Handout

Introduction to Ruby 2.0
Lesson 6 - Step 2
Darren Jones

Introduction

So far we've already seen lots of different built-in classes in Ruby such as `String` and `Integer`, as well as some of their methods. Ruby lets us add extra methods to these classes by opening up the class definitions.

Integer Class

Create a new file `extensions.rb`:

```ruby
class Integer
  # doubles the number
  def double
    self * 2
  end
end
```

```ruby
class String

  #make the reverse method useless
  def reverse
    "sorry no reversing"
  end

  # Make a string plural
  def pluralize
    case self
      when "woman" then "women"
      when "person" then "people"
      when "octopus" then "octopi"
      when "sheep" then "sheep"
      else self + "s"
    end
  end
end
```

I've opened up the `Integer` class by typing the keyword `class` followed by the name. Notice that it starts with a capital letter, which is true for all class names in Ruby.

Anything defined inside the block before the `end` declaration will apply to all integer objects. Inside this block, I've defined a new method that will apply to all integer objects called `double`. It will return the value of the integer multiplied by 2.

Inside this method we can see the `self` keyword, that refers to the object itself.

Double Method

Now all integers will have the `double` method. Open up IRB, require your file and run:

```
3.double
```

It should return the doubled integer that called the method.

You can add new methods for all of the built-in classes, such as strings and arrays. In fact, you can even redefine the built-in methods.

Built-in Methods

For example, we can redefine the `reverse` method:

```
def reverse
  "sorry no reversing"
end
```

Instead of reversing a string we just return this useless string.

Changing the way built-in methods behave is very much frowned upon since people expect methods to work in a certain way. So it's something that's usually best avoided. Remember: with great power comes great responsibility. It's very powerful being able to change class methods but something that we should try and avoid most of the time.

Monkey Patching

Adding new methods though, that add more functionality is a good thing, and it's sometimes known as **monkey patching**. Ruby on Rails actually has a module called `ActiveSupport` which adds lots of extra methods to the `String` class amongst others.

Pluralize

One of the methods in there is called `pluralize`, and it will return the plural version of a string. I've attempted to re-create this method:

```
def pluralize
  case self
    when "woman" then "women"
    when "person" then "people"
    when "octopus" then "octopi"
    when "sheep" then "sheep"
    else self + "s"
  end
end
```

It uses a `case` statement to return the plural of some irregular words.

Now you can call

```
"octopus".pluralize
```

to get “octopi”.