Sun*

Hash

Outline

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

Sun*

1. Hash 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. That value is ***nil*** by default.

2. Creating hash

❖ Using **new** class method

```
➔    h = Hash.new                           # => {}
```

❖ Using the literal

```
➔    h = Hash["a": 100, "b": 200]           # => puts h ⇒ {:a=>100, :b=>200}

➔    h1 = {a: 200, b: 300}                  # => puts h1 ⇒ {:a=>200, :b=>300}
```

3. Accessing hash

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

➔    h[:c]                        # => nil

➔    h.keys                       # => [:a, :b]

➔    h.values                     # => [100, 200]
```

Sun*

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
```

Sun*

5. Equality hashes

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

```ruby
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]
h3 = Hash["a": 100, "b": 200, "c": 400]

puts "h == h1 #=> #{h == h1}"
puts "h == h2 #=> #{h == h2}"
puts "h1 == h2 #=> #{h1 == h2}"

puts "h > h1 #=> #{h > h1}"
puts "h1 > h #=> #{h1 > h}"
puts "h1 != h #=> #{h1 != h}"

puts "h > h3 #=> #{h < h3}"
puts "h <= h3 #=> #{h <= h3}"
puts "h != h3 #=> #{h != h3}"
```

```
#Result
h == h1 #=> false
h == h2 #=> true
h1 == h2 #=> false
h > h1 #=> false
h1 > h #=> true
h1 != h #=> true
h > h3 #=> false
h <= h3 #=> false
h != h3 #=> true
```

Sun*

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. Iterating 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. Except

Returns a copy of self with entries removed for specified keys.

```
➔    h = { a: 1, b: 2, c: 3 }
➔    p h.except(:a) #=> {:b=>2, :c=>3}
```

Sun*

9. Other hash methods

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

References

- http://ruby-doc.org/core-3.1.0/Hash.html
- http://zetcode.com/lang/rubytutorial/hashes/
- https://github.com/awesome-academy/RubyExample_TFW

Sun*

Question & Answer?

Sun*