Конспект лекций курса по Ruby/Sinatra/Ruby on Rails

Конспект составил: Алексей Цаплин-Купайсинов (Ruby-программист) - https://github.com/krdprog

Курс: http://rubyschool.us/

Конспект RubySchool.us [1]

Урок 1

Приветствую! В этом конспекте я фиксирую важные мысли и моменты из видеоуроков rubyschool.us, конспекты первых уроков не содержат много информации, но не переживайте, чем дальше по урокам, тем ценнее материал.

Для более интенсивного освоения рекомендую вам использовать GNU/Linux сразу.

У кого трудности с настройкой рабочего Ruby-окружения на компьютере, чтобы не тормозить изучение, используйте первое время онлайн-сервис: https://repl.it/languages/ruby

Удачного освоения!

git, sql

книга: мифический человекомесяц.

резюме - это рекламный проспект + рекомендации

$30/hour

```
puts
```

Урок 2 Тема урока: переменные

Комментарии в ruby - #

tax = 0.40

если идёт дублирование данных, то лучше вынести в переменную.

переменные задаются до вывода на экран

gets - get string

name = gets.chomp

print - оператор не переводит строку после своей работы, в отличие от puts

Спец.символы:

\n - перевод строки

{foo} - вызов переменной внутри кавычек

Имея требования можно написать программу

.chomp - чтобы чомпнуть перевод строки

функция = метод = процедура

объект = экземпляр класса

Урок 3

```
mkdir -p dir/subdir/subdir2/foo/bar
```

резервное копирование проектов рекомендует в каталог bak/1 ... n, но используй лучше git

```
to_s
to_i
1.class
"foobar".class
1.23.class
```

выдаст тип данных

В Ruby всё - объект

```
string - "2"
fixnum - 2
float - 2.1
array - [...]
hash - {...}
bignum - 2^30+1
```

- https://ru.wikibooks.org/wiki/Ruby
- https://ru.wikibooks.org/wiki/Ruby/Базовые*типы*данных

Преобразование типов: .to_i - строки в числа .to_f - в число с плавающей точкой .to_s - в строку

Урок 4

```
x.chomp!
```

Методы объекта (список):

```
a = "foo bar"
a.methods

a.empty? #=> false
.odd? # нечетные
.even? # чётные
```

.times

```
100.times { |i| puts "i = #{i}" }
```

Homework:

Class: Integer - https://ruby-doc.org/core-2.5.3/Integer.html

Class: String - https://ruby-doc.org/core-2.5.3/String.html

Урок 5

```
5.upto(10) { |x| print x, " " } #=> 5 6 7 8 9 10
50.downto(-50) { |x| print x, " " }
```

i - писать, когда индекс (принимает значение от нуля)

```
.strip # удаляет не только пробелы, но и
# табуляцию и перенос строки вместо чомп
```

```
\n - перенос строки (line feed - lf)
\t - табуляция
\r - возврат на начало строки (carriage return - cr)

cr lf = \r\n
```

вывести обратный слеш:

```
puts "\\" #=> \
```

Урок 6

Блоки:

Стиль принятый для однострочника:

```
10.times { puts "Foo!" }
```

Другой синтаксис с несколькими командами:

```
10.times do
  puts "Foo!"
  puts "Bar!"
end
```

```
10.times { |i| puts i }
```

```
10.times do |i|
  puts "Foo!"
end
```

Счёт не с нуля:

```
# от 10 до 20
10.upto(20) { |i| puts i }

# от 20 до 10
20.downto(10) { |i| puts i }

# OR:
10.upto(20) do |i|
  p i
end

20.downto(10) do |i|
  p i
end
```

Решение задания:

```
# программа делает от 10 до 20 запусков и выводит Foo это количество раз
10.upto(20) do |x|
  1.upto(x) { |i| p "#{i} Foo!" }
end
```

Rand Charset:

```
30.times { print rand(70..120).chr }
```

Homework:

```
arr = %w(t e r m i n a t o r)
dash = "-"

arr.size.times do |i|
  print arr[i]
  sleep 0.6
  print dash
  sleep 0.6
end
```

Урок 7

Выход из программы:

```
a = true
while a == true
  print "Выйти из программы? (Y/N): "
  answer = gets.strip.capitalize

  if answer == "Y"
    a = false
  end

end
```

```
== - равно
!= - не равно
<>
>=
<=
```

Прерывание программы:

```
exit
```

if И, ИЛИ:

```
# И
if a > 1 && b < 3
  puts "Foo!"
end

# ИЛИ
if a > 1 || b < 3
  puts "Foo!"
end
```

Урок 8

Ruby понимает 10_000_000 как 10 млн.

```
a = 10_000
```

Обрывает работу оператора:

```
break
```

```
"Hello".reverse
```

Ranges:

```
(1..10).each { |x| puts x*4 }
```

```
(1..5).each => 1 2 3 4 5
(1...5).each => 1 2 3 4

# можно не только цифры:
('aa'..'ff').each { |x| puts x }
('10aa'..'20ff').each { |x| puts x }
```

GUI framework: Shoes! The easiest little GUI toolkit, for Ruby.

Урок 9

Методы

```
def foo

end
```

Методу можно передать какой-либо параметр, и он вернёт какое-либо значение.

```
def gets_password
  print "Введите пароль: "
  return gets.chomp
end

xx = gets_password

puts "Был введён пароль #{xx}"
```

return - можно не писать

По умолчанию руби возвращает результат последнего выражения.

```
def print_name name
  puts "Hello #{name}"
end

print_name "Alex"
```

Внутри метода все переменые определяются заново (локальные переменные метода)

```
a = 1
def foo
  a = 2
end

puts a # => 1
```

@a - переменная экземпляра

Объекты, экземпляры класса (instance)

"foo" "foo" находятся в разных участках памяти, это разные объекты. Чтобы закрепить за одним участком используются *символы*.

Символы (symbol):

Обозначается как:

```
:foo
```

Можно переменным назначать значение символов.

true or false:

```
puts "aaa" == "aaa"
puts :aaa == :aaa
puts "aaa".equal? "aaa" # false
puts :aaa.equal? :aaa
```

.equal? - сравнивает объекты по ссылке

Проверка id объекта:

```
"aaa".object_id # каждый раз меняется
:aaa.object_id # у символа один и тот же
```

Массивы (arrays)

Class: Array (Ruby 2.5.1)

Массив = набор объектов

```
arr = []

arr2 = %w{foo bar zoo moo faf}

arr << 14
arr << 22
```

Урок 10

```
def movie
  action = [:left, :right, :up, :down]

  x = rand(0..3)

  return action[x]
end

def say
  sleep 0.06
end

while true
  speech = movie
  puts "Go to #{speech}"
  sleep 0.4
end
```

capitalize every name:

```
arr = %w{joe matt mary olga oleg peter vasya}

arr.each do |x|
  puts x.capitalize
end
```

индекс arr[2]

```
arr = [1, 2, 3, 4, 5]
arr2 = arr[1..3]

puts arr2 #=> 2 3 4
```

```
arr = %w{aa bb cc dd ee}
arr[1..3][2]
#=> "dd"
```

Удаление элемента из массива

```
arr = ["aa", "bb", "cc", "dd"]

arr.delete_at 1 # delete "bb"
arr.delete "dd" # delete "dd"
```

Удаление из массива (задача):

```
person = %w{vasiliy mariya joe bob marley}

while true

  person.size.times do |x|
    puts "#{x}. #{person[x].capitalize}"
  end

  puts ""
  print "Кого удалить? (порядковый номер): "
  del = gets.to_i

  person.delete_at del

  puts "====="

end
```

Или вариант с each

```
person = %w{vasiliy mariya joe bob marley}

while true
  i = 0
  person.each do |x|
    puts "#{i}. #{x.capitalize}"
    i += 1
  end

  puts ""
  print "Кого удалить? (порядковый номер): "
  del = gets.to_i

  person.delete_at del

  puts "====="

end
```

Добавление в массив с выводом на экран:

```
arr = []

while true
  print "Enter name to add: "
  name = gets.strip.capitalize

  if name == ""
    break
  end

  arr << name

  puts "==="

  x = 0

  arr.each do |name|
    puts "#{x}. #{name}"
    x += 1
  end

end
```

Массивы можно вкладывать в массив.

Вывод многомерного массива

```
arr = [[1, 2, 3], [4, 5, 6]]

puts arr[0][1] #=> 2
```

Задание:

```
# enter yor name
# enter your age
# in arr
# print all data

persons = []

while true
  person = []

  print "Enter your name: "
  name = gets.strip.capitalize

  if name == ""
    break
  end

  person << name

  print "Enter your age: "
  age = gets.to_i

  person << age

  persons << person

end

puts "", "===="
puts "RESULT:"

persons.each do |x|
  puts "#{x[0]}, #{x[1]}"
end

puts "====", ""
```

Задание: камень, ножницы, бумага:

```
while true

  puts "", "========================"
  print "(R)ock, (S)cissors, (P)aper? "
  s = gets.strip.capitalize

  if s == "R"
    @user_choice = :rock
  elsif s == "S"
    @user_choice = :scissors
  elsif s == "P"
    @user_choice = :paper
  else
    puts "What? I don`t know."
    exit
  end

  arr = [:rock, :scissors, :paper]

  @computer_choice = arr[rand(0..2)]

  # report about win
  def your_win
    puts "You win! Your choice is #{@user_choice} and computer choice is #{@computer_choice}."
  end

  def computer_win
    puts "Computer win! Your choice is #{@user_choice} and computer choice is #{@computer_choice}."
  end

  # game variants
  if @user_choice == @computer_choice
    puts "Nobody wins. Your choice is #{@user_choice} and computer choice is #{@computer_choice}."
  # user:
  elsif @user_choice == :rock && @computer_choice == :scissors
    your_win
  elsif @user_choice == :scissors && @computer_choice == :paper
    your_win
  elsif @user_choice == :paper && @computer_choice == :rock
    your_win
  # computer
  else
    computer_win
  end

end
```

Урок 11

Поменять значения местами

```
 a = 50
b = 20

a, b = b, a

puts a
puts b
```

var 2:

```
a = a + b
b = a - b
a = a - b
```

Обработка двумерного массива:

```
arr = [ [1, 2], [3, 4], [5, 6] ]

arr.each do |item|
  item.each do |x|
    puts x
  end
end
```

.each_with_index - метод заменяющий, что делали выше на прошлых уроках:

```
arr = %w{ Vova Masha Petya Alisha }

arr.each_with_index do |item, i|
  puts "#{i} #{item}"
end
```

Хеши (Hash):

Это структура данных Key value storage - хранилище ключ значение В других языках ещё называют Dictionary

```
hash = Hash.new

hash = {}
```

```
hash = { 'Mike' => '3538300',
        'Jessie' => '4348829'}

puts hash['Mike']
```

> => называется Hash Rocket

Используй одинарные кавычки, а двойные для интерполяции.

```
a = 2+2

puts "#{a}" #=> 4
puts '#{a}' #=> #{a}
```

```
options = {
  :font_size => 10,
  :font_family => 'Arial',
  :arr => [1, 5, 8]
  }

options[:font_size]
options[:font_family]
options[:arr][2]
```

> для исследования используй .inspect и .class

Обращение к хешу по ключу и получаем значение

Добавление элемента в хеш:

```
hash = {}

hash['Alisha'] = 636400
hash[:foo] = 'Bar'
```

Решение задания "Телефонная книга"

```
# Enter name (Enter to stop):
# Enter phone number:
# add to hash

phonebook = {}

while true
  print "Enter name (Enter to stop): "
  name = gets.strip.capitalize

  if name == ""
    break
  end

  print "Enter phone number: "
  number = gets.strip

  phonebook[name] = number
end

puts "", "=== My Phone Book ==="
phonebook.each do |name, phone|
  puts "#{name} number is #{phone}"
end
puts "====================="
```

Важное замечание: в хеше через each элементы могут выводиться не по порядку. Хеш не предназначен для хранения элементов по порядку, в отличие от массива.

Вызвать нужный элемент из хеша:

```
phonebook['Alisha']
```

Решение задачи "Англо-русский переводчик":

```
words = {
  'dog' => 'собака',
  'cat' => 'кошка',
  'frog' => 'лягушка'
  }

while true
  print "Введите слово: "
  user_word = gets.strip.downcase

  if user_word == ""
    break
  end

  puts "Перевод: #{words[user_word]}"
  puts ""
end
```

> У слова может быть несколько переводов, поэтому можно переписать программу, где значение хеша = массив.

```
fruits = []
fruits << {"name"=>"banana", "cost"=>10} << {"name"=>"apple", "cost"=>7}
#=> [{"name"=>"banana", "cost"=>10}, {"name"=>"apple", "cost"=>7}]
```

Урок 12

Перечисление ключей и значений хеша:

```
hash = { right: "Right", left: "Left", top: "Top", down: "Down" }

puts hash.keys
puts hash.values

# ИЛИ
hash.each_key do |key|
  puts key
end

hash.each_value do |value|
  puts value
end
```

hash.keys и hash.values - это массивы, к ним можно применять методы .size и т.п.

Задание: посчитать количество переводов

```
words = {
  'dog' => ['собака', 'шавка', 'бобик'],
  'cat' => ['кошка', 'кошечка'],
  'frog' => ['лягушка', 'квакушка'],
  'mouse' => ['мышь', 'мышка']
}

result = 0

words.each_value { |value| result += value.size }

puts "All words: #{result}"
```

Проверка наличия ключа в хеше:

```
puts words.has_key? 'cat'
# ИЛИ
puts words.include? 'cat'
```

```
if words.has_key? 'cat'
  puts 'Meowww!'
end

# ИЛИ

if words['cat']
  puts 'Meowww!'
end
```

Проверка наличия значения в хеше:

```
puts words.has_value? 'кит'
```

Решение "Однорукий бандит" с хешем:

```
# onehand bandit with hash

win_variant = {
  '111' => 100,
  '222' => 200,
  '333' => 300,
  '444' => 400,
  '555' => 500,
  '666' => 600,
  '777' => 7000,
  '888' => 800,
  '999' => 900,
}

money = 100

while true

  puts 'Press ENTER for game...'
  gets
  random = rand(100..999).to_s


  if win_variant[random]
    puts "Win #{win_variant[random]} dollars."
    money += win_variant[random]
  else
    puts "You lost 10 dollars."
    money -= 10
  end


  puts "Combination: #{random}"
  puts "Your balance is #{money}", ""
end
```

Функции

return

```
def say a
  puts a
end

say "foo"
```

Можно больше параметров задавать, указывать через запятую:

```
def say param1, param2, param3
  # some code
end
```

```
puts "Bar!" if true
puts "Fooo!" if 2+2 = 4
```

Защита от неустановленного параметра:

```
def print_details details
  puts details[:name] if details[:name]
  puts details[:age] if details[:age]
  puts details[:address] if details[:address]
end

hh = { name: "Mike", age: 65, address: "123, West Street"}

print_details hh
```

Урок 13

Ещё вариант добавления в хеш:

```
hh = {}
hh.store('Mike', 65)
```

```
hh = {'mike' => 10, 'alisha' => 42}

hh.keys.each do |key|
  hh[key] = 100
end

puts hh

#=> mike 100, alisha 100
```

Очистить хеш:

```
hh = {11 => 222, 22 => 333}
hh.clear # очистка хеша
```

Удалить элемент из хеша:

```
hh = {'123' => 333, '444' => 7594}
hh.delete '123'
```

Решение задачи:

```
# хеш для хранения данных
@hh = {}

# добавление пары в хеш
def add_person name, age

  # if @hh[name]
  #  puts "Такой пользователь уже существует!"
  # end

  puts "Такой пользователь уже существует!" if @hh[name]

  @hh[name] = age
end

# отображение содержимого хеш
def show_hash
  @hh.each do |name, age|
    puts "#{name} is #{age} years old."
  end
end

while true
  # добавлять пока не введена пустая строка
  print "Enter name: "
  @name = gets.strip.capitalize

  if @name == ""
    show_hash
    exit
  end

  print "Enter age: "
  @age = gets.to_i

  add_person @name, @age
end
```

If в одну строку:

```
a = 10
b = 20

puts 'OK' if a+b == 30
```

Слева можно писать любое выражение:

```
hh['Mike'] = 65 if a+b == 30
```

Перенос параметров метода в хеш:

```
def add_person options
  name = options[:name]
  age = options[:age]
end

options = {:name => 'Mike', :age => 44}
```

```
add_person :name => "Mike", :age => 22
```

> Хеш в качестве параметра метода можно использовать только в конце перечисленных параметров

```
def say_hello param1, param2, hash_param
  # code
end
```

Merge (hash):

Объединение хешей

```
# выводит на экран записную книгу
def show_book book
  book.each do |name, age|
    puts "#{name} is #{age} years old"
  end
end

book1 = { 'Mike' => 65, 'Joe' => 12 }
book2 = { "John" => 88, 'Alisha' => 44 }

show_book book1
show_book book2

# or you can merge 2 hashes
book = book1.merge book2
show_book book

#or you can merge 2 hashes with .merge!
book1.merge! book2
show_book book1
```

Перенаправление вывода:

```
ruby app.rb > file.txt
ruby app.rb >> file.txt
```

знак > заменяет файл (ВНИМАНИЕ!) знак >> добавляет в конец файла

> вносим в нашу программу html-теги и затем делаем вывод в файл app.rb > file.html - получился вывод в html

Урок 14

ООП, Класс, Объект (экземпляр класса, instance)

```
class Animal
  def say
    puts "Foooo!"
  end
end

cat = Animal.new
cat.say #=> Fooooo!
```

Переменные внутри класса:

```
class Animal
  def say
    @name = 'Dog'
    puts "I am #{@name}!"
  end
end
```

Переменные класса в отдельный метод initialize (в других языках это называется - конструктор):

```
def initialize
  @hh = {}
  @param = 200
end
```

Задача записная книжка:

```
# класс записной книжки
class Book

  def initialize
    @hh = {}
    @last_person = ''
  end

  def add_person options
    @last_person = options[:name]

    puts "Уже есть!" if @hh[options[:name]]

    @hh[options[:name]] = options[:age]
  end

  def show_all
    @hh.each do |name, age|
      puts "#{name} is #{age} years old."
    end
  end

  def show_last_person
    puts "Last person: #{@last_person}."
  end

end

b = Book.new

b.add_person :name => "Joe", :age => 44
b.add_person :name => "Mark", :age => 14

b.show_all
b.show_last_person
```

```ruby
class Cat
  def initialize
    @foo = 12
  end

  def aaa
    return @foo
  end
end

bar = Cat.new
bar.aaa => 12
```
attr_reader - делает переменную доступную для чтения:
```ruby
def last_person
  @last_person
end

# заменяется на:
attr_reader :last_person
```

attr_accessor - для чтения и записи:

```ruby
attr_accessor :last_person
```

Самолёт:

```ruby
class Airplane

  attr_reader :model
  attr_reader :speed
  attr_reader :altitude

  def initialize(model)
    @model = model
    @speed = 0
    @altitude = 0
  end

  def fly
    @speed = 800
    @altitude = 10000
  end

  def land
    @speed = 0
    @altitude = 0
  end

  def moving?
    return @speed > 0
  end

end

plane1 = Airplane.new('Boeing-777')
plane1.fly
puts "Model: #{plane1.model}, Speed: #{plane1.speed}, Altitude: #{plane1.altitude}"
plane1.land
puts "Model: #{plane1.model}, Speed: #{plane1.speed}, Altitude: #{plane1.altitude}"
puts "Is moving: #{plane1.moving?}"
```

Самолёт версия 2 - запуск рандомных самолётов:

```
# самолёт
class Airplane

  attr_reader :model
  attr_reader :speed
  attr_reader :altitude

  def initialize(model)
    @model = model
    @speed = 0
    @altitude = 0
  end

  def fly
    @speed = 800
    @altitude = 10000
  end

  def land
    @speed = 0
    @altitude = 0
  end

  def moving?
    return @speed > 0
  end

end

models = ['Il-76', 'Boeing-777', 'Airbus-320']

planes = []

20.times do
  model = models[rand(0..2)]
  plane = Airplane.new(model)

  if rand(0..1) == 1
    plane.fly
  end

  planes << plane

end

planes.each do |plane|
  puts "Model: #{plane.model}, Speed: #{plane.speed}, Altitude: #{plane.altitude}"
  puts "Plane moving: #{plane.moving?}"
end
```

Продолжение конспекта: Урок 15-19 - https://github.com/krdprog/rubyschool-notes/blob/master/rubyschool-notes-02.md

Содержание конспекта: