HASH & SYMBOLS

DATA TYPES WE KNOW

```
42                          # Fixnum
1.25                        # Float
true                        # Boolean
"hello world"               # String
[ "a", "e", "i", "o", "u" ] # Array
```

LET'S TAKE AN EXAMPLE

STUDENTS

```
students =     [ "Peter", "Mary", "George", "Emma" ]
student_ages = [ 24      , 25    , 22       ,  20    ]
```

Write a program to display a list of students with age

```
students.each_with_index do |student, index|
  age = student_ages[index]
  puts "- #{student} (#{age} years old)"
end
```

Do you like this solution?

WHAT IF WE COULD DO?

```
puts students_age["Peter"]
```

WE CAN!

```
students_age = {
  "Peter" => 24,
  "Mary" => 25,
  "George" => 22,
  "Emma" => 20
}
```

HASH

A Hash is a dictionary-like collection of **unique** keys.
For each key, a **value** is associated.

rubydoc

READING KEYS

```
city = {
  "name" => "Paris",
  "population" => 2211000
}
```

You can access hash value by **keys** with:

```
city["name"]        # => "Paris"
city["population"]  # => 2211000
```

ADDING KEYS

```
city = {
  "name" => "Paris",
  "population" => 2211000
}
```

You can add a new key to your hash with:

```
city["star_monument"] = "Tour Eiffel"
```

WRITING KEYS

```
city = {
  "name" => "Paris",
  "population" => 2211000
}
```

You can update the hash with:

```
city["population"] = 2211001
```

SIMILAR TO ARRAY?

```
cities = [ "London", "Paris", "NYC" ]
city = {
  "name" => "Paris",
  "population" => 2211000
}
```

Array are accessed by **indexes**, Hash by **keys**

```
cities[0]    # => "London"
city["name"] # => "Paris"
```

#EACH

```ruby
city = { "name" => "Paris", "population" => 2211000 }

city.each do |key, value|
  puts "The city #{key} is #{value}"
end
```

This code will print:

```
# The city name is Paris
# The city population is 2211000
```

LET'S TAKE TWO CITIES

```
paris = {
  "name" => "Paris",
  "population" => 2211000
}

london = {
  "name" => "London",
  "population" => 8308000
}
```

`name` and **`population`** are keys of the two hashes. They are **identifiers**

SYMBOLS

In Ruby, when in need of internal **identifiers**, we use **symbols**

```
:name  # this is a symbol. This is a new data type
```

ruby-doc.org/core-2.3.0/Symbol.html

SYMBOL PLAY NICELY WITH HASH

```
paris = {
  :name => "Paris",
  :population => 2211000
}
```

is equivalent to:

```
paris = {
  name: "Paris",
  population: 2211000
}
```

As ruby developers, we prefer the latter syntax.

SYMBOL VS STRING

Use symbols when you need an identifier. Perfect for `Hash` keys.

- Data: use a string
- Identifier or tag: use a symbol

Great answer: stackoverflow.com/a/16621092/197944

USING HASH AS A METHOD ARGUMENT

```
tag("h1", "Hello world")
# => <h1>Hello world"</h1>

tag("h1", "Hello world", { class: "bold" })
# => <h1 class='bold'>Hello world"</h1>

tag("a", "Le Wagon", { href: "http://lewagon.org", class: "bt
# => <a href='http://lewagon.org' class='btn'>Le Wagon</a>
```

```
def tag(name, content, attrs = {})
  flat_attrs = attr.map { |key, val| "#{key}='#{val}'" }.joi
  "<#{name} #{flat_attr}>#{content}</#{name}>"
end
```

RAILS WILL USE A LOT OF THIS

DATA FORMAT

- CSV
- JSON / XML

CSV / ARRAY

```
# file.csv
Paris,2211000,"Tour Eiffel"
London,8308000,"Big Ben"
```

```
require "csv"
CSV.foreach("file.csv") do |row|
  # row is an array. For first iteration:
  # row[0] is "Paris"
  # row[1] is 2211000, etc.
end
```

JSON / HASH

A JSON document will look like this:

```
{
  "name": "Paris",
  "population": 2211000
}
```

And in Ruby, we'll get

```
require "json"
JSON.parse('{ "name": "Paris", "population": 2211000 }')
# => { "name" => "Paris", "population" => 2211000 }
```

JSON IS EVERYWHERE IN APIS

Example: Facebook Graph Explorer