درس ۵

Ruby

بلاک و lambda

app.rb

```ruby
def square(x) = x * x
double = ->(x) { x * 2 }
[1, 2].map(&double)
```

همه را یاد بگیرید @MyWebUniversity-Dari

یک کورس رایگان از MyWebUniversity

برنامه‌نویسی Ruby

درس ۵

method ها، بلاک‌ها، proc ها و lambda ها

def، آرگومان‌های پیش‌فرض و keyword، برگشت ضمنی، yield، proc ها و lambda ها



تعریف method ها

def یک method را شروع می‌کند و end آن را می‌بندد. آخرین عبارت خودکار برگردانده می‌شود، پس ...

به رسم، method هایی که جواب بلی یا نه می‌دهند با علامت سوال تمام می‌شوند.

methods.rb

```ruby
def square(x)
  x * x                    # implicit return
end

def passing?(score)
  score >= 50
end

square(7)        # 49
passing? 42      # false
```

01

آرگومان‌های پیش‌فرض و keyword

پارامترها می‌توانند مقدارهای پیش‌فرض داشته باشند. آرگومان‌های keyword، با یک کالن بعد از ...

یک ستاره آرگومان‌های اضافی را در یک آرایه جمع می‌کند، و دو ستاره keyword های اضافی را در یک ...

args.rb

```ruby
def greet(name, greeting = "Salaam")
  "#{greeting}, #{name}!"
end

def book(title:, author:, year: 2026)
  "#{title} by #{author} (#{year})"
end
book(author: "Ferdowsi", title: "Shahnameh")
def total(*nums) = nums.sum       # endless method
```

02

بلاک‌ها و yield

هر method می‌تواند یک بلاک بگیرد. داخل method، yield بلاک را اجرا می‌کند و می‌تواند مقدارها ...

block_given? بررسی می‌کند که آیا یک بلاک فرستاده شده است. map، each و times همین‌طور ساخته ...

yield.rb

```ruby
def twice
  yield 1
  yield 2
end
twice { |n| puts "call #{n}" }

def timed
  start = Time.now
  yield
  puts "took #{Time.now - start}s"
end
timed { sleep 0.5 }
```

03

proc ها و lambda ها

یک بلاک object نیست، اما می‌توانید یکی را به عنوان یک Proc یا یک lambda ذخیره کنید و بعداً ...

نحو تیر lambda، یعنی ->، رایج‌ترین است.
lambda ها تعداد آرگومان را بررسی می‌کنند؛ proc ها نه.

lambda.rb

```ruby
double = ->(x) { x * 2 }
double.call(5)          # 10
double.(5)              # 10
double[5]               # 10

add_tax = proc { |p| p * 1.1 }
[100, 200].map(&double)        # [200, 400]
%w[a b].map(&:upcase)          # ["A", "B"]
```

04

closure ها

بلاک‌ها، proc ها و lambda ها closure هستند: متغیرهای اطراف خود را به خاطر دارند، حتی بعد از ...

اینجا make_counter یک lambda برمی‌گرداند که شمارش خودش را نگه می‌دارد.

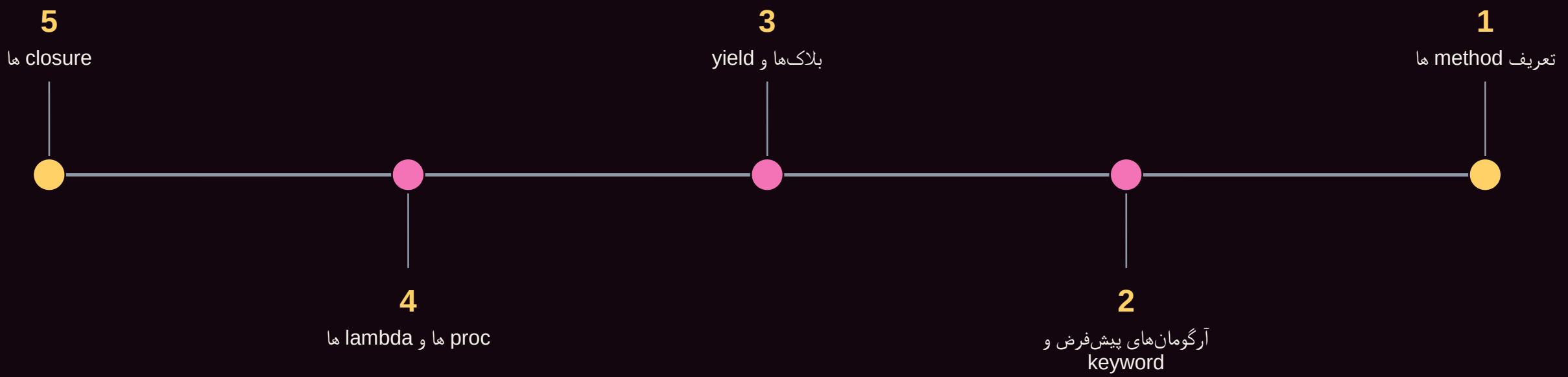
counter.rb

```ruby
def make_counter
  count = 0
  -> { count += 1 }
end

c = make_counter
c.call; c.call
puts c.call         # 3
```

05

نقشهٔ درس


1
تعریف method ها
2
آرگومان‌های پیش‌فرض و keyword
3
بلاک‌ها و yield
4
proc ها و lambda ها
5
closure ها

منابع و مطالعهٔ بیشتر

RUBY PROGRAMMING - 5

methods.rb

```ruby
def square(x)
  x * x                    # implicit return
end

def passing?(score)
  score >= 50
end

square(7)        # 49
passing? 42      # false
```

RUBY PROGRAMMING - 5

yield.rb

```ruby
def twice
  yield 1
  yield 2
end
twice { |n| puts "call #{n}" }

def timed
  start = Time.now
  yield
  puts "took #{Time.now - start}s"
end
timed { sleep 0.5 }
```

تشکر از تماشای شما

MyWebUniversity.com

آموزش رایگان تکنالوژی از سال ۲۰۰۰ — انگلیسی و دری

@MyWebUniversity-Dari · @MyWebUniversity