BEWD 10

LESSON 4

4 LEARNING GOALS

CODE CHALLENGE - LOVE_POST

ITERATORS - EXPLORING EACH

COLLECTIONS - HASH BASICS

RUBY CORE - COMBINING CONCEPTS

CODE ALONG - CAR LOT

CODE ALONG - GET IT USING API'S

GIT TIME

GIT TIME

YOU SHOULD KNOW HOW TO

- CREATE A REMOTE BRANCH
- PULL FROM THE UPSTREAM REPO
- PUSH THE CODE TO YOUR FORKED MASTER BRANCH

GIT TIME - PULL

STEP 1: CHECKOUT YOUR MASTER BRANCH

```
git checkout master
 - checkout (or change to) your master branch
```

STEP 2: PULL THE LATEST VERSION OF `UPSTREAM`

```
git pull upstream master
 - pulls the latest version from the `mother_ship`
```

STEP 3: PUSH THE LATEST TO YOUR FORKED VERSION

```
git push origin +master
 - pushes the latest version from the upstream to your forked version
```

STEP 4: CREATE LESSON_THREE BRANCH

```
git branch lesson_two
 - creates a new branch called lesson_three

git checkout lesson_two
 - changes your current branch to the `lesson_three` branch
```

CODE CHALLENGE!

LOVE POST

LOVE POST!

1 - CAPTURE THE NAME OF A LOVE INTEREST

2 - CAPTURE RESPONSE FROM USER

3 - PROVIDE ADVICE USING A SWITCH STATEMENT

3A- MANAGE `EDGE CASES` WHEN AN INVALID RESPONSE IS PROVIDED

KEYS TO SUCCESS

- ONE BRICK AT TIME
- DEBUG WITH PRY EVERY TIME
- CODE PROLIFICALLY

LOVE POST - SOLUTION

```ruby
def get_love_interest
  puts "Who do you love? \n"
  love_interest = gets.strip
  capture_love_interest_response(love_interest)
end

def capture_love_interest_response(love_interest)
  puts "Are you thinking of #{love_interest}?\n"
  puts "Answer 'Yes' or 'No' \n"
  user_answer = gets.strip.downcase
  get_valid_answer(user_answer, love_interest)
end

def get_valid_answer(user_answer, love_interest)
  case user_answer
    when "yes"
      puts "Maybe you should call #{love_interest}?\n"
    when "no"
      puts "Ok, maybe call them soon. You love #{love_interest}!"
    else
      puts "Your answer is the not valid \n"
      puts "Please put 'Yes' or 'No'\n "
      capture_love_interest_response(love_interest)
    end
end

get_love_interest
```

LET'S CODE!

LOVE POST

COLLECTION

<hash review>

HASH REVIEW: LEARNING GOALS

1 - WHAT IS A HASH

2 - HOW TO CREATE A HASH

3 - HOW TO USE 5 HASH METHODS

WHAT'S A HASH?

- A HASH IS A COLLECTION OF UNIQUE KEYS & THEIR VALUES.
- A HASH PRIMARILY USES A STRING OR A SYMBOL AS A KEY.

KEY

VALUE

```
tesla = {year: 2016, model: "Model X", price: "80000"}
```

KEY / VALUE PAIR

```
"tesla"
  {year: 2016, model: "Model X", price: "80000"}
```

HASHES

3 WAYS TO CREATE A HASH

```
1 - Via Instantiation
    Hash.new

2 - Using the Literal Hash Constructor (curly brackets)
    car = {}

3 - Use the Literal Hash Constructor with keys
     car = {"name" => "Tesla", "model" => 'Model X', "year" => 2017}
           ************** OR ************
     car = {name: "Tesla", model: 'Model X', year: 2017}
```

HASHES

COMMON METHODS

```
1 - .length
2 - .merge and merge!
3 - .select
4 - .keys, .values
5 - .values, values_at
6 - .has_key? , .has_value?
```

RUBY DOCS FOR THE HASH CLASS

HTTP://RUBY-DOC.ORG/CORE-2.2.2/HASH.HTML

HASHES

DISCOVER NEW METHODS

HTTP://RUBY-DOC.ORG/CORE-2.2.2/HASH.HTML

CORE

<combining concepts>

CORE: LEARNING GOALS

1 - CREATE AN ARRAY OF HASHES

2 - ITERATE USING THE .EACH METHOD

3 - PRACTICE WITH `IF` & `UNLESS`

4 - CREATE A METHOD THAT ACCEPTS AN UNLIMITED NUMBER OF ARGUMENTS

CODE ALONG

<car_lot.rb>

KEYS TO SUCCESS

- ONE BRICK AT TIME
- DEBUG WITH PRY EVERY TIME
- CODE PROLIFICALLY

LET'S CODE!

CODE ALONG - CAR_LOT

CAR LOT

```ruby
require 'pry'

def show_all_cars(cars)
  cars.each do |car|
    puts "This is a #{car[:brand]}"
    puts "** #{car[:brand]} is environmentally friendly. **" if car[:electric] == true
    # puts "** #{car[:brand]} is environmentally friendly. **" unless car[:electric] == fal
  end
end

#BONUS -> write a method that accepts an unlimited number of cars (or arguments)
def add_cars
end

tesla = {brand: "Tesla", model: "Model X", year: "2016", price: "80000", electric: true}
ford = {brand: "Ford", model: "Escape", year: "2015", price: "17000", electric: false }
porshe= {brand: "Porshe", model: "Speedter", year: "1955", price: "250000", electric: false


####
  cars = []
  cars.push(tesla,ford,porshe)
  show_all_cars(cars)
```

CORE: LEARNING GOALS

1 - WHAT IS AN API?

2 - HOW TO CONNECT & `GET` DATA

3 - LEARN BASIC RETRIEVAL TECHNIQUES

WHAT IS AN API?

<application programming interface>

WHY AN API?

<they allow applications to talk to each other>

GET IT! - PART 1

3 GOALS

1 - MAKE A `GET` REQUEST

2- PARSE DATA USING THE JSON LIBRARY

3- FIND & RETRIEVE REDDIT STORIES

HTTP://WWW.REDDIT.COM/.JSON

CODE ALONG

<get_it.rb>

HOMEWORK

1 - FINISH CAR_LOT & LOVE_POST ON YOUR OWN

2 - BUILD `GET_IT` USING ALL THREE APIS

- REDDIT
- MASHABLE
- DIGG

* STRETCH GOAL - SAVE RESULTS TO A CSV FILE ****