

Programming for Everybody

7. Refactoring

The beauty of Ruby

Ruby *emphasises human needs over those of computers*, and is the programming language more similar to spoken english. And the more intuitive a language's syntax is, the more productive (and happy) the programmer will be! 🌟😊

Since programmers happiness is Ruby's first goal, *there are a lot of syntax shortcuts* that can help you write code in a faster, cleaner, and more efficient way

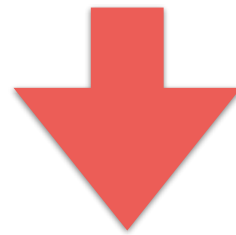
Try to use them, and you'll see your code quickly becoming similar to a normal language!

One-line Conditionals

When the *block* inside a conditional statement (like `if` or `unless`) is **just taking one line**, you can write all the statement on one line, and place the condition after the block (without specifying the end keyword):

```
age = 20
```

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



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

Ternary Operator (one-line if-else statement)

An quicker and more concise version of the if-else statement the **ternary conditional expression**

It takes three arguments: a condition, some code to execute if the condition is true, and some code to execute if the condition is false.

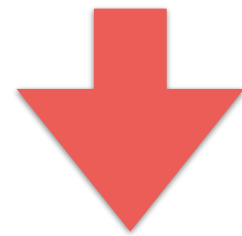
condition ? do this if true : do this if false

Ternary Operator (one-line if-else statement)

Use it again **only** when the condition can be written in one-line!

```
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 (a.k.a Switch)

Use it when you have a lot of elsifs conditions, it's faster and cleaner!

```
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

Use it to assign a variable **only** if it hasn't been assigned yet!

```
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}!"
```

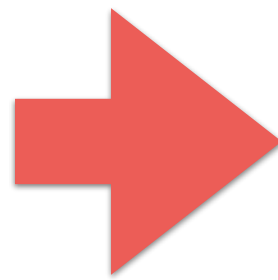
Implicit Return

In Ruby you don't always need the `return` keyword to give back a value from a method (in most of programming languages you do!)

If you don't specify a `return`, the method will return the result of its last line of code

Use `return` **only** when is not in the last line of your method!

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



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


One-line Blocks

When a block (remember, *a block is the code inside a pre-defined method*) is taking just one line, you can put the entire method on one line, and use curly brackets instead of `def` and `end`

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



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

**Let's see some more live
examples!**

Thank you! :)