

Core Ruby 41st Batch

Home ► PORPC101-41C ► 19 April - 25 April ► Some useful Ruby methods

Navigation

Home

■ My home

Site pages

My profile

Current course

PORPC101-41C


Participants


General

5 April - 11 April


12 April - 18 April


19 April - 25 April

 Week 3: Tutorial


 Week 3:
Exercises

 Week 3: Forum

 FizzBuzz Forum
(Special
Discussion Area)

 **Some useful
Ruby methods**

 Recipe

 *PARALLEL
TRACK* Create a
simple webpage

 Week 3: Quiz

26 April - 2 May

3 May - 9 May

10 May - 16 May

17 May - 23 May

24 May - 30 May

31 May - 6 June

7 June - 13 June

Some useful Ruby methods

inject

The **inject** method (defined in the module **Enumerable**) lets you accumulate a value across the members of a collection. For example, you can sum all the elements in an array, and find their product, using code such as:

```
puts [1,3,5,7].inject(0) {|sum, element| sum+element}
```

```
puts [1,3,5,7].inject(1) {|product, element| product*element}
```

inject works like this: the first time the associated block is called, sum is set to inject's parameter (in this case 0) and element is set to the first element in the collection. The second and subsequent times the block is called, sum is set to the value returned by the block on the previous call (respectively 1, 4, 9). The final value of **inject** is the value returned by the block the last time it was called.

There's one final wrinkle: if **inject** is called with no parameter, it uses the first element of the collection as the initial value and starts the iteration with the second value. This means that we could have written the previous examples as:

```
puts [1,3,5,7].inject {|sum, element| sum+element}
```

```
puts [1,3,5,7].inject {|product, element| product*element}
```

clear

arr.clear -> arr

The **Array clear** method removes all elements from arr.

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

[My courses](#)

Settings

[Course administration](#)[My profile settings](#)

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

concat

arr.concat(other_array) -> arr

The **Array concat** method appends the elements in other_array to arr.

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

pop

arr.pop -> obj or nil

The **Array pop** method removes the last element from arr and returns it or returns nil if the array is empty.

```
a = [ "a", "m", "z" ]
```

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

```
puts a # => [ "a", "m" ]
```

empty?

arr.empty? -> true or false

The **Array empty?** method returns true if arr array contains no elements.

```
puts [].empty? # => true
```

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

IO.read(name, [length [, offset]]) -> string

The **IO.read** method opens the file, optionally seeks to the given offset, and then returns length bytes (defaulting to the rest of the file). **read** ensures the file is closed before returning.

A file testfile.txt contains:

```
"This is line one\nThis is line two\nThis is line
```

Example:

```
IO.read("testfile")
```

```
=begin
```

Output is:

```
"This is line one\nThis is line two\nThis is line three\n
```

```
And so on...\n"
```

```
=end
```

```
IO.read("testfile", 20) # => "This is line one\nThi"
```

```
IO.read("testfile", 20, 10) # => "ne one\nThis is line "
```

io.stat -> stat

Returns status information.

```
# Assuming that the file hellousa.rb exists
# new opens the file hellousa.rb in default "r" m
# and returns a new File object.
f = File.new("hellousa.rb")
s = f.stat
puts s.atime # => Sat Jan 26 15:54:52 +0530 2008
```

atime returns the last access time for this file as an object of class **Time**.

file.path -> filename

Returns the pathname used to create *file* as a string. An example:

```
# Assume the following folders exist,
# then puts File.new("E:/TalimBackup/rubyprograms
```

displays:

```
E:/TalimBackup/rubyprograms/smt.tmp
```

io.lineno -> int

Returns the current line number in io. The stream must be opened for reading.

io.rewind -> 0

Positions io to the beginning of input, resetting lineno to zero.

Create a file called *newtestfile.txt* whose contents are:

```
This is line one\n This is line two\n
```

Run this program:

```
f = File.new("newtestfile.txt")
puts f.readline # => "This is line one\n"
f.rewind # => 0
puts f.lineno # => 0
puts f.readline # => "This is line one\n"
```

The output is:

```
This is line one\n 0 This is line one\n
```

sleep(numeric=0) -> fixnum

The **Kernel sleep** method suspends the current thread for numeric seconds (which may be a Float with fractional seconds). Returns the actual number of seconds slept (rounded), which may be less than that asked for if the thread was interrupted. An argument of zero causes **sleep** to sleep forever.

The program:

```
puts Time.now
sleep 1.9 # roughly 2 seconds
puts Time.now
```

gives the output:

```
Thu Jan 31 10:39:48 +0530 2008
Thu Jan 31 10:39:50 +0530 2008
```

Note: The Enumerable mixin provides collection classes with traversal and searching methods and with the ability to sort. The class must provide a method `each`, which yields successive members of the collection. Ruby 1.9 adds a substantial number of methods to this module, as well as changing the semantics of many others.

Last modified: Saturday, 9 June 2012, 1:50 AM

You are logged in as [C. O'Keefe](#) ([Logout](#))



© 2008-2013 RubyLearning.org : A Ruby Learning Hub