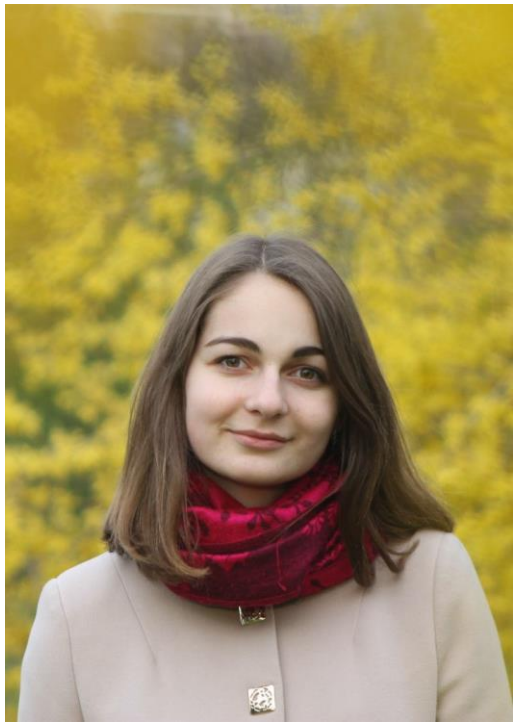
Ruby Essential

Методы и блоки

Автор курса



Юля Гончаренко

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Доступ можно получить через руководство вашего учебного центра

Проверьте как Вы усвоили данный материал на TestProvider.com

Тема

Методы и блоки.

Методы и блоки

1. Что такое методы?
2. Определение простых методов.
3. Работа с аргументами метода.
4. Вызов метода.
5. Определение синглтон-методов.
6. Блоки-определение и применение.
7. Создание Proc и Lambda объектов.

Понятие метода

Метод — это блок кода, который связан с одним или более объектами. Определение метода происходит через ключевые слова def и end.После ключевого слова def задается имя метода, а далее в круглых скобках задаются параметры метода.
Параметры метода используются в виде переменных в теле самого метода. Значения параметров берутся из аргументов вызова метода.

```ruby
def sum(a, b)
  puts "#{a} + #{b} = #{a+b}"
end

sum(1, 9) #=>  puts "#{a} + #{b} = #{a+b}"
```

Вызов метода

Для того, чтобы просто вызвать метод, необходимо лишь написать его имя в нужной части программного кода.

Для вызова метода с аргументами, необходимо после имени метода в круглых скобках или без них указать нужные аргументы.

```
sum(1, 9) #=> puts "#{a} + #{b} = #{a+b}"
```

Работа с аргументами методов

Можно «свернуть» аргументы с помощью звёздочки — тогда метод получит массив в качестве аргумента. В таком случае метод принимает неограниченное количество элементов, и им можно пользоваться как массивом и в теле функции.

Также через звездочку можно разделить аргументы на обязательные и необязательные, просто пометив последний аргумент «звёздочкой».

Если методу будут переданы только обязательные аргументы, в переменной «со звёздочкой» в теле метода будет пустой массив.

```
def sum(*members)
  members[0] + members[1]
end
sum(1, 2)
```

Синглтон-методы

Синглтон методы-это методы, которые определены для единственного указанного объекта. Такой метод будет доступен только в отдельном взятом объекте.

Синглтон метод создается путем включения ссылки на объект в определение метода. То есть для определения синглтон-методов в имени метода указывается имя конкретного объекта через точку.

Синглтон методы полезны в том случае, если нужно добавить метод к объекту, а писать класс-наследник неуместно.

Для вызова синглтон-метода следует указать имя объекта.имя метода.

Код программы

....

.....

....

Объект

....

```
def имя_объекта.имя_метода(параметры)  #Объявление

...
Тело метода
....

end

имя_объекта.имя_метода #Вызов синглтон-метода
```

Блоки

Блоки в Ruby используют настолько часто, что порой ими даже удобнее заменить метод. Но такие ситуации случаются далеко не всегда, и всегда стоит помнить о различиях между блоком и методом:

1)Блоки видят переменные, которые объявлены в области действия блока, а методы - нет.

2)Существует большая разница между чувствительностью методов и блоков к принимаемым аргументам. Методы очень чувствительны к аргументам, можно передать только то количество аргументов которые описаны в методе. Иначе вы получите ошибку вместо результата.

```
{ тело блока }

do
Тело блока
end
```

Proc

Так как с блоками нельзя работать как с объектами, можно воспользоваться аналогами блоков, которые являются объектами — proc и lambda.

Proc объекты по своему поведению напоминают блоки, а lambda схожи с методами. Оба объекта принадлежат к классу Proc.

Proc - это объект, содержащий блок. Данный объект можно копировать, передавать в различные функции и выполнять.

Как и блок - Proc выполняется в том же контексте, где он был объявлен.

Одним из преимуществ Proc-а перед блоком является то, что это обычный объект, а следовательно в одну функцию можно передать несколько таких объектов.

```
def max_element
p = Proc.new { return [1,2,3,4] }
array = p.call
return array.max # здесь был Proc
end
max_element # => [1,2,3,4]
```

Lambda функции

На первый взгляд, lambda ничем не отличается от Proc-а, но на самом деле, несколько различий есть:

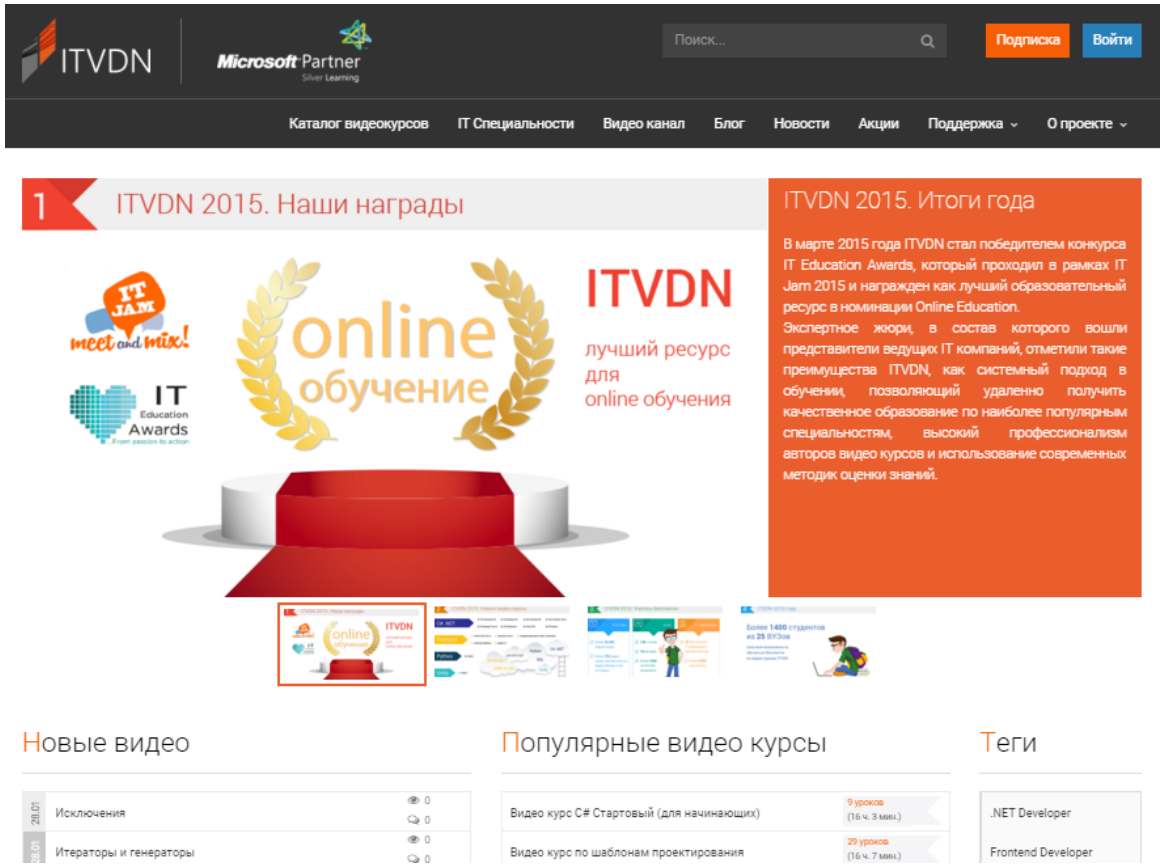
1)lambda-функции больше похожи на обычный метод и поэтому накладывают дополнительные ограничения на входные параметры.

2)lambda-функции похожи на обычные еще и тем, что вызов конструкции return приводит к выходу из lambda-функции, а не из родительской.

```
s=lambda do |a,b|  return a+b
end
puts s.call(1,2)
```

Смотрите наши уроки в видео формате

ITVDN.com



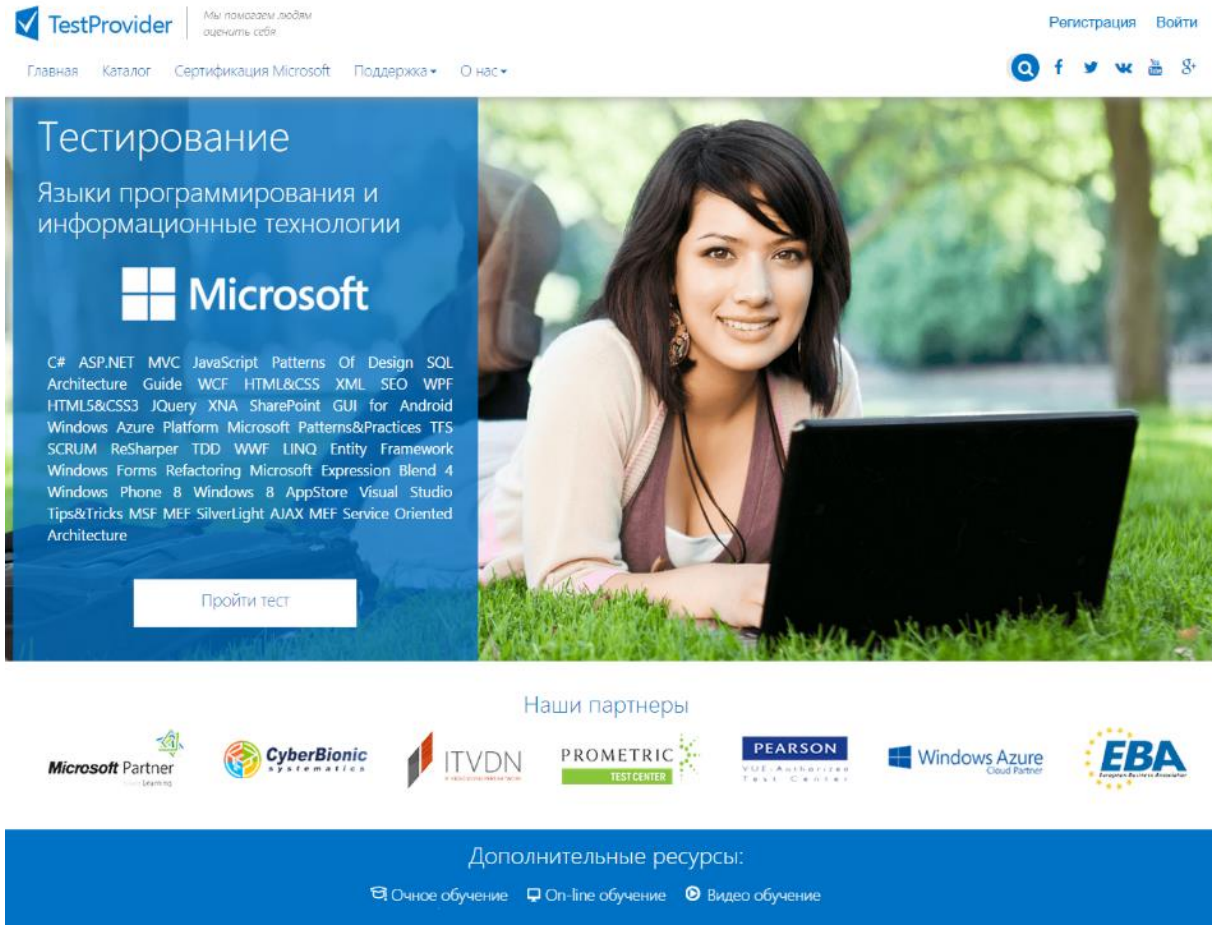
Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com



TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association