RUBY PREDEFINED VARIABLES

Ruby's predefined variables affect the behavior of the entire program, so their use in libraries is not recommended.

The values in most predefined variables can be accessed by alternative means.

Following table lists all the Ruby's predefined variables.

Variable Name	Description
$!	The last exception object raised. The exception object can also be accessed using => in *rescue* clause.
$@	The *stack backtrace* for the last exception raised. The *stack backtrace* information can retrieved by Exception#backtrace method of the last exception.
$/	The input record separator *newlinebydefault*. *gets, readline,* etc., take their input record separator as optional argument.
$\	The output record separator *nilbydefault*.
$,	The output separator between the arguments to print and Array#join *nilbydefault*. You can specify separator explicitly to Array#join.
$;	The default separator for split *nilbydefault*. You can specify separator explicitly for String#split.
$.	The number of the last line read from the current input file. Equivalent to ARGF.lineno.
$<	Synonym for ARGF.
$>	Synonym for $defout.
$0	The name of the current Ruby program being executed.
$$	The process pid of the current Ruby program being executed.
$?	The exit status of the last process terminated.
$:	Synonym for $LOAD_PATH.
$DEBUG	True if the -d or --debug command-line option is specified.
$defout	The destination output for *print* and *printf* (*$stdout* by default).
$F	The variable that receives the output from *split* when -a is specified. This variable is set if the -a command-line option is specified along with the -p or -n option.
$FILENAME	The name of the file currently being read from ARGF. Equivalent to ARGF.filename.
$LOAD_PATH	An array holding the directories to be searched when loading files with the load and require methods.
$SAFE	The security level 0 --> No checks are performed on externally supplied *tainted* data. *default*

> 1 --> Potentially dangerous operations using tainted data are forbidden.
>
> 2 --> Potentially dangerous operations on processes and files are forbidden.
>
> 3 --> All newly created objects are considered tainted.
>
> 4 --> Modification of global data is forbidden.

$stdin	Standard input *STDINbydefault*.
$stdout	Standard output *STDOUTbydefault*.
$stderr	Standard error *STDERRbydefault*.
$VERBOSE	True if the -v, -w, or --verbose command-line option is specified.
$- x	The value of interpreter option -x $x = 0, a, d, F, i, K, l, p, v$. These options are listed below
$-0	The value of interpreter option -x and alias of $/.
$-a	The value of interpreter option -x and true if option -a is set. Read-only.
$-d	The value of interpreter option -x and alias of $DEBUG
$-F	The value of interpreter option -x and alias of $;.
$-i	The value of interpreter option -x and in in-place-edit mode, holds the extension, otherwise nil. Can enable or disable in-place-edit mode.
$-I	The value of interpreter option -x and alias of $:.
$-l	The value of interpreter option -x and true if option -lis set. Read-only.
$-p	The value of interpreter option -x and true if option -pis set. Read-only.
$_	The local variable, last string read by gets or readline in the current scope.
$~	The local variable, *MatchData* relating to the last match. Regex#match method returns the last match information.
$n(1, 2, 3...)	The string matched in the nth group of the last pattern match. Equivalent to m[n], where m is a *MatchData* object.
$&	The string matched in the last pattern match. Equivalent to m[0], where m is a *MatchData* object.
$`	The string preceding the match in the last pattern match. Equivalent to m.pre_match, where m is a *MatchData* object.
$'	The string following the match in the last pattern match. Equivalent to m.post_match, where m is a MatchData object.
$+	The string corresponding to the last successfully matched group in the last pattern match.