GRABBING STUFF FROM THE WEB

APIS AND COLLECTIONS

OBJECTIVES

- Practice with loops, arrays, symbols, and hashes
- Crush the most popular code challenge ever
- Gems
- APIs
- Using the iTunes API with Ruby!

THIS WILL BE THE COOLEST CLASS SO FAR

FIZZ BUZZ

- Write a program that prints numbers from 1 to 100.
- But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”.
- For numbers which are multiples of both three and five print “FizzBuzz”.
- If the number doesn’t fit those three categories, print the number itself.
- There are like a million ways to do this!!!

NOW

SOME REVIEW

WITH A PARTNER

- Write down all of the characteristics of Arrays that you can think of
- Then do the same for Hashes
- 5 minutes!

SYMBOLS

- They always start with a colon (:my_symbol)
- They’re almost strings, but they cannot be changed, even if you reset a variable, the symbols remains.
- They are, at their core, simply an identifier that you may use.
  - :symbol
  - “string”

SYMBOLS

- We use symbols as hash keys very frequently, but not always.
- Symbols take Ruby less time to compare than strings.
- To compare strings, Ruby has to evaluate them first.
- Since symbols' object id never changes, Ruby can just compare object_id to object_id.

HASHES

- Kind of like a dictionary.
- Hashes always have a key and a value, both can be of any data type, usually strings or symbols.
- They key is to reference the value easily
- A Hash is a collection of key-value pairs
  - "key" => "value".
- It is similar to an Array, except that retrieving data is not limited to only integers

HASHES

- Two types of syntax to worry about unfortunately
- With symbols:

```
myhash = { name: “vincent”, languages: [“english”,
“japanese”], friends: [“paola”, “derek”, “kejal”, “brian” ],
home: “Brooklyn” }
```

- With any other type:

```
myhash = { "name" => “vincent”, "languages" => [“english”,
“japanese”], "friends" => [“paola”, “derek”, “kejal”,
“brian” ], "home" => “Brooklyn” }
```

HASHES

- Updating a hash with strings as keys
- `myhash = { “string” => “my hashes value” }`
- `myhash[“another”] = “my second value”`
- The hash now has 2 keys, “string” and “another”. Both are String types.

HASHES

- Updating a hash with symbols as keys
- `myhash = { :mysymbol => “my key is a symbol” }`
- `myhash[:another] = “my second value with a symbol key”`
- The hash now has 2 keys, :mysymbol and :another. Both are Symbol types.

PARTNER UP AGAIN

- Make sure you have a class4 directory under your name
- In that directory, create a file **music.rb**
- In the next 5 minutes, please complete:
  - Assign a variable to an array
  - In the array, include 5 singers or bands as strings
- 5 minutes. Seriously!

ITERATION:
THE REPETITION OF A PROCESS OR UTTERANCE

Google

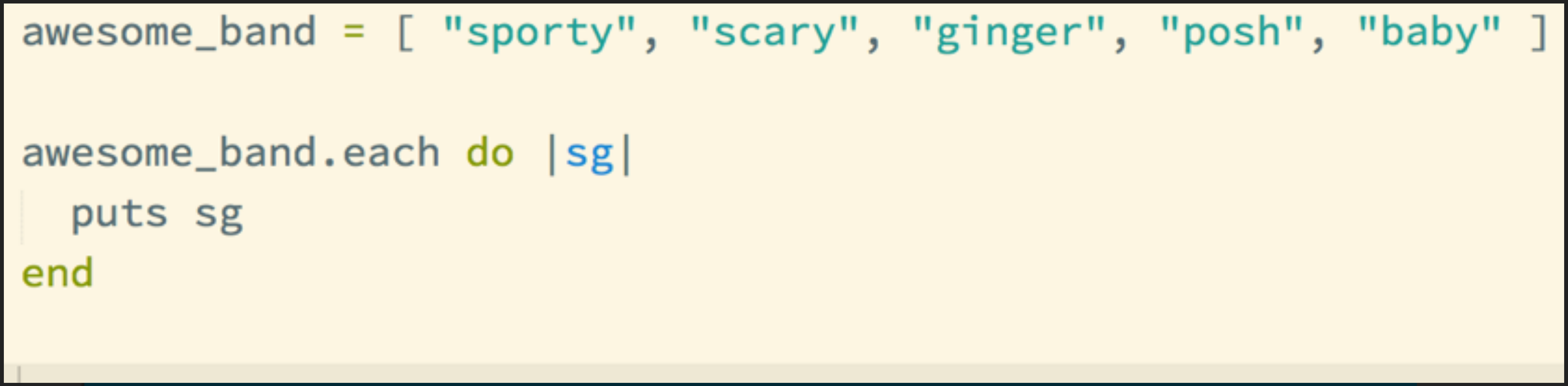
ITERATING IN RUBY

- Ruby collections (arrays and hashes) can be iterated over.
- This is just a fancy way of saying you take the first one and do something with it, take the next one, do something with it, and so on until you reach the end of the collection.

RUBY'S .EACH METHOD

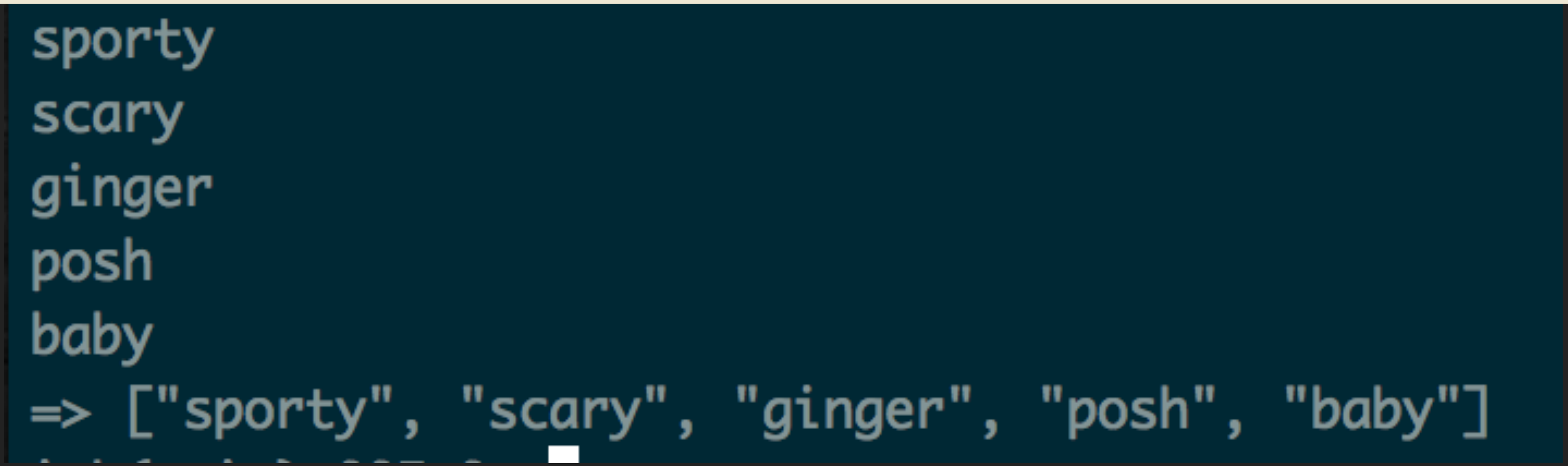
- A way to perform an operation on "each" item in a collection.
- An item in a collection is a thing in an array, or a value in a hash
- The each method allows us to perform a routine on every item in the collection

THE .EACH METHOD OUTPUTS THE ORIGINAL ARRAY



```ruby
awesome_band = [ "sporty", "scary", "ginger", "posh", "baby" ]

awesome_band.each do |sg|
  puts sg
end
```



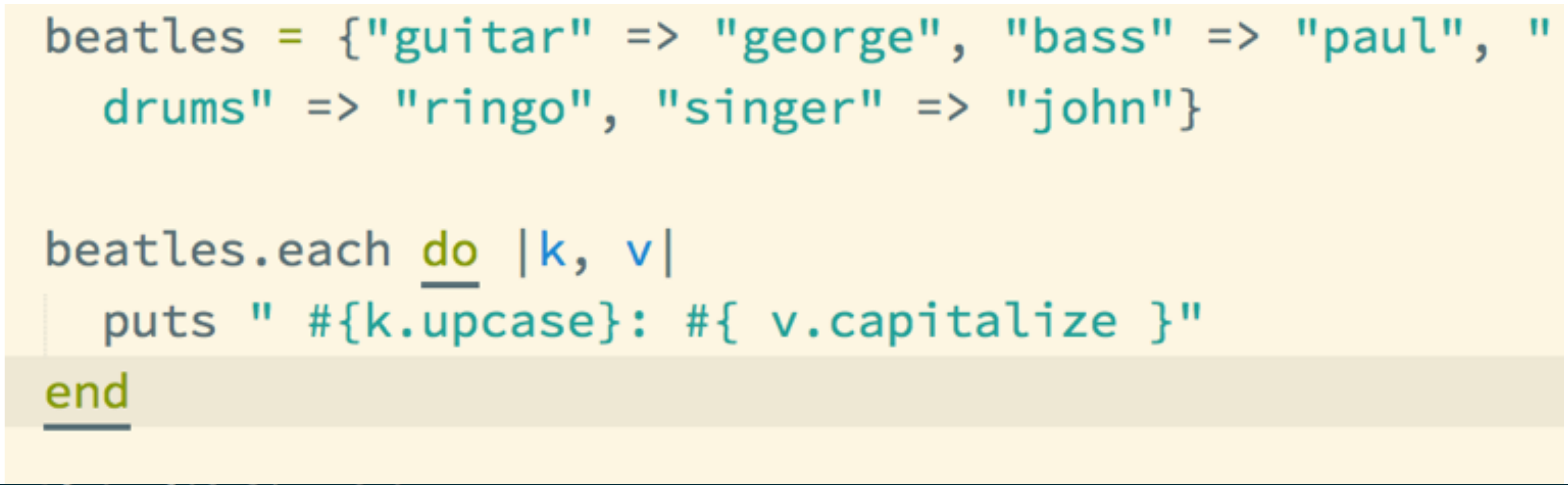
```
sporty
scary
ginger
posh
baby
=> ["sporty", "scary", "ginger", "posh", "baby"]
```

GIVE IT A TRY ON YOUR BAND ARRAY

- Edit the music.rb file to iterate over the bands array.
- In your iteration loop, try some of the String methods that you know like `.reverse`, `.upcase`, etc.

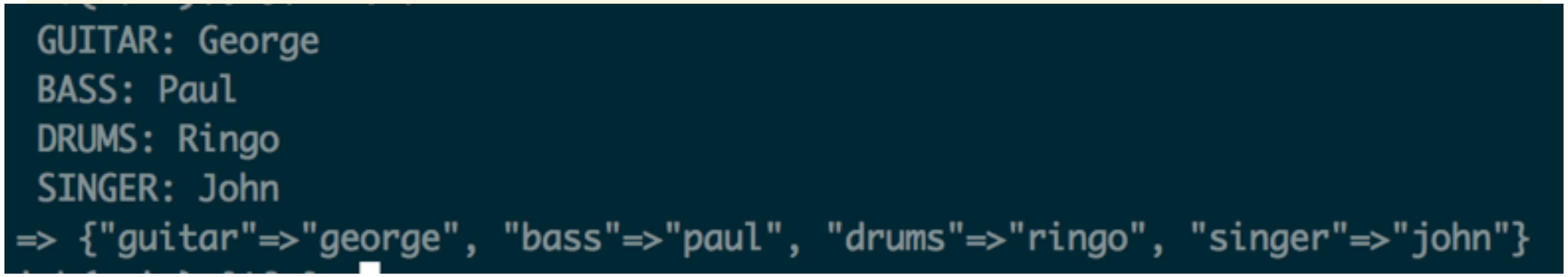
ITERATING OVER A HASH

- You can use .each on a hash too, but you need to have an iterator for both key and value.



```ruby
beatles = {"guitar" => "george", "bass" => "paul", "
  drums" => "ringo", "singer" => "john"}

beatles.each do |k, v|
  puts " #{k.upcase}: #{ v.capitalize }"
end
```



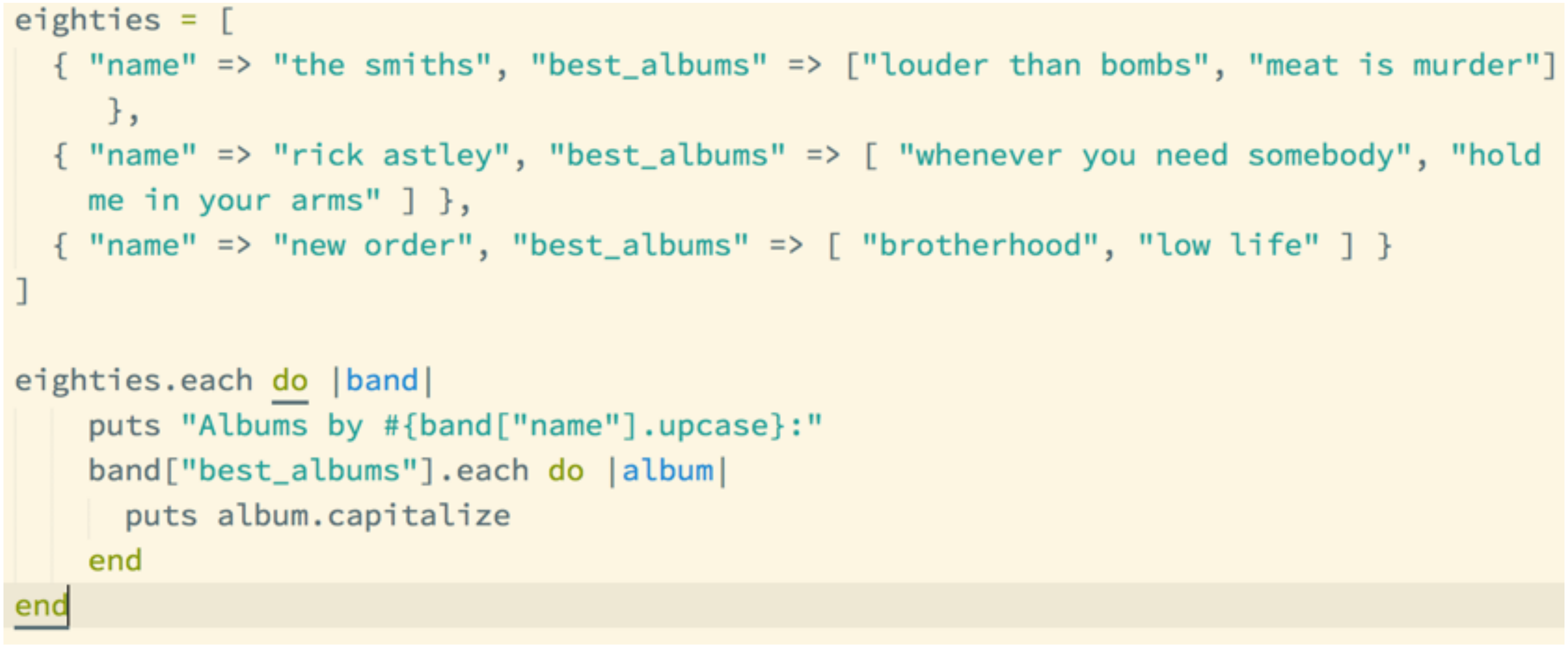
```
 GUITAR: George
 BASS: Paul
 DRUMS: Ringo
 SINGER: John
=> {"guitar"=>"george", "bass"=>"paul", "drums"=>"ringo", "singer"=>"john"}
```

Hash Exercise

- Create a hash that represents the plural version of these animals:
- - moose
- - sheep
- - dog
- - walrus
- - mouse
- Iterate over the hash, Print the singular version, and what the plural version is on the screen

ARRAYS AND HASHES TOGETHER

- A very common pattern is to have an array in a hash, which you can iterate over as you iterate over a hash.



```
eighties = [
  { "name" => "the smiths", "best_albums" => ["louder than bombs", "meat is murder"]
     },
  { "name" => "rick astley", "best_albums" => [ "whenever you need somebody", "hold
    me in your arms" ] },
  { "name" => "new order", "best_albums" => [ "brotherhood", "low life" ] }
]

eighties.each do |band|
    puts "Albums by #{band["name"].upcase}:"
    band["best_albums"].each do |album|
      puts album.capitalize
    end
end
```

Hash Exercise 2

- Create an array of hashes that represents countries and the language(s) that are spoken there. Keep in mind that some have more than one language
- - Sweden
- - Japan
- - Switzerland
- - India
- Print the name of the country, and what languages are spoken there as strings.

WHAT IS THE POINT?

IMAGINE A MATURE WEB APP

- Imagine you own facebook.
- A user wants to see each of his/her status updates.
- You can list them in order on the profile page
- So far we have been typing in our data. Today, we will leverage Apple iTunes for data.

GEMS!

APIS!

APIS

- API, an abbreviation of application program interface, is a set of routines, protocols, and tools for building software applications.
- You can use external API's (ones other people have created) for your own use
- We can use this data for our own benefits. IE: Weather, Traffic, even Subway Info!
- API's can provide information very easily. Today, we will access music info from iTunes.

RUBY GEMS

- Gems are packages of ruby code that you can install and use. These are open source, but
- There are 96,000+ gems available for public install
- https://www.ruby-toolbox.com is a convenient place to search for gems that might solve a problem for you.
- rubygems.org is the community run host for gems.
- You install gems with gem install {gemname}

LET'S HTTPARTY

- go to your terminal (or iTerm2) and type `gem install httparty`
- run `gem list` and make sure you can see it
- httparty is a client for HTTP.

JSON

- `{"NY":"New York","LA":"Los Angeles”,"SYD":"Sydney","LDN":"London"}`
- Does this look a little bit like a ruby data type we know?
- A standard format used to represent data that computers can understand.
- Stands for Javascript Object Notation
- Has arrays, hashes, strings, numbers, null (nil), and booleans
- API’s mostly use JSON to represent their data.

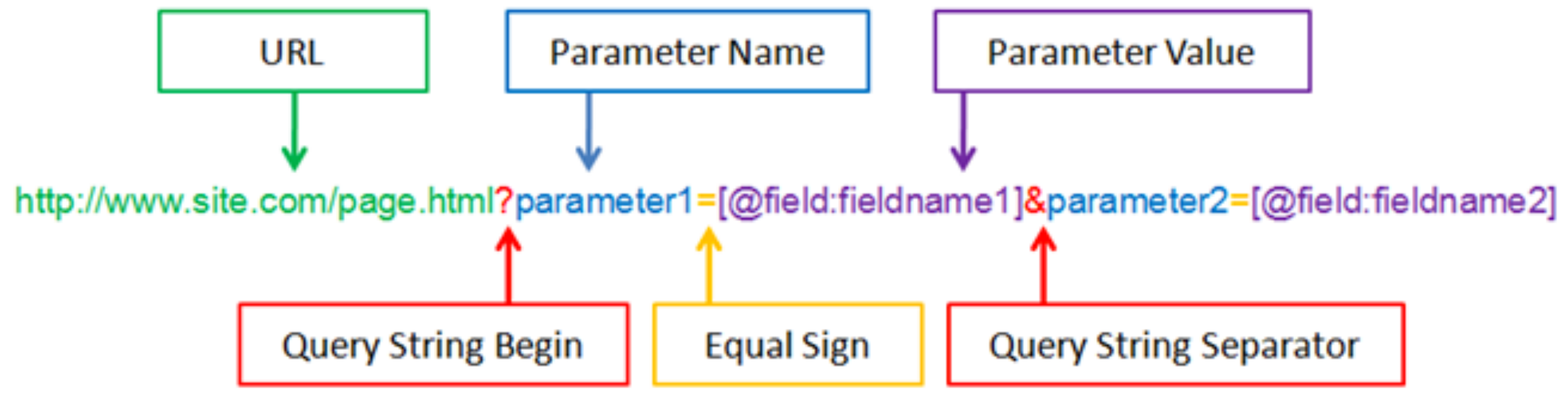
ITUNES API CODEALONG

Head to

https://www.apple.com/itunes/affiliates/resources/documentation/itunes-store-web-service-search-api.html

QUERY PARAMETERS

- parts of a url:


URL
Parameter Name
Parameter Value
http://www.site.com/page.html?parameter1=[@field:fieldname1]&parameter2=[@field:fieldname2]
Query String Begin
Equal Sign
Query String Separator

- Query parameters are a way of passing data to the host in the form of a hash!
- Search terms are one common use of query params

INSTALL JSONVIEW

- Great chrome extension

- https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc?hl=en

RECAP

- Use .each to iterate over an array or hash.
- Some companies are cool enough to let you get their data as APIs. This is often in the form of JSON, which is a nice lingua franca of data.
- Ruby gems like HTTParty are libraries of code that you can use in your project for free.
- Use require 'somegem' to bring gem capabilities into your application.