Web App Developer PT

Day 01 - Intro To Ruby & OOP

Learn the basics of the Ruby language and object oriented programming.

Welcome!

- Welcome to the first day of the rest of your life.
- You are on a journey that will help you achieve anything you want to.
- You will be a confident coder and be able to build complex web applications by the end of this course.

?

Day 01 - Agenda

1. Course intro
2. Programming intro
3. Ruby intro
4. Arrays, loops, hashes
5. Objects and classes
6. Your own app idea

?

Course Information

Outcomes of course

1. You will have built several web applications including your own app idea
2. Be confident in being able to build any web application you want
3. Be able to solve your own challenges by connecting with the resources on the web
4. Be able to take on any other technology you want to in the future

?

Methods

- Hands on coding every class
- Just enough theory
- Relatable projects
- In class challenges - to reinforce learning and gain confidence
- Work on your own app project

?

Your own project

- The idea
- Application design
- Customer development
- Iterative design and development
- Deployment and optimisation
- Pitching your app to customers/investors

?

Course content

1. Programming
2. Application architecture and design
3. Databases and internet technologies
4. Full stack Rails and web applications
5. Deployment and optimisation

?

Object Oriented Programming

Computer programming

- A process
- Solves problems
- Involves creating a sequence of instructions (algorithms) that automate tasks to solve the problem

Ada Lovelace

?

Programming activities

1. Analysis
2. Understanding
3. Design
4. Building
5. Testing (plus debugging)
6. Implementation

?

Quality software

1. Reliability
2. Robustness
3. Usability
4. Portability
5. Maintainability
6. Efficiency

?

Procedural Programming

1. Contain a series of steps to be carried out
2. Contain procedures (routines/sub-routines/methods/functions) and they can be called by other procedures and even itself
3. Languages: COBOL, C, Fortran, BASIC

?

Object Oriented Programming

1. Helps create modular and re-usable software
2. Objects can correlate to items in the real world
3. Have a ‘state’ (data) and ‘behaviours’ (methods/ actions)
4. Languages: Objective-C, C++, Python, Ruby, Java, PHP, Smalltalk, Javascript, Perl, C#

?

Object Oriented Programming

1. An object has attributes
   - Eg. A customer in an ecommerce application has a name, address, credit card number, email address
2. An object has behaviours (methods)
   - A customer could ‘update address’, ‘make payment’

?

Object Oriented Programming

CUSTOMER

Attributes

- name
- address
- email
- credit card number

Methods

- update address
- update email
- make payment
- update credit card number

?

Classes and Instances

- A class is the declaration of the object's attributes and available methods. The idea of an object.
- An instance is a single realisation of an object.
- eg. A customer object vs. Customer with ID: 23
- We can instantiate an object (create a new realisation of the object).

?

Syntax

Every language has a similar but different syntax.

Symbols are used along with words to help us translate our intentions into instructions that a computer can understand.

?

Data Types

So that we can work with types of data, we group data into different data types.

- integer - whole number
- decimal - number with a decimal place (0000000.00000000)
- float - a bigger number with a decimal place
- string - any value (letters, numbers, symbols, spaces) 256 characters
- text - same as string but heaps more space
- date, time & datetime
- Boolean - true/false, on/off, 1/0
- array - a list of items separated by commas in square brackets ['dog', 'cat', 'cow']
- hash - a list of named pairs (key value pairs) {['name', 'Pedro'], ['name', 'James']}

?

Variables

A word or symbol that holds a value.

This value can change during the operation of an application (during runtime).

The variable can be referenced and depending on what datatype its value is, operations can be performed on it.

When we assign a value to a variable, we do it ‘right to left’.

```
# a string being assigned to the variable 'name'
name = "Pedro"
```

?

Ruby

About

“I hope to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy.

That is the primary purpose of Ruby language.”

- Yukihiro Matsumoto

?

Features

Everything is an object, even numbers, strings etc.

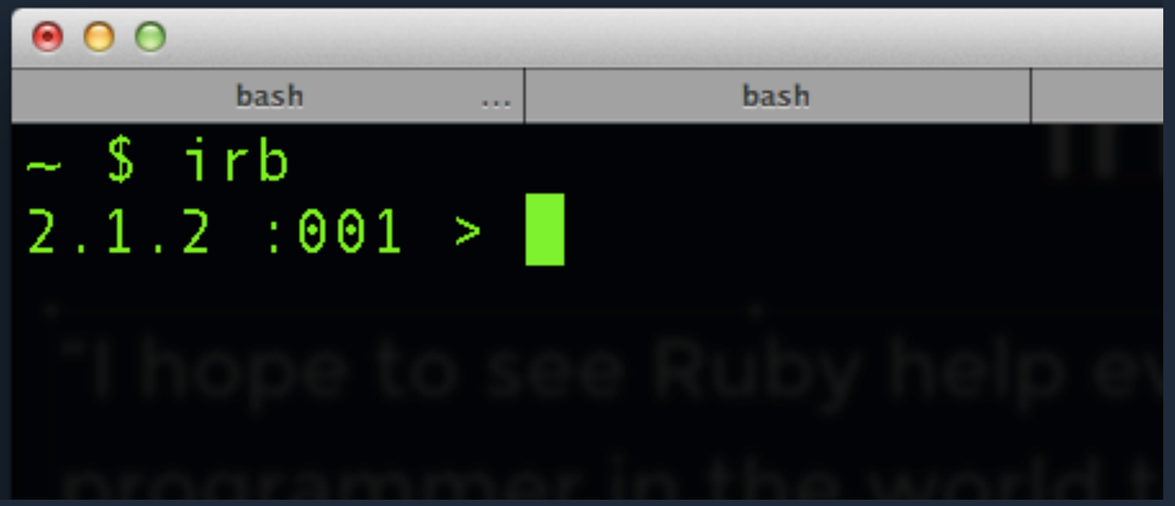
Every method is a function, and methods are called on objects.

Centralised package manager - RubyGems

Garbage collection, exception handling

?

irb



Interactive Ruby shell

?

Ruby basics

```
~ $ irb
2.1.2 :001 > 1 + 1
 => 2
```

Simple maths

```
2.1.2 :004 > puts 1 + 1
2
 => nil
```

puts put to screen

?

Ruby strings and methods

```
2.1.2 :002 > "hello"
 => "hello"
2.1.2 :003 > puts "hello"
hello
 => nil
```

Strings in quotes

```
~ $ irb
2.1.2 :001 > def hello
2.1.2 :002?>   puts "Hello World"
2.1.2 :003?>   end
 => :hello
```

Define a method

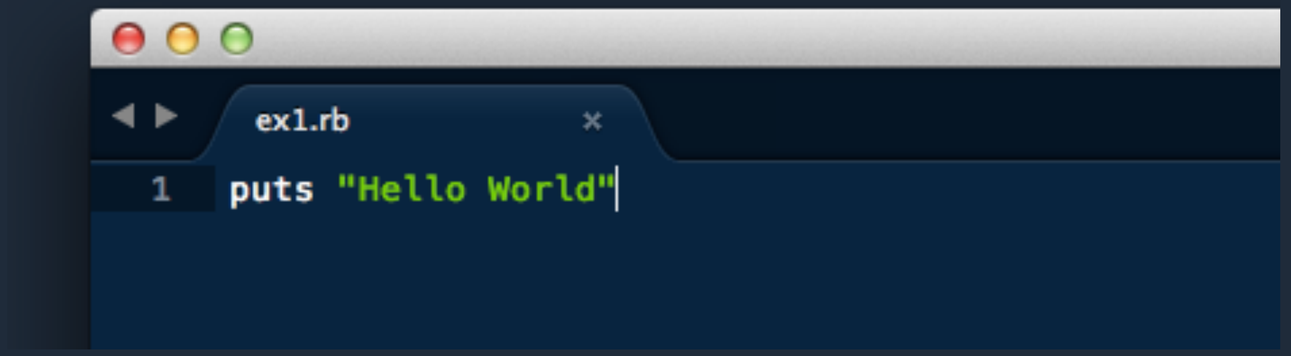
always finish with end

```
2.1.2 :004 > hello
Hello World
 => nil
```

Call method with name.

?

Your first Ruby programme



1. Open Sublime Text
2. New file
3. Type on line 1: `puts “Hello World”`
4. Save file as `/Users/YourName/ex1.rb`
5. In terminal type: `ruby ex1.rb`

?

Exercise 2

1. Create a new file
2. Type:
   puts “Today is the 15th September”
   puts “3 + 3 = #{3 + 3}”
3. Save file as `ex2.rb`
4. In terminal type: `ruby ex2.rb`
5. Try `puts “5 > 7 = #{ 5 > 7 }”`

?

Exercise 3

1. Create a new file
2. Type:

ex3.rb

```ruby
def hello(name)
  result = "Hello " + name
  return result
end

puts hello('Pedro')
```

4. Save file as `ex3.rb`
5. In terminal type: `ruby ex3.rb`

passing variables, return, calling a method

?

ex4.rb

```
1  print "How old are you? "
2  age = gets.chomp
3  print "What's your name? "
4  name = gets.chomp
5  print "Where do you live? "
6  location = gets.chomp
7
8  puts "Hi #{name}, you're #{age} years old, and live in #{location}."
```

?

Exercise 5 - arguments

1. Create a new

2. Type:

```
argument = ARGV

puts "your first argument is #{argument}"
```

4. Save file as `ex5.rb`

5. In terminal type: `ruby ex5.rb hello`

?

Truth - Booleans

1. Create a new file
2. Type:

```
ex6.rb
1 puts true && true
2 puts true || false
3 puts true && false
4 puts 5 < 7 && 4 == 4
```

4. Save file as `ex6.rb`
5. In terminal type: `ruby ex6.rb`
6. `NOT, = vs ==`

?

Your first game

1. Control Structures
2. if, elsif, else

ex7.rb

```ruby
puts "You enter a room with two doors.
      Do you open door 1 or door 2?"

print "> "
door = gets.chomp

if door == "1"
  puts "You won $5000!"
elsif door == "2"
  puts "You stare into the endless abyss :("
else
  puts "You failed"
end
```

?

Arrays
Loops
Hashes

Arrays

An array is a list of items. They can be strings, numbers, objects and any combination of these things.

Arrays are ‘0 indexed’

They look like: [‘dog’, ‘cat’, ‘cow’, 3, ‘2014-05-04’]

You can have multi-dimensional arrays. [[1, 2, 3],[0,4]]

ex8.rb

```ruby
# ex8.rb
animals = ['rat', 'dragon', 'horse']
starsigns = ['aries', 'taurus', 'gemini']

puts "The first item in list of animals is #{animals[0]}"
puts "The second item in the starsigns list is #{starsigns[1]}"
```

Save your file and in your terminal run the programme

Loops

Loops are a way for us to tell the computer to do the same thing over and over as long as a condition remains true.

We give our programme a list of items, it then performs an action for each of the items.

In Ruby, a basic loop looks like:

```
animals.each do |animal|
  animal
end
```

```ruby
# ex8.rb
animals = ['rat', 'dragon', 'horse']
starsigns = ['aries', 'taurus', 'gemini']

puts "The animals in the list are:"
animals.each do |animal|
  puts animal
end

puts "The starsigns are:"
starsigns.each do |ss|
  puts ss
end
```

Hashes

Hashes are a more advanced version of an array.

They contain key-value pairs (or named pairs).

It is how our objects are passed around in a Rails app. Allows us to access the different data associated with the object.

```
customer = { "name" => "Pedro", "starsign" => "Taurus", "age" => 21 }

puts customer["name"]
```

Hashes

1. Create a file called ex9.rb
2. Create the code that describes a customer with 4 attributes and then prints to screen each attribute
3. Run and test your programme.

Hashes

1. Create a file called ex10.rb
2. Create the code that describes 3 customers with 4 attributes and then prints to screen each customer [Hint]
3. Run and test your programme.

Classes
&
Objects

Classes and objects

A class describes a type of ‘thing’.

An object is one of these ‘things’.

The class is a blueprint so that when we tell our programme to create an object (instantiate), it looks to the class to see what to create.

We create an object and assign it to a variable so that we can work with it.

ex8.rb ex9.rb ex10.rb ex11.rb

```ruby
# ex11.rb
class Customer

  def initialize(customer)
    @customer = customer
  end

  def print_customer_details()
    puts "Customer details:"
    puts "Name: #{@customer['name']}"
    puts "Starsign: #{@customer['starsign']}"
    puts "Age: #{@customer['age']}"
    puts "=" * 15
  end
end

pedro = Customer.new({ "name" => "Pedro", "starsign" => "Taurus", "age" => 21 })

pedro.print_customer_details()
```

Classes and objects

1. Add another customer

ex8.rb ex9.rb ex10.rb ex11.rb

```ruby
# ex11.rb
class Customer

  def initialize(customer)
    @customer = customer
  end

  def print_customer_details()
    puts "Customer details:"
    puts "Name: #{@customer['name']}"
    puts "Starsign: #{@customer['starsign']}"
    puts "Age: #{@customer['age']}"
    puts "=" * 15
  end
end

pedro = Customer.new({ "name" => "Pedro", "starsign" => "Taurus", "age" => 21 })

pedro.print_customer_details()
```

Challenge

1. Add another method to the Customer class while taking in a response from the user. eg. Update name of customer (ex12.rb)

2. Create your own programme that defines a class and then a method that works with an object of the class. (ex13.rb)

OOP Terms

class : Tell Ruby to make a new type of thing.

object : Two meanings: the most basic type of thing, and any instance of some thing.

instance : What you get when you tell Ruby to create a class.

def : How you define a function inside a class.

self : Inside the functions in a class, self is a variable for the instance/object being accessed.

Your app idea

What is a web app?

A website that involves user interaction beyond viewing/reading.

An application used in a browser.

Can be used on a PC or mobile device.

Usually requires connection to the internet.

?

What is your idea?

1. What problem are you trying to solve?
2. How will your web application solve this problem?
3. Are there enough people who will pay to have this problem solved?
4. What is the minimum viable product (MVP)?

?

Homework

1. Ruby the hard way http://learnrubythehardway.org/book
2. Hackhands tutorial https://hackhands.com/beginners-guide-ruby/
3. Answer the questions about your app idea on previous slide.

?

End of Day 01