

TechSlam.Net - Ruby array methods

Cheatsheet

Ruby version :

```
C:\>ruby -v
```

```
ruby 1.9.2p0 (2010-08-18) [i386-mingw32]
```

1. &

```
[ 1, 1, 3, 5 ] & [ 1, 2, 3 ] # => [1, 3]
```

This will return a new array containing elements common to both the arrays and with no duplicates.

2. *

```
[ 1, 2, 3 ] * 3 # => [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Returns an array built by concatenating n copies of the array

```
[ 1, 2, 3 ] * "--" # => "1--2--3"
```

Returns a string.

3. +

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

Returns a new array by concatenating the two arrays.

4. -

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

Removes the elements from the first array, that is in second array and returns a new array.

5. <<

```
[ 1, 2 ] << "c" << "d" << [ 3, 4 ] # => [1, 2, "c", "d", [3, 4]]
```

Returns the same array by pushing the objects to the end of the array.

6. ==

```
[ "a", "c" ] == [ "a", "c", 7 ] # => false
```

```
[ "a", "c", 7 ] == [ "a", "c", 7 ] # => true
```

```
[ "a", "c", 7 ] == [ "a", "d", "f" ] # => false
```

It returns either 'true' or 'false' making a comparison of the two arrays. If both the arrays have same elements and same length, it will return 'true' else it will return 'false'.

7. |

```
[ "a", "b", "c" ] | [ "c", "d", "a" ] # => ["a", "b", "c", "d"]
```

This will return a new array joining the two arrays and removing the duplicates.

8. at

```
a = [ "a", "b", "c", "d", "e" ]
```

```
a.at(0) # => "a"
```

```
a.at(-1) # => "e"
```

```
a.at(5) # => nil
```

Returns the element at index supplied.

9. clear

```
a = [ "a", "b", "c", "d", "e" ]
```

```
a.clear # => []
```

This will remove all the elements from the array.

10. collect!

```
a = [ "a", "b", "c", "d" ]
```

```
a.collect! { |x| x + "$" } # => ["a$", "b$", "c$", "d$"]
```

```
a # => ["a$", "b$", "c$", "d$"]
```

This will loop through the each element of the array and will replace the element with the value returned by the block.

11. compact

```
[ "a", nil, "b", nil, "c", nil ].compact # => ["a", "b", "c"]
```

Returns a copy of array with all nil elements removed.

12. concat

```
[ "a", "b" ].concat( ["c", "d"] ) # => ["a", "b", "c", "d"]
```

Appends the elements in second array to the first array.

13. count

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

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

```
[1, 2, 3, 4].count { |obj| obj > 2 } # => 2
```

Returns the count of the elements in the array which is either equal to the parameter passed or satisfies the block condition.

14. delete

```
a = [ "a", "b", "b", "b", "c" ]
```

```
a.delete("b") # => "b"
```

```
a # => ["a", "c"]
```

```
a.delete("z") # => nil
```

```
a.delete("z") { "not found" } # => "not found"
```

Removes the element from the array which is equal to the

parameter passed.

15. each

```
a = [ "a", "b", "c" ]  
a.each { |x| print x, " -- " }  
result :  
a -- b -- c --
```

Iterates through the array and returns the value from the block.

16. each_index

```
a = [ "a", "b", "c" ]  
a.each_index { |x| print x, " -- " }  
result :  
0 -- 1 -- 2 --
```

Here instead of array elements, the index for the element is being passed to the block

17. empty?

```
[].empty? # => true  
[ 1, 2, 3 ].empty? # => false
```

Returns true if array array contains no elements.

18. join

```
[ "a", "b", "c" ].join # => "abc"  
[ "a", "b", "c" ].join("-") # => "a-b-c"
```

Returns a string created by converting each each element of the array to a string and con-catenating them, separated each by separator.

19. last

```
[ "w", "x", "y", "z" ].last # => "z"
```

```
[ "w", "x", "y", "z" ].last(1) # => ["z"]
```

```
[ "w", "x", "y", "z" ].last(3) # => ["x", "y", "z"]
```

Returns the last element, or last count elements, of array.

20. length

```
[ 1, nil, 3, nil, 5 ].length # => 5
```

Returns the number of elements in array.

21. pop

```
a = %w{ f r a b j o u s }
```

```
a.pop # => "s"
```

```
a # => ["f", "r", "a", "b", "j", "o", "u"]
```

```
a.pop(3) # => ["j", "o", "u"]
```

```
a # => ["f", "r", "a", "b"]
```

Removes the last element or last n elements from the array and returns them.

22. push

```
a = [ "a", "b", "c" ]
```

```
a.push("d", "e", "f") # => ["a", "b", "c", "d", "e", "f"]
```

Appends the given argument(s) to array.

23. replace

```
a = [ "a", "b", "c", "d", "e" ]
```

```
a.replace([ "x", "y", "z" ]) # => ["x", "y", "z"]
```

```
a # => ["x", "y", "z"]
```

Replaces the contents of an array with the new array.

24. reverse

```
[ "a", "b", "c" ].reverse # => ["c", "b", "a"]
```

```
[ 1 ].reverse # => [1]
```

Returns a new array using array's elements in reverse order.

25. shift

```
args = [ "-m", "-q", "-v", "filename" ]
```

```
args.shift # => "-m"
```

```
args.shift(2) # => ["-q", "-v"]
```

```
args # => ["filename"]
```

Removes the first element from the array or first n elements from the array and returns them. Also it will shift all other elements down by one.

26. unshift

```
a = [ "b", "c", "d" ]
```

```
a.unshift("a") # => ["a", "b", "c", "d"]
```

```
a.unshift(1, 2) # => [1, 2, "a", "b", "c", "d"]
```

prepends the object to the array. Also it will shift all other elements up by one.

27. shuffle

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

Returns an array containing the elements of array in random order.

28. sort

```
a = [ "d", "a", "e", "c", "b" ]
```

```
a.sort! # => ["a", "b", "c", "d", "e"]
```

```
a # => ["a", "b", "c", "d", "e"]
```

Returns an array containing the elements of array in sorted

order.

29. values_at

```
a = %w{ a b c d e f }  
a.values_at(1, 3, 5) # => ["b", "d", "f"]  
a.values_at(1, 3, 5, 7) # => ["b", "d", "f", nil]  
a.values_at(-1, -3, -5, -7) # => ["f", "d", "b", nil]  
a.values_at(1..3, 2...5) # => ["b", "c", "d", "c", "d", "e"]
```

Returns an array containing the elements in array corresponding to the given selector(s).

30. uniq

```
a = [ "a", "a", "b", "b", "c" ]  
a.uniq # => ["a", "b", "c"]
```

Returns a new array by removing duplicate values in array.

*No rights reserved: Share it: Enjoy_Ruby_Programming:

Mohammed Aslam
Cofounder at [BukLuv](#)
aslam9895[at]gmail.com

twitter.com/tweeslam
github.com/techslam
techslam.net