BEWD 13

LESSON 3A

3 LEARNING GOALS

GIT TIME - BRANCH REVIEW

CODE REVIEW - MY REVERSE

COLLECTIONS - INTRO TO HASHES

GIT TIME

GIT TIME - PART 1

STEP 1 - CHANGE TO BEWD_SF_13 DIRECTORY

```
A - Change directories
    - `cd` to your `bewd_sf_13`

B - Make sure you in your `bewd_sf_13`
    - `pwd` your directory should end with `bewd_sf_13`
```

STEP 2 - COMMIT LESSON_ONE CHANGES

```
A - Check your branch
    - `git branch`

B - Check to see what's been staged on the current branch
    - `git status`

    Have notes that you'd like to keep from lesson_two?
    - Add it with: `git add . `
    - Commit it with: `git commit -m "Lesson Two Notes"

C - Create a remote branch on github for lesson_two
      -git push origin +lesson_two
```

GIT TIME - PART 2

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
```

GIT TIME

1 - REVIEW LOCAL BRANCH CREATION

2 - REVIEW REMOTE BRANCH CREATION

CODE CHALLENGE!

REVERSE IT!!

KEYS TO SUCCESS

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

REVERSE IT!

- WRITE OUR OWN 'REVERSE' METHOD
-USE IT TO DETERMINE IF A WORD IS A PALINDROME

RUBY DOCS FOR THE STRING CLASS

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

REVERSE IT! - ANSWER

```ruby
def my_reverse(string)
  char = string.downcase.chars
  word = ""
  until char.length == 0
    word << char.pop
  end
  word.capitalize
end

def is_palindrome?(word)
  if word.downcase == my_reverse(word).downcase
    "Yay! A Palindrome!"
  else
    "Shucks, Not A Palindrome"
  end
end

#####
puts "Please provide a word \n"
word = gets.strip

puts my_reverse(word)
puts is_palindrome?(word)
```

COLLECTION

<objects>

COLLECTIONS

- *CONTAINER FILLED WITH OBJECTS*
- TWO KINDS - ARRAYS & HASHES
- WE REVIEWED ARRAYS LAST SESSION

ARRAY

```
cars = ["tesla", "ford", "bugatti"]
```

HASH

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

HASH: LEARNING GOALS

1 - WHAT IS A HASH

2 - HOW TO CREATE A HASH

3 - HOW TO USE 5 HASH METHODS

4 - DISCOVER METHODS VIA RUBY DOCS

WHAT'S A HASH?

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

KEY

VALUE

```
cars = {}
cars[:tesla] = { year: 2016, model: "Model X", price: "80000" }
cars[:ford] = { year: 1967, model: "Mustang", price: "30000" }
```

KEY / VALUE PAIR

```
cars[: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

LET'S DISCOVER NEW METHODS

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

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

ITERATORS

ARRAYS - USING THE `EACH` METHOD

```
rock_stars = ["Beyonce", "Beatles", "Carlos Santana", "Taylor Swift", "Kanye West"]


#use curl braces a block that's a one-liner
   rock_stars.each { |rock_star} puts "#{rock_star" }

#use the "do/end" format for multiple lines of code
rock_stars.each do |rock_star|
  puts "Hello, I am #{rock_star}. I am a rock star!"
  puts "My name starts with a B" if rock_star.start_with("B")
end
```

LET'S CODE!

CODE ALONG