

Programming for Everybody

7. Refactoring



le wagon

@lewagonargentina

Lo lindo de Ruby

Ruby *prioriza las necesidades del humano por sobre las de las computadoras*, y es el lenguaje de programación más parecido al habla inglesa. Y al ser el más intuitivo, lo hace más productivo (y feliz) al programador! 🌟😊

Dado que la felicidad del programador es el principal objetivo de Ruby, hay muchísimos *syntax shortcuts* (*atajos*) que ayudan a escribir código rápido, limpio y de una forma más eficiente.

Usenlos, y verán que su código se irá transformando a un lenguaje normal!

One-line Conditionals

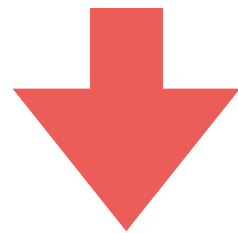
Cuando el bloque dentro de un condicional (como `if` o `unless`) está solo tomando una línea, lo podemos escribir todo en la misma línea y dejar la condición después del bloque (sin especificar el `end` keyword):

```
age = 20
```

```
if age >= 18
```

```
  puts "you can vote!"
```

```
end
```



```
puts "you can vote!" if age >= 18
```

Operador Ternario (one-line `if-else` statement)

Una más rápida y concisa versión del `if-else` statement es la expresión condicional ternaria

Toma tres argumentos: una condición, código a ejecutar si la condición es `true`, y código a ejecutar si la condición es `false`.

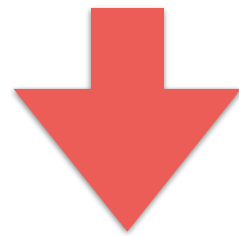
```
condition ? do this if true : do this if false
```

Operador Ternario (one-line if-else statement)

Usarlo solamente cuando la condición se puede escribir en una línea!

```
if age >= 18  
  puts "you can vote!"  
else  
  puts "you can't vote yet!"  
end
```

puts can be
before, DRY
(don't repeat
yourself!)



```
puts age >= 18? "you can vote!" : "you can't vote yet!"
```

Case Statement (o Switch)

Usarlo cuando tenemos muchos `elsifs` conditions, es más rápido y limpio!

```
puts "Which language are you learning?"  
language = gets.chomp  
  
case language  
  when "ruby" then puts "Web apps!"  
  when "css" then puts "Style!"  
  when "python" then puts "Data science!"  
  else puts "Sounds interesting!"  
end
```

Conditional Assignment

Usarlo para asignar una variable solo sino fue asignada aún!


```
teacher = nil
```

```
puts "Let's start today lecture!"
```

```
teacher = "gabriele"
```

```
teacher ||= "mariana"
```

It won't be reassigned here,
because the variable
teacher is not empty!



```
puts "Today's teacher is #{teacher}!"
```

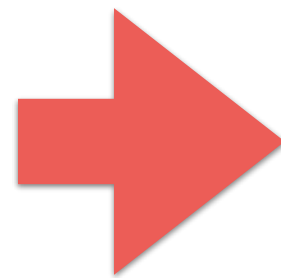
Implicit Return

En Ruby nos siempre necesitas `return` keyword para devolver un valor de un método (en la mayoría de los lenguajes de programación lo necesitas!)

Si no especificas un `return`, el método devuelve el resultado de la última línea del código

Usar `return` solo cuando no es la última línea del método!

```
def sum(a, b)  
  return a + b  
end
```

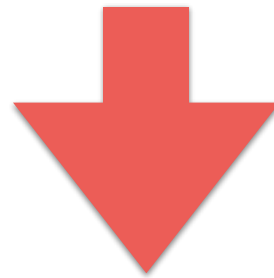


```
def sum(a, b)  
  a + b  
end
```


One-line Blocks

Cuando un bloque (recuerden, un bloque es el código dentro de un método predefinido) es de solo una línea, puedes poner el método entero en una línea y usar curly brackets { } en vez de `def` y `end`

```
[ "gabriele", "mariana" ].each do  
  |name| puts name.capitalize  
end
```



```
[ "gabriele", "mariana" ].each { |name| puts name.capitalize  
}
```

Gracias!!



le wagon

@lewagonargentina