5 Interesting Fundamentals of Ruby

By Henry van Wagenberg

Based on Research in March & April 2018

Simple Stuff That was a Good Reminder for Me

- Syntax versus Semantics
- When you type ruby my_program.rb ruby IS the interpreter
- ./ and . Mean "current working directory"
- .. or ../ mean "move up into the folder above
- / takes you to root

- Ruby interpreter fun fact:
- ruby -cw my_program.rb will have the interpreter a syntax check

Global Variables and Load Path

- Global Variables ($) generally dangerous but Ruby puts useful stuff into them for the running of each program, for example "$:" contains the program's load path

What's the right way to think about a Class?

- The most important concept in Ruby is the concept of the object. Closely related and playing an important supporting role is the concept of the class.
- Classes define clusters of behavior or functionality
- Every object is an instance of exactly one class
- Objects are represented by either literal constructors (like "" for a String object) or by variables to which they've been bound

Methods and Message Sending

- Message sending is achieved via the special dot operator: the message (i.e. a method name and maybe arguments (arguments are also objects)) to the right of the dot is sent to the object to the left of the dot.
- The left-of-the-dot object is called the receiver (object) of the message
- The whole universe of a Ruby program consists of objects and the messages that are sent to them.
- A ruby programmer spends his time either specifying the things you want objects to be able to do (by defining methods) or asking the objects to do those things (by sending them messages / calling methods).
- Some method calls take the form of bareword-style invocations, like the call to puts:
- Puts "Hello"
- Message Argument Receiver - Self

Syntax versus Standard Library

Syntax

- if
- =
- ==
- nil
- self
- #
- " "
- -5
- Capital letter starts a constant; lowercase first letter starts a variable
- @
- __FILE__

Standard Library (written in C ("extensions") and in Ruby

- to_i
- Time
- FileUtils
- StringIO
- open-uri
- String (I think)
- Math
- Hash (I think)

Require-ing features and files

load

- load "my_program.rb"

- Append to load path:
- $: << "."

require

- require "./my_program.rb"

require_relative

- require_relative "my_program" or "lib/music/my_sonata_program.rb"

RbConfig

- A Constant referring to a Hash
- rbconfig is a Ruby library package. You can load it into irb with the -r command line flag and request information about how the Ruby library is configured. For example, what is the load path for different libraries that are part of the Ruby installation. Give the Hash key for the library you want.
- The standard library has the key "rubylibdir"

```
RbConfig::CONFIG["rubylibdir"]
```