Hash

Outline

1. Introduction
2. Creating hash
3. Accessing hash
4. Converting to hash
5. Equality hashes
6. Element assignment
7. Iterating over hash
8. Other methods

1. Introduction

- A Hash is a collection of **key-value** pairs
- It is similar to an Array, except that indexing is done via arbitrary keys of any object type, not an integer index
- Hashes enumerate their values in the order that the corresponding keys were inserted
- Hashes have a *default value* that is returned when accessing keys that do not exist in the hash. By default, that value is ***nil***

2. Creating hash

- Using **new** class method
  - h = Hash.new #=> {}
  - h1 = Hash.new "a" #=> puts h1 ⇒ {} (default value is ***nil***)
  - h2 = Hash.new "a": 1 #=> puts h2 ⇒ {}
- Using the literal
  - h = Hash["a": 100, "b": 200] #=> puts h ⇒ {:a=>100, :b=>200}
  - h1 = Hash[[["a", 100], ["b", 200]]] #=> puts h2 ⇒ {:a=>100, :b=>200}

3. Accessing hash

```
h = Hash["a": 100, "b": 200]
```

- ➔ h[:a] #=> 100
- ➔ h[:c] #=> *nil*
- ➔ h.keys #=> [:a, :b]
- ➔ h.values #=> [100, 200]

4. Converting to hash

Using **try_convert(obj)** return *hash* or *nil*

- ➔ Hash.try_convert {1=>2} #=> {1=>2}
- ➔ Hash.try_convert "1=>2" #=> nil

5. Equality hashes

Operator: **==**, **>**, **<**, **>=**, **<=** ⇒ return **true/false**

```
h = Hash["a": 100, "b": 200, "c": 300]

h1 = Hash["a": 100, "b": 200, "c": 300, "d": 400]

h2 = Hash["b": 200, "c": 300, "a": 100]
```

- ➔ h == h1 #=> false
- ➔ h == h2 #=> true
- ➔ h1 == h2 #=> false
- ➔ h > h1 #=> false
- ➔ h1 > h #=> true

6. Element assignment

```
h = {"a": 100, "b": 200}
```

```
➔  h["a"] = 10          #=> h ⇒ {"a"=>10, "b"=>200}
➔  h["c"] = 300         #=> h ⇒ {"a"=>10, "b"=>200, "c"=> 300}
➔  h.store "d", 400     #=> h ⇒ {"a"=>10, "b"=>200, "c"=> 300, "d"=>400}
```


7. Interating over hash

- each {| key, value | block}
  - ➔ h.each {|key, value| puts "#{key} is #{value}"}
- each_key {| key | block}
  - ➔ h.each_key {|key| puts key}
- each_value {| value | block}
  - ➔ h.each_value {|value| puts value}
- ...

8. Other methods

- compact (!)
- any?
- empty?
- include?
- length
- merge (!)
- has_key?
- reject (!)
- size
- shift
- has_value?
- select (!)
- ...

References

- http://zetcode.com/lang/rubytutorial/hashes/
- http://ruby-doc.org/core-2.4.1/Hash.html

Thank you for listening!