# Array

### basic examples

numbers = [1,2,3,4]

strings = ["foo", "baz", "boo"]

empty\_ar = [], empty\_ar = Array.new

#init an 8 x 8 multi-dim array this way.

game\_board = Array.new(8) { Array.new(8) }

ar[i]      #=> ith element

ar[-2]   #=> 7, 2nd from end

ar[1..3]        #=> [6,7,8]

ar[2,2]        #=> [7,8]

### << shovel

Appends to the end of the array.

let: ar = [ 5, 6, 7 ]

ar << 9

Returns: [ 5, 6, 7, 9 ]

let: ar = [ 5, 6, 7 ]

ar << [ 1, 2, 3 ]

Returns: [ 5, 6, 7, 8, [ 1, 2, 3 ] ]

### **+ plus**

Same as .concat(element)

Combines (concatenates) the elements of arrays.

[ 5, 6, 7 ] + [ 1, 2, 3 ]

Returns: [ 5, 6, 7, 9, 1, 2, 3]

### **- minus, subtract**

Removes all occurrences of the elements of the second operand.

[ 5, 6, 7, 6, 7 ] - [ 6, 7 ]

Returns: [ 5 ]

### **\* string multiply**

Combines copies of the array into a new array.

let: ar = [ "Ouch!" ]

ar \* 3

Returns: "Ouch! Ouch! Ouch!"

### **all?**

### 

Returns true if all elements of the array pass the block's test.

let: ar = [ 5, 6, 7 ]

ar.all?{ |el| el > 4 }

Returns: true

### 

### **any?**

Returns true if any element of the array passes the block's test.

let: ar = [ 5, 6, 7 ]

ar.any?{ |el| el > 6 }

Returns: true

### **count**

### **count ( element )**

### **count { block }**

Without parameter: returns the length of the array.

With parameter: returns the number of times element appears in the array.

With block: returns the number of elements that pass the block test.

let: ar = [ 5, 6, 7, 7 ]

ar.count

Returns: 4

ar.count( 7 )

Returns: 2

ar.count{ |el| el >= 6 }

Returns: 3

### **delete ( element )**

Returns the element if it was deleted from the array, otherwise nil.

The array itself is modified.

let: ar = [ 5, 6, 7, 7 ]

ar.delete( 7 )

Returns: 7

ar is now [ 5, 6 ]

### **delete\_at ( index )**

Deletes the element at index.

The array itself is modified.

Returns the deleted element, or nil if there was no index'th element.

let: ar = [ 5, 6, 7, 7 ]

ar.delete\_at( 1 )

Returns: 6

ar is now [ 5, 7, 7 ]

### **delete\_if { block }**

Delete elements which pass the block test.

Returns the original array which has been modified.

let: ar = [ 5, 6, 7, 7 ]

ar.delete\_if{ |el| el >= 6 }

Returns: [5]

ar is now [5]

### **each { block }**

Performs the block on each element of the array. Original array is not modified, thus it's is all about the side effects.

Returns the unmodified array.

let: ar = [ "Joe", "Fred", "Sally" ]

ar.each{ |el| puts "Hi " + el +"!"}

StdOut:

Hi Joe!

Fred!

Hi Sally!

Returns: [ "Joe", "Fred", "Sally" ]

(the unmodified array)

each\_with\_index

find

find\_all

find\_index

first

inject

map

drop

flatten

include?

index

insert

join

length

max

min

minmax

pop

push

select

size

shift

shuffle

take

uniq