Programación para todos

8. Procs y Lambdas

le wagon

@lewagonargentina

Bloques Repaso

Bloques son bloques de código ejecutados dentro de un método (como `each`, `sort`, etc.)

Son similares a los métodos pero no necesitan nombre o ser asignados porque ya están dentro de otra función.

```
names = [“gabriele”, “mariana”, “shannon”]

names.each do|name|
        reversed_name = name.reverse
        putsreversed_name.upcase
    end
```

Nameless block of code

Yield

Y si queremos escribir un método customizado que acepte un bloque, como pasa con `each` y las demás funciones standards?

Usemos **`yield`** keyword!

Cuando el método sea llamado con el bloque, el bloque reemplazará la `yield` keyword dentro del método

```
def welcome_message
   “Welcome!”
yield
end
```

```
puts welcome_message do
   print “Today we’ll see procs”
   puts “and lambdas!
end
```

Everything inside this block replaces the yield!

Procs

Hasta ahora hemos escrito diferentes bloques de código dentro de otros métodos pero nunca almacenados en variables.

Entonces, cómo hacemos para almacenarlos y reusarlos?

Procs fueron hechos para ello!

```
def welcome_message
 puts "Welcome!"
 yield
end
```

```
today_lecture = Proc.new do print
 "Today we'll see procs"
  puts "  and lambdas!
end

welcome_message(&today_lecture)
```

Pass the Proc like this (with an & before)!

Procs with Symbols

En Ruby, el nombre de un método puede ser llamado con un *symbol*, de esta forma `:to_i`, `:to_s`, `:capitalize`, etc.

Y cómo hacemos si queremos pasar ese método con un proc?

We can easily do it by converting the symbol to a proc!

```
names = [ “gabriele”,  “mariana”, “shannon” ]
```

~~`names.map! { |name| name.capitalize }`~~

```
names.map!(&:capitalize)
```

↑

Note the colon, we are using a symbol here!

Lambdas

Con algunas pocas excepciones en sintaxis y comportamiento,
lambdas son identicas a los procs

```
def welcome_message
 puts “Welcome!”
 yield
end
```

```
today_lecture_proc = Proc.new do
 print “Today we’ll see procs”
 puts “and lambdas!
 end

today_lecture_lambda = lambda do
 print “Todaywe’ll see procs”
 puts “ and lambdas!
end

welcome_message(&today_lecture_lambda)
```

Gracias! :)

le wagon

@lewagonargentina