Full Stack Notes

Introduction to Ruby / Methods and Blocks

Methods and Blocks

Methods do things, they are the verbs of computer programming.

Table of Contents

Methods

In Ruby (unlike in Java, for example) we can define methods that aren't explicitly held within a class.

```ruby
def say_goodnight(name)
  puts "Goodnight, #{name}."
```

```
end


say_goodnight('John Boy')

say_goodnight('Matz')
```

Output:

```
Goodnight, John Boy.

Goodnight, Matz.
```

Method Return Values

All methods return data to their caller. Ruby returns the last evaluated expression in the method.

```
def square(x)

  x * x

end


# Note that multiple parameters are separated by commas.

def share_pizza(pieces_left, people)

  if (pieces_left < people)
```

```
    "Sorry I don't have enough pizza."
  else
    "Let's share. Any leftovers go to the dog."
  end
end


puts square(5)
puts share_pizza(4, 3)
```

Output:

```
25
Let's share. Any leftovers go to the dog.
```

Blocks

Any code surrounded by curly braces is a block. With blocks, you can group a set of instructions together so that they can be passed around your program.

Some methods can take blocks as parameters:

```
42.times { puts 'Forty Two' }
```

☝️ The `times` method takes one block argument. That in this case will be executed 42 times.

If your block is longer than one line it’s common practice to replace the curly braces with `do` and `end`:

```
42.times do
  puts 'Forty Two'
  puts 'Wiggle Wiggle'
end
```

In a sense you can think of blocks as being unnamed methods, like anonymous functions in Javascript.

Block Arguments

Block arguments are a set of variables surrounded by pipe characters and separated by commas. They are used to pass data into a block.

```
42.times { |i| puts "#{i} is the meaning of life, the universe, and everything." }
```

☝️ The `times` method takes a block as its argument. If this block accepts an argument then `times` will provide it with the current value of its iteration counter.

Output:

```
0 is the meaning of life, the universe, and everything.

1 is the meaning of life, the universe, and everything.

2 is the meaning of life, the universe, and everything.

<skip a few >

41 is the meaning of life, the universe, and everything.
```

Here is a block that takes two parameters and sums them:

```
{ |x, y| x + y }
```

This could also have been written as:

```
do |x, y|

   x + y

end
```