Cheatography

# Ruby Enumerable Quick Ref Cheat Sheet

by dwapi via cheatography.com/43665/cs/13023/

## Sorting

**partition:** Returns two arrays, the first containing the elements which the block evaluates to true, the second containing the rest.

```
(1..6).partition { |v| v.even? }  # => [[2, 4, 6], [1, 3, 5]]
```

**sort_by:** Returns array sorted by the return value of the block.

```
[apple pear fig].sort_by {|word| word.length}  # => [fig, pear, apple]
```

**max and min:** Returns max/min element based on sorting by { |a, b| a <=> b }

```
[fish dog horse].max { |a, b| a.length <=> b.length }  # => "horse"
[fish dog horse].max(2) {|a, b| a.length <=> b.length  # => [horse, fish]
```

**max_by and min_by:** Returns element with max/min block return value

```
[fish dog horse].max_by { |x| x.length }  # => "horse"
[fish dog horse].max_by(2) {|x| x.length }  # => [horse, fish]
```

## Searching

**select and select!** Returns array of all elements where block returns true

```
[1,2,3,4,5].select { |num| num.even? }  # => [2, 4]
```

**reject and reject!** Returns array of all elements where block returns FALSE **

```
[1, 2, 3, 4, 5].reject { |num| num.even? }  # => [1, 3, 5]
```

**grep:** without block

```
[1, 'a', 2, 'b'].grep(Integer)  # => [1,2]
[dog cat tree doggie].grep(/dog/)  # => [dog, doggie]
```

**grep:** With Block

```
['a', 1, 2, 'b'].grep(String, &:upcase) # find strings & upcase  #=> [A, B]
```

**index:** Returns the index of the first object == to value

```
[ "a", "b", "c" ]index("b")  # => 1
```

## Iterators

## Iterators (cont)

**cycle** repeats contents

```
["a", "b", "c"].cycle { |x| puts x } # print, a, b, c, a, b, c,.. forever.
["a", "b", "c"].cycle(2) { |x| puts x } # print, a, b, c, a, b, c.
```

## Misc

**sample** Chose a random element or n random elements from array

```
[1,2,3,4,5,6].sample(3) # => [5, 6, 2]
```

**unshift** Add to front of array

```
[2,3,4].unshift(1) # => [1, 2, 3, 4]
```

**shift** deletes first item from arr

```
[1, 2, 3, 4, 5].shift # => [2, 3, 4, 5]
```

**pop** remove the last x items

```
[1, 2, 3, 4, 5].pop(2) # => [4, 5]
```

**delete** delets item passed

```
[2, 3, 4, 8].delete(8) # => [2, 3, 4]
```

**clear** Remove all elements from array

```
[2,3,4].clear # => []
```

**compact** remove nils from array

```
[2,3,nil,4, nil, 8, nil].compact # => [2, 3, 4, 8]
```

**uniq** removes duplicates from arr

```
[1, 2, 3, 4, 5, 1, 2].uniq # => [1, 2, 3, 4, 5]
```

**reduce or inject:** Iterate over a collection of data and perform proc/block

```
(5..10).reduce(:+) # => 45
longest = [cat sheep bear].inject do |memo, word|
  memo.length > word.length ? memo : word
end
longest # => "sheep"
```

## Hash Methods

**reverse_each** Same as each but in reverse

```
[dog cat abc].reverse_each { |word| str += "#{word} " }   => abc
cat dog
```

**each_cons** Iterates the given block for each array of consecutive elements.

```
(1..10).each_cons(3) { |a| p a }
#outputs:
[1, 2, 3]
[2, 3, 4]
[3, 4, 5] #etc...
```

**each_slice** terates the given block for each slice of elements.

```
(1..10).each_slice(3) { |a| p a }{
# outputs below
[1, 2, 3]
[4, 5, 6]
[7, 8, 9] #etc...
```

**delete_if / keep_if** Deletes/keeps every key-value pair from hsh for which block evaluates to true.

```
h.delete_if {|key, value| key >= "b" }     # => {"a"=>100}
```

**has_key?(key)** Returns true if the given key is present in hsh.

```
h.has_key?("a")     # => true
```

**has_value?(value)** Returns true if the given value is present in hsh.

```
h.has_value?(200)     # => true
```

**invert** Returns a new hash created by using hsh's values as keys, and the keys as values.

```
h.invert     # => {100=>"a", 200=>"b", 300=>"c"}
```

**reject / reject!** Same as Hash#delete_if, but works on (and returns) a copy of the hsh. Equivalent to hsh.dup.delete_if.

```
h.reject {|key, value| key >= "b" }     # => {"a"=>100, "c"=>300}
```

**select /select!** Returns a new hash consisting of entries for which the block returns true.

```
h.select {|k,v| k > "a"}     # => {"b" => 200, "c" => 300}
```

**Examples above use h = { "a" => 100, "b" => 200, "c" => 300 }**

By **dwapi**
cheatography.com/dwapi/

Published 4th October, 2017.
Last updated 17th October, 2017.


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com