

# **HASH & SYMBOLS**

# DATA TYPES WE KNOW

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

# **FROM ARRAYS TO HASH**

**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 their age

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

## DO YOU LIKE THIS SOLUTION?

- Imagine with **10k students** in the array.
- You have to know that "**sebastien**" has index **6783** to read his age.
- Hard to maintain, you'll make mistakes when updating or injecting students.

# 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

```
paris = {  
  "country" => "France",  
  "population" => 2211000  
}
```

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

```
paris["country"]    # => "France"  
paris["population"] # => 2211000
```

# ADDING KEY/VALUE

```
paris = {  
  "country" => "France",  
  "population" => 2211000  
}
```

You can add a new key/value to your hash with:

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

# UPDATING VALUE

```
paris = {  
  "country" => "France",  
  "population" => 2211000  
}
```

You can update the hash with:

```
paris["population"] = 2211001
```

# DELETING KEY/VALUE

```
paris = {  
    "country" => "France",  
    "population" => 2211000,  
    "star_monument" => "Tour Eiffel"  
}
```

You can delete a key/value with:

```
paris.delete("star_monument")
```

# #EACH

```
paris = { "country" => "France", "population" => 2211000 }  
  
paris.each do |key, value|  
  puts "Paris #{key} is #{value}"  
end
```

This code will print:

```
# Paris country is France  
# Paris population is 2211000
```

# CUSTOM METHODS

```
paris = {  
  "country" => "France",  
  "population" => 2211000,  
  "star_monument" => "Tour Eiffel"  
}  
p paris.has_key?("country")  
p paris.has_key?("language")  
p paris.keys  
p paris.values
```

# 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"
```

# MORE READABLE FOR RICH DATA

Which one do you prefer?

```
cities = [ ["London", "England", "Big Ben"], ["Paris", "France", "Tour Eiffel"] ]

cities = {
  "London" => { "country" => "England", "monument" => "Big Ben" },
  "Paris" => { "country" => "France", "monument" => "Tour Eiffel" }
}
```

Well, what's the more readable?

```
cities[1][2]
cities["Paris"]["monument"]
```



# SYMBOL

Cousin of `String` used for **keywords** of your code

[ruby-doc.org/core/Symbol.html](http://ruby-doc.org/core/Symbol.html)

# LET'S TAKE TWO CITIES

```
paris = {  
  "country" => "France",  
  "population" => 2211000  
}  
  
london = {  
  "country" => "England",  
  "population" => 8308000  
}
```

**country** and **population** are keys of the two hashes. They are **keywords** more than data.

# USE SYMBOLS FOR IDENTIFIERS

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

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

# SYMBOL PLAYS NICELY WITH HASH

```
paris = {  
  :country => "France",  
  :population => 2211000  
}
```

is equivalent to:

```
paris = {  
  country: "France",  
  population: 2211000  
}  
  
# New syntax does not change the way we read a key  
puts paris[:population]
```

As ruby developers, we prefer the latter syntax.

# SYMBOL VS STRING

Strings for **data**. Symbols for **keywords**.

```
# Text data => String
"Sebastien Saunier"
"seb@lewagon.org"
"ruby on Rails"
"Paris"

# Text identifiers => Symbol
:fullname
:email
:skill
:city
```

[Great answer on StackOverflow](#)

# **HASH AS LAST METHOD ARGUMENT**

# LET'S CODE AN HTML GENERATOR

```
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: "btn" })  
# => <a href='http://lewagon.org' class='btn'>Le Wagon</a>
```

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

**RAILS WILL USE SIMILAR METHODS**



# 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: [GitHub Api: /users/ssaunder](#)

**YOUR TURN!**