HTML and server concepts

<span></span> is inline

<div></div> is inblock

Query parameters - www.sex.com/about ?p=1 & g=neat

Fragment parameters - www.sex.com/about #fragment

port parameters - http://localhost:8000/

telnet GET/foo HTTP/1.1 #method, path , version

SHOW INVISIBLE FILES ON MAC

defaults write com.apple.finder AppleShowAllFiles YES

CONFIGURE GIT

git config --global user.name "John Doe";

git config --global user.email johndoe@example.com

$ git config --list #to see the config

git config --global core.editor brackets #change to brakets the editor

ssh -T [git@github.com](mailto:git@github.com) #TO LOG IN MY GITHUB VIA SSH

START A REPO IN A FOLDER WITH GIT

git init

git status # to see the status

git add notes.txt # to keep track of the git we will add

.gitignore # to create a file to ignore

git add . # to add all the files

git commit # it starts to keep track all the changes you do

git commit -m “notes.txt” #it actually works

git log # to see the log

RUBY AND DATATYPES

movie = 'Finding Nemo'

puts ‘Sharon\’s favorite movie is #{movie}'

STRING METHODS

name.capitalize #capitalize a strin

name.split(‘,’) #will return an array containing th string val

name.length #find out the number of characters

name.include? ‘ass’ #find out if include the letter you want

INTEGERS AND FLOATS

age.odd? #return true or false on odd

age.even? #return true or false on even

age.round(1) #it will round an integer

3.times{puts ‘ass’} #will print 3 times ass

ARRAY

numbers = [ ‘one’ ,2 ,3]

puts numbers[0] #will print ‘one’

my\_array = []

my\_array << "A" #add elements to array

my\_array.push "B" #add elements to array

my\_array.push "C" #add elements to array

puts my\_array

my\_array.delete\_at 2 #delete elements to array at postion []

puts my\_array

[4,6,2,4,7,12,9,12].sort

['g', 'e', 'n', 'w', 'a'].sort

["chocolate", "mint", "strawberry", "vanilla", "caramel", "chili"].join

=> 'chocolatemintstrawberryvanillacaramelchili'

HASH - arrays - have numbers associates, hashes - have names associated

my\_hash = {}

my\_hash["AST"] = "Asturias"

my\_hash['GAL'] = "Galicia"

puts my\_hash["AST"]

puts my\_hash['GAL']

puts my\_hash

Above will return :

Asturias

Galicia

{"AST"=>"Asturias", "GAL"=>"Galicia"}

METHODS FOR HASHES

my\_hash = {}

my\_hash["AST"] = "Asturias"

my\_hash['GAL'] = "Galicia"

my\_hash.has\_key?**("AST")**

=> true

my\_hash.has\_key?**("CAT")**

=> false

my\_hash.has\_value?**("Galicia")**

=> true

my\_hash.**select** { |**key**, **value**| **key**.include?("G") }

=> { "GAL" => "Galicia" }

FIND OUT WHAT TYPE OF DATA IS SOMETHING

'what is this?'.**class**

3.**class**

[1,2,3].**class**

REPL ( read–eval–print loop)

As I understand pry helps you find out the right syntax to any shit you need

TO DO

gem install pry

pry #to call pry

<http://ironhack.com:3000/#/learning_unit/223> #to check often

gem install pry-coolline

ls “Marco” #ls method tells you all the methods available for the thing

“Marco”.up HIT tab twice #will help find all the up something methods

[1] pry(main)> show-doc "a string".upcase

U CAN USE SHELL COMMANDS FROM PRY WITH A DOT BEFORE

.ls

.cd ..

DEBUG PRY

require ‘pry’

binding.pry

IRB

irb #from terminal to try shit out

LOGICAL OPERATORS

== #equal, the rest are the same < > <= etc..

&& # both must be true to get true

|| # one have to be true to get true

IF – ELSIF - ELSE

**if** (condition)

code to be executed **if** **this** condition **is** **true**

elsif (another condition)

code to be executed **if** **this** condition **is** **true**

elsif (even another condition)

code to be executed **if** **this** condition **is** **true**

**#add more if needed**

**else** **# the final else is optional!**

code to be executed **if** none of the above are **true**

**end**

LOOPS

FOR

numbers = [1,2,3]

**for** element **in** numbers

puts "-> #{element}"

**end**

WHILE

string = ""

**# While the string's length is less than 10**

**while** string.length < 10

**# Add an 'a'**

string = string + 'a'

end

puts "The final string is #{string}"

ITERATING ARRAYS #EACH

numbers = [ "One", 2, "Three" ]

numbers.each **do** |item|

puts " #{item}"

**end**

ITERATING hashes

my\_hash = {}

my\_hash["AST"] = "Asturias"

my\_hash["GAL"] = "Galicia"

my\_hash["CAT"] = "Catalunya"

my\_hash.each **do** |key,value|

puts "#{key} stands for #{value}"

end

FUNCTIONS

def add(a, b) #2 arguments

puts a + b

end

add**((5 \* 5), (5 + 3))**

=> equals 33

Example 1

puts "What's your food?"

def food(input)

if input == "bacon"

puts "yum"

elsif input == "spinach"

puts "urghh"

else

puts "can i have some bacon"

end

end

input = gets.chomp

food(input)

Example 2

def add(a, b)

calc1 = a + a

calc2 = a + b

calc3 = b + b

**return** calc1, calc2, calc3

end

puts add**(2, 4)**

=> prints [4, 6, 8]

CLASSES

.new #function create instances, called “constructor”

Initialize #initialize the fucntion

@color = color # INSTANCE VARIABLE or is called “state”

**class** **Car**

**def** initialize(color)

@color = color

**end**

**def** honk

puts "Beeeeeeeeep!"

**end**

**def** print\_color

puts @color

**end**

**end**

my\_car = Car.new "red"

other\_car = Car.new "grey"

my\_car.print\_color

other\_car.print\_color

Other method tricky

**class** **Car**

**attr\_accessor** :color

**def** initialize(color)

@color = color

**end**

**def** honk

puts "Beeeeeeeeep!"

**end**

**end**

my\_car = Car.new "yellow"

other\_car = Car.new "brown"

my\_car.color = "grey"

puts my\_car.color

puts other\_car.color

Class Inheritance

**class** **Animal**

**def** initialize(name)

@name = name

**end**

**def** describe

puts "This animal's name is #{@name}"

**end**

**end**

**class** **Dog** < Animal

**end**

**class** **Cat** < Animal

**end**

**class** **Human** < Animal

**def** initialize(name, salary)

**super**(name)

@salary = salary

**end**

**def** describe\_with\_salary

puts "This human's name is #{@name} and its salary is #{@salary}"

**end**

**end**

dog = Dog.new "Winston Furchill"

cat = Cat.new "David Meowie"

human = Human.new "Johnny Appleseed", 12000

dog.describe

cat.describe

human.describe

human.describe\_with\_salary

HTML – CSS