Learning Ruby - 3

Ruby Hashes

Guess What?

- Ruby hashes are cool!
- Unlike arrays, which associate object references with numbered indices, hashes associate object references with names
- AKA "associative arrays", "maps" and "dictionaries" - stick with "hash"
- The name is AKA the "key", while the object reference is AKA the "value"
- Together, hash entries are referred to as "key/value" or "name/value" pairs

Playing with Ruby Hashes

```
songs = {}

songs = { "July" => :Mundy,
          "I Predict a Riot" => :KaiserChiefs,
          "Rainbow" => :Mundy }

classic_rock = {
   'Smoke on the Water' => 'Deep Purple',
   'Stairway to Heaven' => 'Led Zeppelin' }

songs["La La La La La"] = :KaiserChiefs
songs["Na Na Na Na Na"] = :KaiserChiefs

songs
classic_rock
```

Ruby Hash Methods

```
songs.keys
songs.values
songs.sort
songs.collect

classic_rock.collect do | song, artist |
   puts "#{artist} performs '#{song}'."
end # of do.


puts songs.collect { | song, artist | "#{artist} performs '#{song}'." }

puts songs.collect { | song, artist | "#{artist} performs '#{song}'." }.sort
```

Working with Hashes

```
songs.delete( 'La La La La La' )

classic_rock.empty?
classic_rock.has_key?( 'Satisfaction' )

songs.length

all_songs = songs.merge( classic_rock )

puts all_songs.collect do | song, artist |
   "#{artist} performs '#{song}'."
end.sort # of do.

my_all_time_fav = all_songs.to_a
my_all_time_fav[2]
```

More ... Ruby So Far

- Ruby hashes are cool!
- Hashes can store **any object reference** ... another hash, another array, another object ... anything!
- This can be very, very flexible, efficient and (if you pay attention) **bug-free!**