dog
  name = found_dog.name
  message = "We found our dog #{name}!"
else
  message = 'No luck'
end
```

- Функциональный стиль (в реальном коде не рекомендуется)

```
message =
  (if found_dog == my_dog
    name = found_dog.name
    "We found our dog #{name}!"
  else
    'No luck'
  end)
```



29.09.2020

# Сравнения внутри метода

---

- Императивный стиль

```
def get_best_object(obj1, obj2, obj3)
  return obj1 if obj1.price < 20
  return obj2 if obj2.quality > 3
  obj3
end
```

- Функциональный стиль

```
def get_best_object(obj1, obj2, obj3)
  if obj1.price < 20
    obj1
  elsif obj2.quality > 3
    obj2
  else
    obj3
  end
end
```

# Отложенные вычисления

## > Ruby 2.0

---



29.09.2020

- Императивный стиль

```
n, num_elements, sum = 1, 0, 0
while num_elements < 10
  if n**2 % 5 == 0
    sum += n
    num_elements += 1
  end
  n += 1
end
p sum #=> 275
```

- Функциональный стиль

```
p (1..1.0/0).lazy.select { |x| x**2 % 5 == 0 }.take(10).inject(:+) #=> 275
# 1.0/0 == Float::INFINITY
```

- <https://github.com/tokland/tokland/wiki/RubyFunctionalProgramming>



# Отложенные вычисления

## Создание lazy-объекта



29.09.2020

```
class Fib
  include Enumerable

  def each
    a = b = 1
    loop { print "curent: #{a}!t"; yield a;    a, b = b, a + b;    puts 'new val: ' + a.to_s    }
  end
end
```

```
Fib.new.take(5).each { |i| print format "%*: %d, ", i }
```

```
# curent: 1    new val: 1
# curent: 1    new val: 2
# curent: 2    new val: 3
# curent: 3    new val: 5
# curent: 5    *: 1, *: 1, *: 2, *: 3, *: 5,
```

```
Fib.new.lazy.take(5).each { |i| print format "%*: %d, ", i }
```

```
# curent: 1    *: 1, new val: 1
# curent: 1    *: 1, new val: 2
# curent: 2    *: 2, new val: 3
# curent: 3    *: 3, new val: 5
# curent: 5    *: 5,
```

# Отложенные вычисления

## Порядок вычислений



29.09.2020

- Короткая форма

```
Fib.new.lazy.select(&:even?).take(3).each { |i| print format " res: %d, ", i }
```

- Разложение по объектам

```
fib_lazy  = Fib.new.lazy
even_fib  = fib_lazy.select { |i| print " -sel #{i} #{i.even?}- "; i.even? }
res       = even_fib.take(3)
           res.each { |i| print format " res: %d, ", i }
```

```
# curent: 1 -sel 1 false- new val: 1
# curent: 1 -sel 1 false- new val: 2
# curent: 2 -sel 2 true-  res: 2, new val: 3
# curent: 3 -sel 3 false- new val: 5
# curent: 5 -sel 5 false- new val: 8
# curent: 8 -sel 8 true-  res: 8, new val: 13
```

Fib.new -> объект класса Fib  
.lazy -> объект Lazy Enumerator  
.select(&:even?) -> объект Lazy Enumerator  
.take(3).

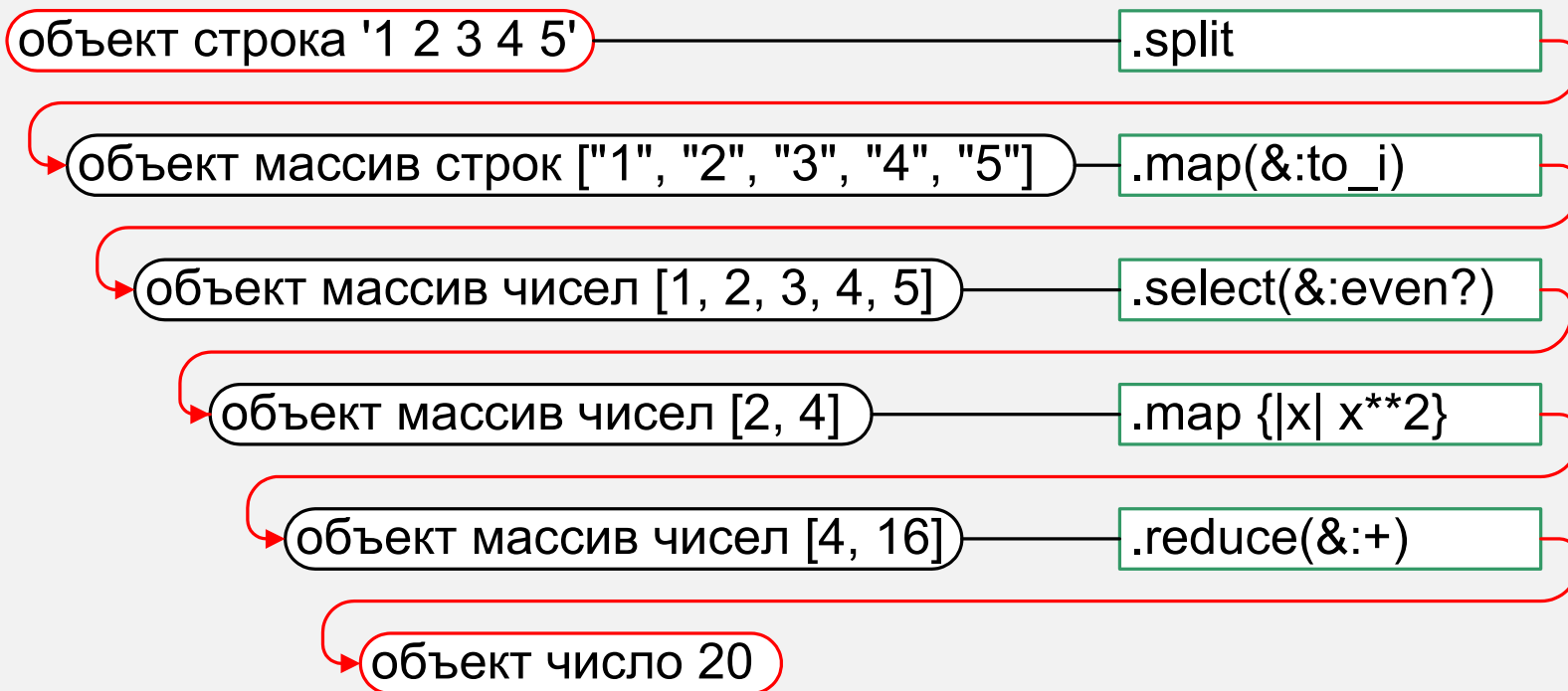
# Краткий итог

## Цепочка вызовов



29.09.2020

```
'1 2 3 4 5'.split.map(&:to_i).select(&:even?).map {|x| x**2}.reduce(&:+)
```



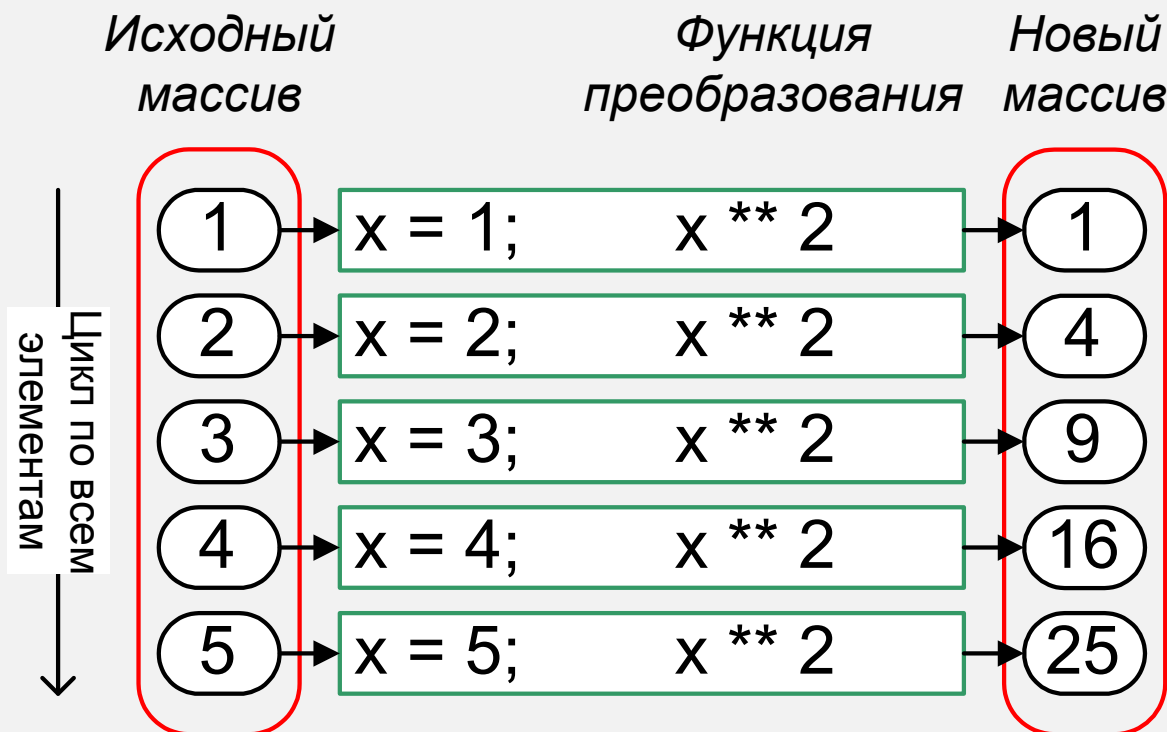
- Варианты передачи функции обработки в `map`, `select`, `reduce`, ...:  
`&:even?`      `&:"even?".to_proc`      `&method(:even?)`

# Краткий итог map / collect



29.09.2020

`[1, 2, 3, 4, 5].map { |x| x ** 2 }`  
# => **[1, 4, 9, 16, 25]**

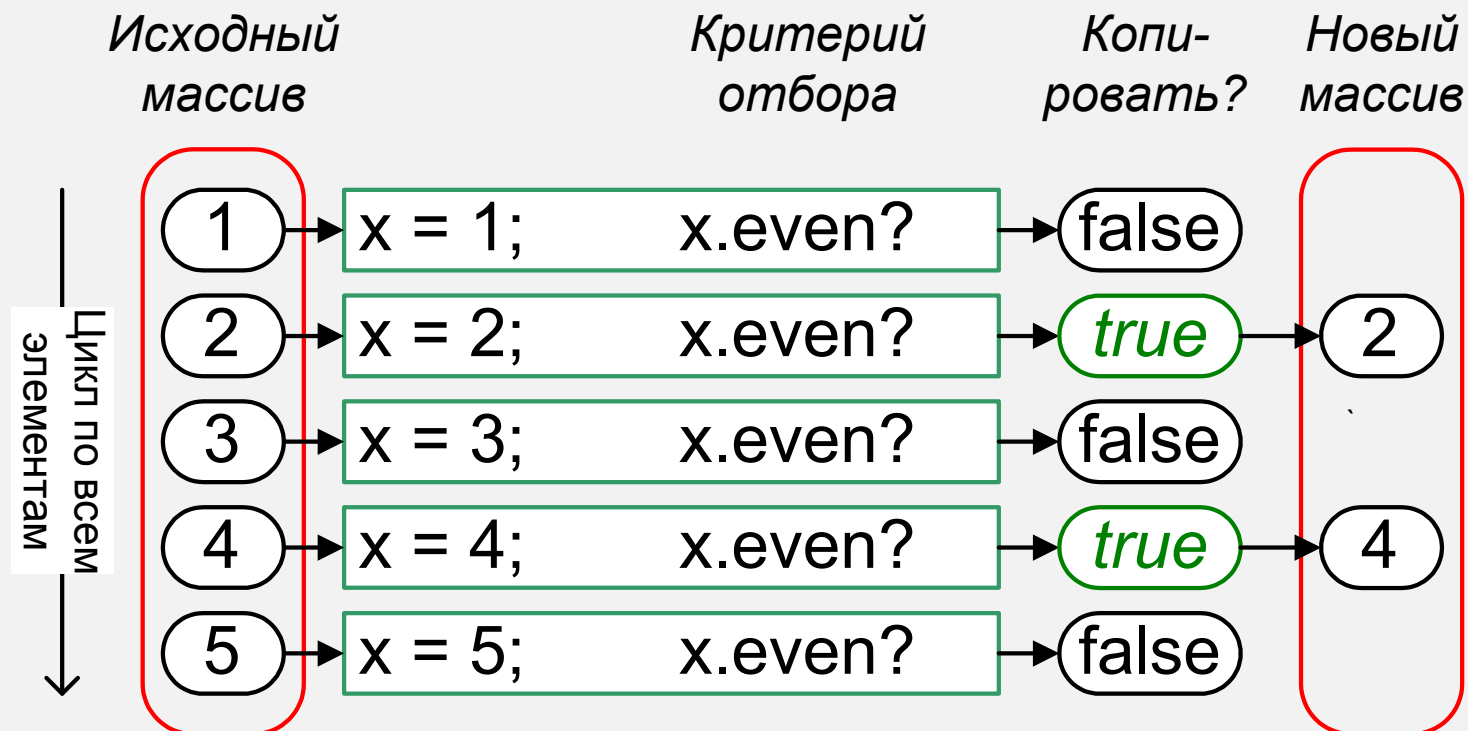


# Краткий итог select



29.09.2020

`[1, 2, 3, 4, 5].select { |x| x.even? }`  
# => **[2, 4]**



# Краткий итог inject / reduce



29.09.2020

`[1, 2, 3, 4, 5].inject(0) { |acc, x| acc + x }`  
# => **15**



# Полезные методы

## #tap

---



29.09.2020

- #tap — модифицировать себя

**object.tap {...} → object**

```
('a'..'z').each_with_index.  
  select { |_, i| i.odd? }.  
  map { |sym, _| sym.upcase }.  
  tap { |list| [2,3,5,7].each { |i| list.delete_at i } }
```

```
# ["B", "D", "H", "L", "N", "R", "T", "X", "Z"]
```

# Полезные методы

## #yield\_self in Ruby 2.5

---



29.09.2020

- #yield\_self — преобразовать объект

**object.yield\_self { func(object) } → new\_object**

['https://api.github.com/repos/rails/rails'](https://api.github.com/repos/rails/rails)

```
.yield_self { |x| URI.parse(x) }  
.yield_self { |x| Net::HTTP.get(x) }  
.yield_self { |x| JSON.parse(x) }  
.yield_self { |x| x.fetch('stargazers_count') }  
.yield_self { |x| "Rails has #{x} stargazers" }  
.yield_self { |x| puts x }
```

- <http://mlomnicki.com/yield-self-in-ruby-25/>



# dry-rb

## next-generation Ruby libraries collection

---



29.09.2020

- <http://dry-rb.org/>
- **dry-validation** — логика предикатов для проверки любых значений

```
schema = Dry::Validation.Form do
  required(:name).filled
  required(:age).filled(:int?, gt?: 18)
end
```

```
schema("name" => "Jane", "age" => "30").to_h
# => {name: "Jane", age: 30}
```

```
schema("name" => "Jane", "age" => "17").messages
# => {:age=>["must be greater than 18"]}
```

# dry-rb, dry-types



29.09.2020

- Конструирование своих типов данных с заданными ограничениями

```
module Types
```

```
  include Dry::Types.module
```

```
  Greeting = Strict::String.enum("Hello", "Hola", "Hallo")
end
```

```
Types::Greeting["Hello"]
```

```
# => "Hello"
```

```
Types::Greeting["Goodbye"]
```

```
# Dry::Types::ConstraintError: "Goodbye" violates constraints...
```

- Добавление проверки типов атрибутов

```
class Person < Dry::Struct
```

```
  attribute :name, Types::Strict::String
```

```
  attribute :age, Types::Strict::Int
```

```
end
```

```
Person.new(name: "Jane", age: 30)
```

```
# => #<Person name="Jane" age=30>
```

# Dry-rb, Monads & pattern matching



29.09.2020

- Безопасная навигация

```
Dry::Monads::Maybe(user)  
  .fmap(&:address)  
  .fmap(&:street)  
  
# If user with address exists  
# => Some("Street Address")  
# If user or address is nil  
# => None()
```

```
# Ruby 2.3 имеет  
# Safe Navigation Operator  
# (&.)
```

```
# user&.address&.street
```

- Шаблоны

```
class CreateArticle  
  include Dry::Monads::Result::Mixin  
  include Dry::Matcher.for(:call, with: Dry::Matcher::ResultMatcher)  
  
  def call(input)  
    if success?(input)  
      output = some_action(input)  
      Success(output)  
    else  
      Failure(input)  
    end  
  end  
end
```

```
create = CreateArticle.new  
create.(input) do |m|  
  m.success do |output|  
    # Handle success  
  end  
  
  m.failure do |err|  
    # Handle failure  
  end  
end
```

# Dry-monads

## Do notation



29.09.2020

```
require 'dry/monads'
```

```
class CreateAccount
  include Dry::Monads[:result]

  def call(params)
    validate(params).bind { |values|
      create_account(values[:account]).bind { |account|
        create_owner(account, values[:owner]).fmap { |owner|
          [account, owner]
        }
      }
    }
  end

  def validate(params)
    # returns Success(values) or Failure(:invalid_data)
  end

  def create_account(account_values)
    # returns Success(account) or Failure(:account_not_created)
  end

  def create_owner(account, owner_values)
    # returns Success(owner) or Failure(:owner_not_created)
  end
end
```



```
class CreateAccount
  include Dry::Monads[:result]
  include Dry::Monads::Do.for(:call)

  def call(params)
    values = yield validate(params)
    account = yield create_account(values[:account])
    owner = yield create_owner(account, values[:owner])

    Success([account, owner])
  end

  ....
end
```

# Примеры решения задач



29.09.2020

- Скопировать все отрицательные элементы и перемножить

- Не оптимально

```
x = Array.new(20) { rand(-10..10) }
```

```
c = []
```

```
x.each { |el| c << el if el < 0 }
```

```
a = c.inject(1) { |a, el| a * el }
```

```
p x, c, a
```

- Лучше

```
x = Array.new(20) { rand(-10..10) }
```

```
c = x.select { |el| el < 0 }
```

```
a = c.inject(1, :*)
```

```
p x, c, a
```

# Примеры решения задач

---



29.09.2020

- Получить частное от деления max/min

```
p list=Array.new(20) { rand(200) - 100 }  
min, max = list.minmax  
quotient = (min != 0 ? max.to_f / min : 0)  
  
puts max, min, quotient
```

# Примеры решения задач



29.09.2020

- Поменять местами минимальный и максимальный элементы

```
p b=Array.new(40) { rand(100) }
```

```
p min = b.each_with_index.min
```

```
p max = b.each_with_index.max
```

```
b[max[1]], b[min[1]] = b[min[1]], b[max[1]]
```

```
p b
```

# Примеры решения задач

---



29.09.2020

- Максимальный элемент поместить на место последнего отрицательного

```
p a=Array.new(20) { rand(50) - 10 }
```

```
idx = a.rindex { |x| x < 0 }
```

```
if idx # reverse index
```

```
  a[i] = a.max
```

```
end
```

```
# a[i] = a.max if idx
```

```
p a
```





29.09.2020

# Примеры решения задач

---

- **Найти** гласные, которые встречаются **во всех** словах

`words = gets.split` # прочитать из консоли и разбить по словам

```
result = ['a', 'e', 'i', 'o', 'u', 'y'].find_all do |c|  
  words.all? { |word| word.include? c }  
end
```

`p result`

# Полезные особенности Ruby

---



29.09.2020

- Динамическое объявление методов
- Расширенный синтаксис без скобок
- Большое количество встроенных функций
- Наличие большого количества сторонних библиотек

# Генерация методов с помощью define\_method

---



29.09.2020

```
class String
  [:red, :green, :blue].each do |m|
    define_method(m) {
      "<span style='color: #{m}'>#{self}</span>"
    }
  end
end

puts "test".green, "test".red, "test".blue
# <span style='color: green'>test</span>
# <span style='color: red'>test</span>
# <span style='color: blue'>test</span>
```

# Пример кода с неизвестными методами

---



29.09.2020

```
result = html do
  head { title { "Test page" } } +
  body do
    div { "Uncolored string" + br + "test string".green }
  end
end
```

puts result

- Результат в консоли

```
<html><head><title>Test page</title>
</head>
<body><div>Uncolored string<br/><span style='color: green'>test
  string</span></div>
</body>
</html>
```

# Перехват обращения к не существующему методу



29.09.2020

# конструируем Hash в форме symbol => lambda

```
Elements = {  
  html: ->(var) { "<html>#{var}</html>\n" },  
  head: ->(var) { "<head>#{var}</head>\n" },  
  title: ->(var) { "<title>#{var}</title>\n" },  
  body: ->(var) { "<body>#{var}</body>\n" },  
  div: ->(var) { "<div>#{var}</div>\n" },  
  br: -> { "<br/>" }  
}
```

# перехватываем любое упоминание неизвестного метода

```
def method_missing(meth, *args, &block)  
  if (func = Elements[meth])  
    block ? func.call( block.call ) : func.call  
  else  
    # если не знаем, что это, вызываем обработчик предка  
    super.method_missing(meth, *args, &block)  
  end  
end
```

# Domain Specific Language (DSL)

---



29.09.2020

- DSL - Язык, ориентированный на предметную область

Пример — опросник (quiz)

- Интерфейс пользователя:

Who was the first president of the USA?

1 - Fred Flintstone

2 - George Washington

Enter your answer:

- Текст программы тестирования:

question 'Who was the first president of the USA?'

wrong 'Fred Flintstone'

right 'George Washington'

- [http://jroller.com/rolsen/entry/building\\_a\\_dsl\\_in\\_ruby](http://jroller.com/rolsen/entry/building_a_dsl_in_ruby)



```
def question(text)
  puts "Just read a question: #{text}"
end
def right(text)
  puts "Just read a correct answer: #{text}"
end
def wrong(text)
  puts "Just read an incorrect answer: #{text}"
end
load 'questions.qm'
```

# DSL и сущность «блок»



29.09.2020

- Фрагмент описания теста на DSL RSpec с использованием блоков Ruby

```
describe "launch the rocket" do
  context "all ready" do ...
  end

  context "not ready" do ...
  end
end
```
- Описание теста с явной передачей ссылок на функции

```
def context_block1; ...; end

def context_block2; ...; end

def describe_block1
  context "all ready", &context_block1
  context "not ready", &context_block2
end

describe "launch the rocket", &describe_block1
```



# Области применения Ruby

---



29.09.2020

- Администрирование ОС
- Веб-приложения
- Вспомогательные GUI-приложения
- Автоматизация тестирования
- Обучение



- Специальный заголовок для автоматического запуска из консоли (Linux, MacOS, etc)

`#!/usr/bin/env ruby`

- Запуск внешних процессов из Ruby-кода:

1. `result = `ls -l``

2. `f = open("ls -l")  
result = f.read()`

3. `result = IO.popen(["ls", "-l"]).read`

# Скрипты администрирования



29.09.2020

- Добавить в SVN все не добавленные файлы

```
#!/usr/bin/env ruby
```

```
STATUS_FILE = "status#{Time.now().to_i}.txt"
```

```
system "svn status > #{STATUS_FILE}"
```

```
File.open(STATUS_FILE) do |file|
```

```
  while (line = file.gets) do
```

```
    if (line[0] == '?' &&
```

```
      !(line.include?( STATUS_FILE )) &&
```

```
      !(line.include?( __FILE__ )) ) then
```

```
        system "svn add #{line[3..-1]}"
```

```
      end
```

```
    end
```

```
  end
```

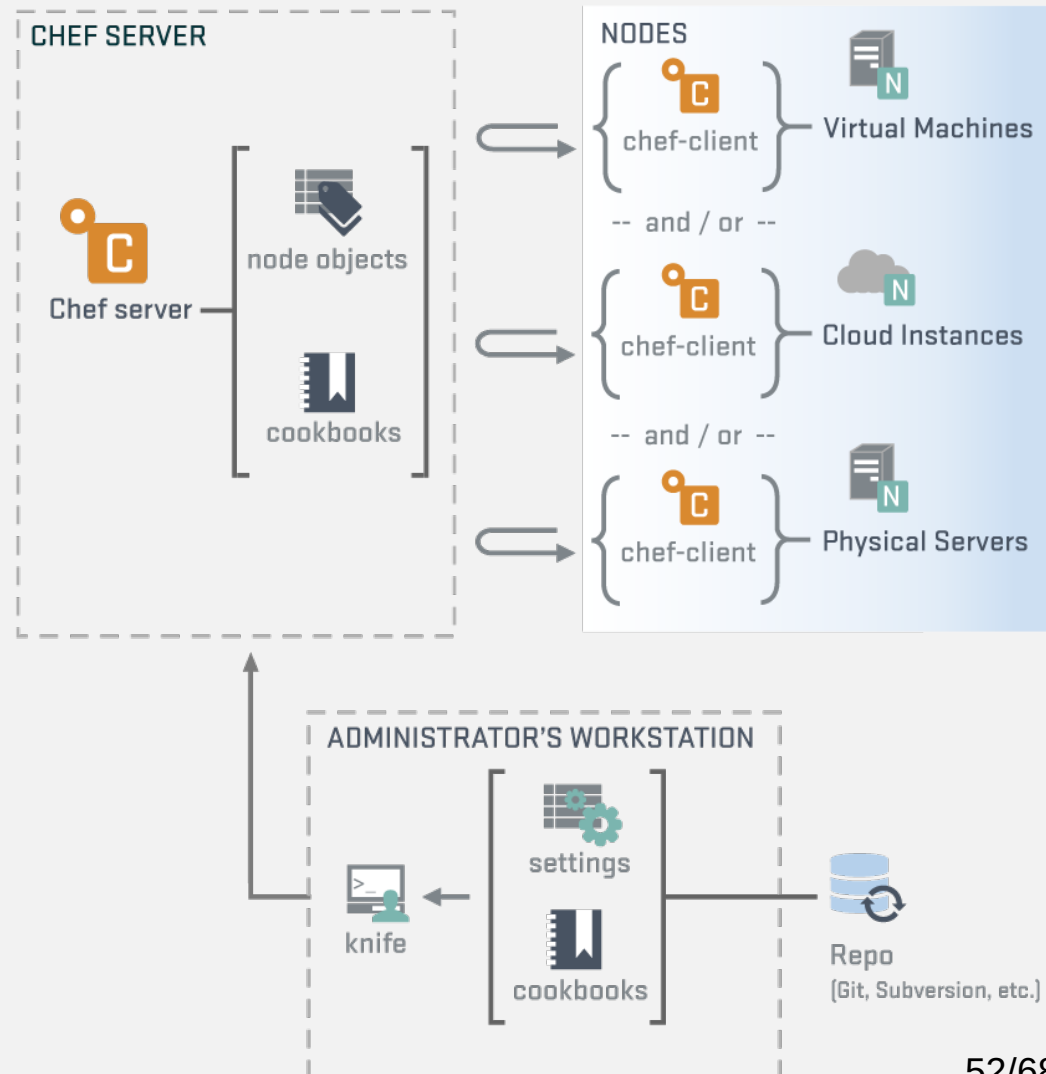
```
File.delete STATUS_FILE
```

# Управление развертыванием приложений



29.09.2020

- Chef  
<http://www.opscode.com/chef>
- Puppet  
<https://puppet.com/>
- ...



# Пример «рецептов» Chef



29.09.2020

```
# web servers for both HTTP and HTTPS
name "webserver"
description "Systems that serve HTTP and HTTPS"
run_list(
  "recipe[apache2]",
  "recipe[apache2::mod_ssl]"
)
default_attributes(
  "apache" => {
    "listen_ports" => ["80", "443"]
  }
)

# create a postgresql database
postgresql_database 'mr_softie' do
  connection ({
    host: '127.0.0.1', port: 5432,
    username: 'postgres',
    password: node['postgresql']['password']['postgres']})
  action :create
end
```

# Мониторинг сервисов в распределенной системе

---



29.09.2020

- God - <http://godrb.com/>

```
God.watch do |w|  
  w.name = "simple"  
  w.start = "ruby /full/path/to/simple.rb"  
  w.keepalive(  
    :memory_max => 150.megabytes,  
    :cpu_max => 50.percent)  
end
```

# Создание «настольных» приложений



29.09.2020

- Qt-приложение

```
#!/usr/bin/env ruby
```

```
$VERBOSE = true;
```

```
$.unshift File.dirname($0)
```

```
require 'Qt'
```

```
a = Qt::Application.new(ARGV)
```

```
hello = Qt::PushButton.new('Hello World!', nil)
```

```
hello.resize(100, 30)
```

```
hello.show()
```

```
a.exec()
```

# Тестирование приложений

---



29.09.2020

- MiniTest::Unit
- RSpec, Cucumber, ...
- Selenium Web-driver  
DSL Capybara
- Автоматизация тестирования GUI -  
<http://www.sikulix.com/>
- Автоматизация тестирования мобильных приложений  
Appium, Calabash, ...



# Пример RSpec



29.09.2020

```
# bowling_spec.rb
```

```
require 'bowling'
```

```
RSpec.describe Game do
```

```
  describe "#score" do
```

```
    it "returns 0 for an all gutter game" do
```

```
      game = Game.new
```

```
      20.times { game.roll(0) }
```

```
      expect(game.score).to eq(0)
```

```
    end
```

```
  end
```

```
end
```

```
# bowling.rb
```

```
class Game
```

```
  def roll(pins)
```

```
  end
```

```
  def score
```

```
    0
```

```
  end
```

```
end
```

# Katalon Recorder Selenium IDE



29.09.2020

Расширение  
для Firefox и  
Chrome,  
записывает  
действия  
пользователя

The screenshot shows the Selenium IDE 1.9.0 interface. The main window displays a test case titled 'Untitled \*' with a table of commands and their targets/values. The commands are: 'open' with target '/', 'type' with target 'id=v1' and value '10', 'type' with target 'id=v2' and value '3', and 'click' with target 'xpath=()/input[@id='o...'. The interface also includes a 'Log' tab at the bottom and a 'Reference' tab.

Command	Target	Value
open	/	
type	id=v1	10
type	id=v2	3
click	xpath=()/input[@id='o...	

# Selenium + Ruby



29.09.2020

- `gem install selenium-webdriver`
- Пример подключения к сайту <http://google.com>

```
require 'selenium-webdriver'
```

```
driver = Selenium::WebDriver.for :firefox  
driver.get "http://google.com"
```

```
element = driver.find_element :name => "q"  
element.send_keys "Cheese!"  
element.submit  
puts "Page title is #{driver.title}"  
wait = Selenium::WebDriver::Wait.new(:timeout => 10)  
wait.until { driver.title.downcase.start_with? "cheese!" }  
puts "Page title is #{driver.title}"  
driver.quit
```

# Capbara и RSpec



29.09.2020

```
describe "the signup process", :type => :request do
  before :each do
    User.make(:email => 'user@example.com',
              :password => 'caplin')
  end

  it "signs me in" do
    within "#session" do
      fill_in 'Login', :with => 'user@example.com'
      fill_in 'Password', :with => 'password'
    end
    click_link 'Sign in'
  end
end
```



- Возможность вызова любых Java-классов
- Возможность запуска в окружении Java-машины
- Графическое приложение на Java

```
require 'java'
```

```
frame = javax.swing.JFrame.new  
frame.getContentPane.add javax.swing.JLabel.new('Hello, World!')  
frame.setDefaultCloseOperation  
  javax.swing.JFrame::EXIT_ON_CLOSE  
frame.pack  
frame.set_visible true
```

- <http://jruby.org/>

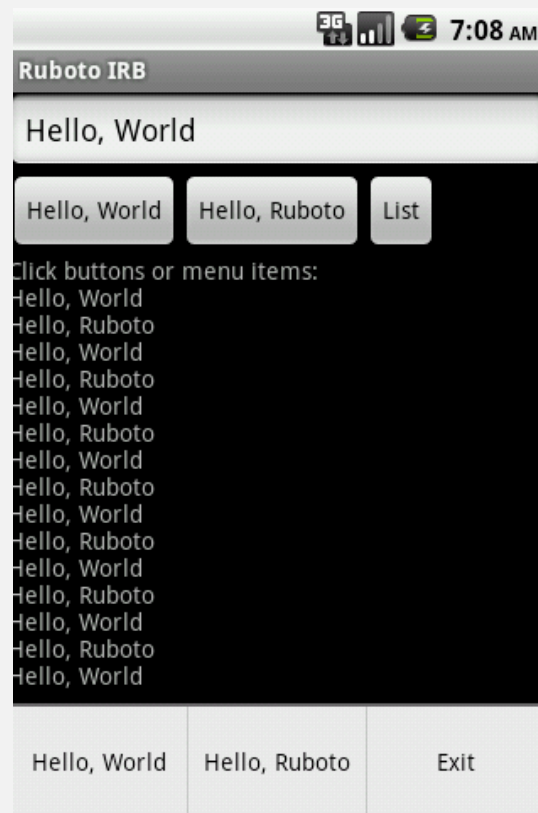
<http://en.wikipedia.org/wiki/JRuby>

# Программы для Android



29.09.2020

- jRuby + <http://ruboto.org/>

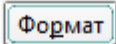
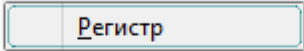
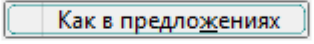
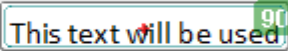
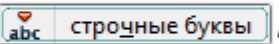
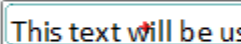
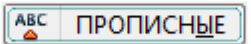
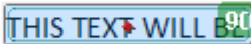
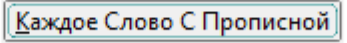
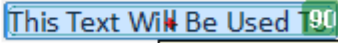
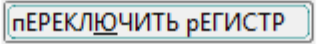



# Sikulix

## Визуальное тестирование



29.09.2020

```
119 describe "Format menu", :open => :yes, :text_select => :yes do
120   include_context "open document"
121   include_context "text selection"
122   include_context "clickable container", 
123
124   describe "case selector submenu" do
125     include_context "self clean"
126     include_context "hoverable menu", 
127     describe "sentence case" do
128       it_behaves_like "clickable", , 
129     end
130     describe "lower case" do
131       it_behaves_like "clickable and hides something", , 
132     end
133     describe "upper case" do
134       it_behaves_like "clickable", , 
135     end
136     describe "camel case" do
137       it_behaves_like "clickable", , 
138     end
139     describe "switch case" do
140       it_behaves_like "clickable", , 
141     end
142   end
end
```

# KNIME

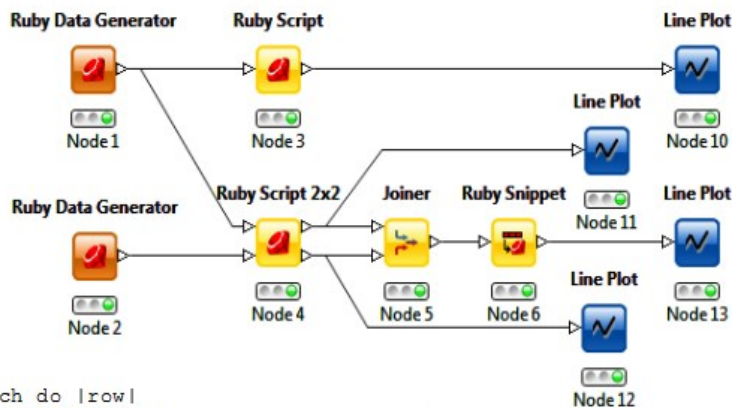
## Бизнес-аналитика



29.09.2020

```
1000.times do |i|
  x = i * 0.1 / Math::PI
  $outContainer << Cells.new.double(x).double(Math.sin(x)).double(Math.sin(x + Math::PI/3))
end
```

```
$outContainer.rowKey = 100000
1000.times do |i|
  x = i * 0.1 / Math::PI
  $outContainer << Cells.new.double(x).double(Math.cos(x)).double(Math.cos(0.3*x))
end
```



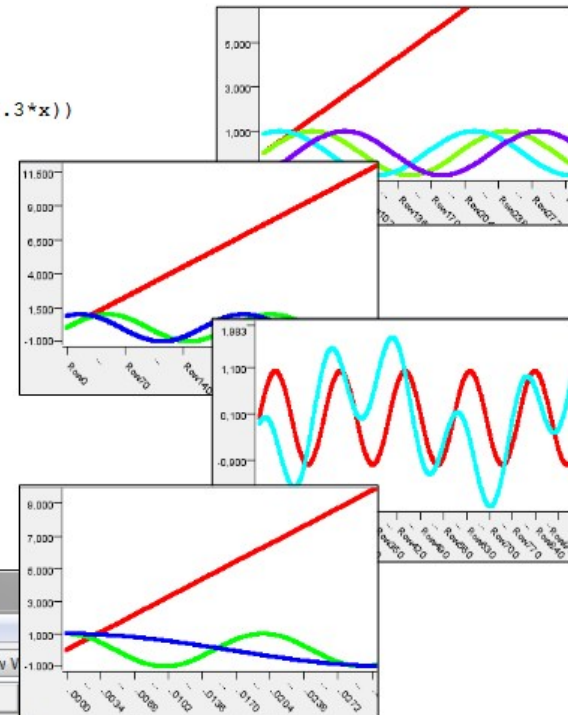
```
$inData0.each do |row|
  $outContainer0 << row
end
```

```
$inData1.each do |row|
  $outContainer1 << row
end
```

```
Cells.new.
  double(row[1].to_f).
  double(row[2].to_f - row[4].to_f)
```

Joined table - 2:15 - Joiner(Node 5)

File	Table "default" - Rows: 1000	Spec - Columns: 5	Properties	Flow V
Row ID	D x	D y1	D y2	
Row0_Row100000	0	0	0.866	1
Row1_Row100001	0.032	0.032	0.881	0.999
Row2_Row100002	0.064	0.064	0.896	0.998
Row3_Row100003	0.095	0.095	0.91	0.995
Row4_Row100004	0.127	0.127	0.923	0.992
Row5_Row100005	0.159	0.158	0.934	0.987
Row6_Row100006	0.191	0.19	0.945	0.982
Row7_Row100007	0.223	0.221	0.955	0.975
Row8_Row100008	0.255	0.252	0.964	0.968
Row9_Row100009	0.286	0.283	0.972	0.960





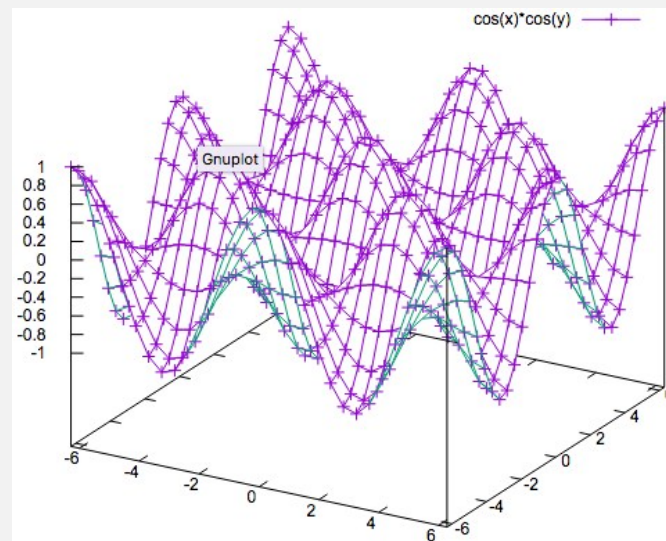


- <http://sciruby.com/>
- Jupyter notebook - <http://jupyter.org/> + <https://github.com/SciRuby/iruby>
- <https://github.com/SciRuby/sciruby-notebooks>

```
include Math
```

```
double_pi = PI * 2
```

```
plot3d = Splot.new(  
  'cos(x)*cos(y)',  
  xrange: -double_pi..double_pi,  
  yrange: -double_pi..double_pi,  
  style: 'function linespoints',  
  hidden3d: true,  
  isosample: 30  
)
```



# Прототипирование протоколов взаимодействия

---



29.09.2020

- Взаимодействие с USB-устройством через интерфейс библиотеки libusb

```
require "libusb"
```

```
usb = LIBUSB::Context.new
```

```
device = usb.devices(:idVendor => 0x04b4, :idProduct => 0x8613).first
```

```
device.open_interface(0) do |handle|
```

```
  handle.control_transfer( :bmRequestType => 0x40,
```

```
                           :bRequest => 0xa0,
```

```
                           :wValue => 0xe600,
```

```
                           :wIndex => 0x0000,
```

```
                           :dataOut => 1.chr)
```

```
end
```

# Ruby-подобные языки программирования

---



29.09.2020

- Crystal, statically type-checked, Ruby similar  
<https://crystal-lang.org>
- Mirah, JVM based, Ruby with static types  
<https://github.com/mirah/mirah>
- Groovy, JVM based  
<http://www.groovy-lang.org>
- Elixir, Erlang based  
<http://elixir-lang.org>

# Полезные ссылки



29.09.2020

- <http://ru.wikibooks.org/wiki/Ruby> - учебник на русском языке.
- <http://www.ruby-lang.org> - основной сайт Ruby.
- <http://www.ruby-doc.org> - официальная документация Ruby на английском языке.
- <http://rubymonk.com>, <http://tryruby.com>, <https://www.codecademy.com/ru/tracks/ruby> — сборники интерактивных учебников с возможностью написать мини-программу и проверить её работу.