Introducción a Ruby

Bloques

Overview

- Bloques
- Cómo son utilizados
- Cómo incorporarlos dentro de nuestros métodos

Bloques

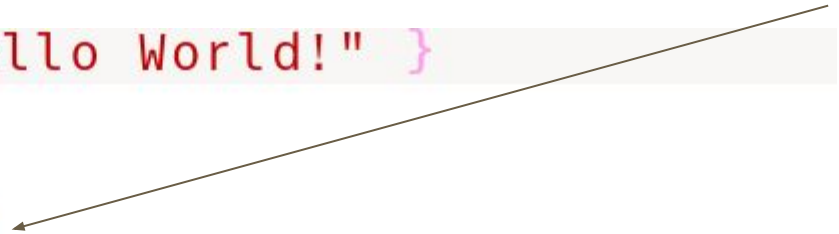
- No son métodos.
- Son porciones de código.
- Encerrados entre:
  - llaves {}: cuando es una sola línea de código.
  - do y end: cuando ocupa varias líneas de código.
- Son pasados a los métodos como parámetros.
- Son utilizados comúnmente en los iteradores.
- Pueden recibir argumentos, a menudo encerrados entre ||

Bloques - Parámetros

```ruby
1.times { puts "Hello World!" }
# => Hello World!

2.times do |index|
  if index > 0
    puts index
  end
end
# => 1

2.times { |index| puts index if index > 0 }
# => 1
```



Codificando con bloques

Existen dos formas de utilizar un bloque en nuestros métodos:

- Implícitamente
  - Utilizar bloque**?** para verificar que el bloque fue pasado como parámetro.
  - Utilizar **yield** para llamar al bloque.
- Explícitamente
  - Utilizar **&** en frente del parámetro que recibe el método (def metodo (&bloque))
  - Utilizar **call** para llamar al bloque.

Implícitamente

Se necesita verificar block_given? sino una excepción es lanzada.

```
def imprimir_dos_veces_implicito
  return "No se recibió ningún bloque" unless block_given?
  yield
  yield
end

puts imprimir_dos_veces_implicito { print "Hola!"}
# => Hola!
# => Hola!

puts imprimir_dos_veces_implicito # => No se recibió ningún bloque
```

Explícitamente

Se necesita verificar si el bloque es nil

```ruby
def imprimir_dos_veces_explicito (&i_am_a_block)
  return "No se recibió ningún bloque" if i_am_a_block.nil?
  i_am_a_block.call
  i_am_a_block.call
end

puts imprimir_dos_veces_explicito # => No se recibió ningún bloque
imprimir_dos_veces_explicito { puts "Hola!"}
# => Hola!
# => Hola!
```

Entonces...

- Los bloques son solamente código que es pasado a los métodos como parámetro.
- Para incorporarlos a nuestros métodos se los puede invocar de manera explícita o implícita.