

Ruby Cheatsheet

v.1 for Ruby 1.8.4

Types	Expressions	Variables
12345	if <i>expr</i> [then]	local
123.45	elsif <i>expr</i> [then]	@instance
1.23e-4	else	@class
0xFF00	end	CONSTANT
0b01100	unless <i>expr</i> [then]	
1..5	else	Operators and Precedence
1...5	end	::
,a'..'z'	<i>expr</i> if <i>expr</i>	[]
,a'...'z'	<i>expr</i> unless <i>expr</i>	**
,string <i>sq</i> '		+ - ! ~
,string <i>dq</i> "	case <i>expr</i>	* / %
,#{ <i>expr</i> }"	when comp	<< >>
,\t\r\n"	else	&
%q(string <i>sq</i>)	end	^
%Q(string <i>dq</i>)	while <i>expr</i> [do]	> >= < <=
%(string <i>dq</i>)	end	<=> == === !=
<<id string id	do	=~
:symbol	while <i>expr</i>	&&
/regex/opt	do
%r regex	until <i>expr</i>	= (+= -=)
[1, 2, 3]	for var in <i>expr</i> [do]	not
%w(1 2 3)	end	and or
%W(1 2 #{ <i>expr</i> })	<i>expr</i> .each [do]	
{1=>2, :s=>'v'}	end	Constants
	break next redo retry	__FILE__
		__LINE__
Exceptions	Module/Class	ENV
begin	module <i>Name</i>	ARGF
rescue ex =>	end	ARGV
var	class <i>Name</i>	
else	end	
ensure		
end		
StandardError	class <i>Name</i> < <i>Sup</i>	
ZeroDivisionError	end	
RangeError	class << <i>obj</i>	
SecurityError	end	
IOError	def <i>name</i> (<i>args</i> ...)	
IndexError	end	
RuntimeError	def <i>inst.name</i> (...)	
	end	
	public	
	protected	
	private	
	attr_reader	
	attr_writer	
	attr	
	attr_accessor	
	alias <i>new old</i>	

Predefined Variables	
\$!	Exception information
\$@	Array of backtrace
\$&	String of last match
\$'	String left of last match
\$'	Str right of last match
\$+	Last group of last match
\$N	Nth group of last match
\$~	Info about last match
\$=	Case insensitive flag
\$/	Input record separator
\$\	Output record separator
,\$	Output field separator
\$.	Line number of last file
\$>	Default output
\$_	Last input line of string
\$*	Command line args
\$0	Name of script
\$\$	Process number
\$"	Module names loaded
\$stderr	Standard error output
\$stdin	Standard input
\$stdout	Standard output

Regex	
.	all characters
[]	any single char in set
[^]	any single char not in set
*	zero or more
+	one or more
?	zero or one
	alteration
()	Group
^	Beginning of line or str
\$	End of line or string
{1,5}	1 to 5
\A	Beginning of a string
\b	Word boundary
\B	Non-word boundary
\d	digit, same as [0..9]
\D	Non-digit
\s	Whitespace
\S	Non-whitespace
\w	Word-character
\W	Non-word-character
\z	End of a string
\Z	End of string, before nl

Ruby arguments	
-c	Check
-d	Debug
-e	One Line
-h	Help
-n	gets loop
-rL	require L
-v	verbose
-w	warnings
-y	comp debug

Reserved Words	
alias	
and	
BEGIN	
begin	
break	
case	
class	
def	
defined?	
do	
else	
elsif	
END	
end	
ensure	
false	
for	
if	
in	
module	
next	
nil	
not	
or	
redo	
rescue	
retry	
return	
self	
super	
then	
true	
undef	
unless	
until	
when	
while	
yield	

Object

Obj#class -> class
Obj#freeze -> object
Obj#frozen? -> true or false
Obj#inspect -> string
Obj#is_a? (class) -> true or false
Obj#methods -> array
Obj#respond_to? (sym) -> true or false
Obj#to_s -> string

String

Str#[num, num/range/regex] -> str
Str#capitalize! -> string
Str#center (int [,str]) -> str
Str#chomp! ([str]) -> str
Str#count -> integer
Str#delete! ([string]) -> string
Str#downcase! -> string
Str#each ([str]) do str ... end
Str#each_line do line ... end
Str#gsub! (rgx) do match ... end
Str#include? (str) -> true / false
Str#index (str/reg [,off]) -> int
Str#insert (int, string) -> string
Str#length -> integer
Str#ljust (int [,padstr]) -> str
Str#rindex (str/reg [,off]) -> int
Str#rjust (int [,padstr]) -> str
Str#scan (rgx) do match ... end
Str#split (string) -> array
Str#strip! -> string
Str#sub! (rgx) do match ... end
Str#swapcase! -> string
Str#to_sym -> symbol
Str#tr! (string, string) -> string
Str#upcase! -> string

Kernel

block_given?
eval (str [,binding])
raise (exception [,string])
fork do ... end => fixnum or nil
proc do ... end => proc
print (obj)
warn (msg)

Array

Array::new (int [,obj]) -> array
Array#clear
Array#map! do x ... end
Array#delete (value) -> obj or nil
Array#delete_at (index)-> obj or n
Array#delete_if do x ... end
Array#each do x ... end
Array#flatten! -> array
Array#include? (value) -> t or f
Array#insert (idx, obj...)-> array
Array#join ([string]) -> string
Array#length -> integer
Array#pop -> obj or nil
Array#push (obj...) -> array

Hash

Hash#clear
Hash#delete (key) -> obj or nil
Hash#delete_if do k, v ... end
Hash#each do k, v ... end
Hash#has_key? (k) -> true or false
Hash#has_value? (v) -> t or f
Hash#index (value) -> key
Hash#keys -> array
Hash#length -> integer
Hash#select do k, v ... end -> array
Hash#values -> array

Test::Unit

assert (boolean [,msg])
assert_block (message) do ... end
assert_equal (expected, actual [,msg])
assert_in_delta (exp, act, dlt [,message])
assert_kind_of (klass, object [,msg])
assert_match (pattern, string [,msg])
assert_nil (object [,msg])
assert_no_match (pattern, string [,msg])
assert_not_equal (expected, actual [,msg])
assert_not_nil (object [,msg])
assert_not_same (expected, actual [,msg])
assert_respond_to(obj, method [,msg])
assert_same (expected, actual [,msg])

File

File#new (path, modestring)-> file
File#new (path, modestring) do file ... end
File#open (path, modestring) do file ... end
File#exist? (path) -> t or f
File#basename (path [,suffix]) -> string
File#delete (path, ...)
File#rename (old, new)
File#size (path) -> integer
r Read-only, from beginning
r+ Read-write, from beginning
w Write-only, trunc. / new
w+ Read-write, trunc. / new
a Write-only, from end / new
a+ Read-write, from end / new
b Binary (Windows only)

Dir

Dir[string] -> array
Dir::chdir ([string])
Dir::delete (string)
Dir::entries (string) -> array
Dir::foreach (string) do file ... end
Dir::getwd -> string
Dir::mkdir (string)
Dir::new (string)
Dir::open (string) do dir .. end
Dir#close
Dir#pos -> integer
Dir#read -> string or nil
Dir#rewind

DateTime

DateTime::now
DateTime::parse (str)
DateTime::strptime (str, format)
DateTime#day
DateTime#hour
DateTime#leap?
DateTime#min
DateTime#month
DateTime#sec
DateTime#yday
DateTime#year