Ruby для Java программистов

Содержание курса:

- 3 основных столпа Ruby
- Все является объектом, а каждая операция - это вызов метода.
- ООП в Ruby
- Рефлексия и метапрограммирование
- Функциональные идиомы и итераторы
- "Утиная типизация" и примеси (mix-ins)
- Блоки и yield

Ruby

- интепритируется
- объектно-ориентирован
- все является объектом
- каждая операция - это вызов метода для какого-то объекта
- Динамическая типизация: объекты имеют типы, но переменные нет
- динамический язык
- добавлять, менять код во время выполнения (метапрограммирование)
- спрашивать объекты о них ("рефлексия")
- итог: все программирование - метапрограммирование

Соглашения о названиях

- ClassNames use UpperCamelCase

```
class FriendFinder ... end
```

- methods & variables use snake_case

```
def learn_conventions ... end def faculty_member? ... end def
charge_credit_card! ... end
```

- CONSTANTS (scoped) & $GLOBALS (not scoped)

```
TEST_MODE=true $TEST_MODE=true
```

- *symbols:* immutable string whose value is itself

```
favorite_framework = :rails :rails.to_s == "rails" "rails".to_sym == :rails :rails
== "rails" # => false
```

Переменные, массивы, хэши

- не существует описаний переменной
- локальные переменные должны быть назначены перед использованием

- instance & class variables == nil until assigned

- OK: x = 3; x = 'foo'

- Wrong: Integer x=3

- Array: x = [1,'two',:three] x[1] == 'two' ; x.length==3
- Hash: w = {'a'=>1, :b=>[2, 3]} w[:b][0] == 2
  w.keys == ['a', :b]

Методы

- Всё (кроме fixnums) is передаётся по ссылке

```
def foo(x,y) return [x,y+1]
end

def foo(x,y=0) # y is optional, 0 if omitted [x,y+1] # last exp returned as
  result
end

def foo(x,y=0) ; [x,y+1] ; end
```

- Call with: a,b = foo(x,y)or a,b = foo(x) when optional arg used

Основные конструкции

• Утверждения заканчиваются на ';' или новой строкой, но могут и разбивать линию, если разбор однозначен

✔raise("Boom!") unless ✖ raise("Boom!") (ship_stable)

unless (ship_stable)

- Basic Comparisons & Booleans: == != < > =~

!~ true false nil

- Обычный поток управляющих конструкций

Строки и регулярные выражения

"string", %Q{string}, 'string', %q{string} a=41 ; "The answer is #{a+1}"

• Сравните строку с регулярным выражением:

"fox@berkeley.EDU" =~ /(.*)@(.*)\.edu$/i /(.*)@(.*)\.edu$/i =~ "fox@berkeley.EDU"

• Если нет совпадений, значение ложное

• Если совпадают, то значение не ложное and $1...$n *capture* parenthesized groups ($1 == 'fox', $2 == 'berkeley')
/(.*)$/i or %r{(.*)$}i

or Regexp.new('(.*)$', Regexp::IGNORECASE)

Пример

```
rx = {:fox=>/^arm/, 'fox'=>[%r{AN(DO)$}, /an(do)/i]}
```

Какое выражение вернёт результат, отличный от nil ?

```
"armando" =~ rx{:fox

rx[:fox][1] =~ "ARMANDO"

x['fox'][1] =~ "ARMANDO

"armando" =~ rx['fox', 1]
```