

Perl 5 Constants and Variables

Perl Sigils	Sigil	Examples	Meaning	Extra Info	
Scalar	\$	\$foo \$days[28] \$days{'Feb'} \${days} \$Dog::days \$Dog'days \$#days \$days->[28] \$days[0][2] \$d{99}{ 'Feb' } \$d{99, 'Feb' }	Simple scalar value 29 th element of array @days Value associated with the <i>Feb</i> key of hash %days Same as \$days, but unambiguous before alphanumerics. Useful inside strings for interpolation of variables followed by other letters. The \$days variable inside the Dog package. Same as above. However this is an archaic use of the single quote. Last index of array @days. 29 th element of array pointed to by reference \$days. Multi-dimensional array Multi-dimensional hash Multi-dimensional hash emulation		
Array	@	@days @days[3,4,5] @days[3..5] @days{'J','F'}	Array containing (\$days[0], \$days[1], ... \$#days[\$#days]) . Array slice containing (\$days[3], \$days[4], \$days[5]) . Array slice containing (\$days[3], \$days[4], \$days[5]) . Hash slice containing (\$days{'J'}, \$days{'F'}) .		
Hash/associative array	%	%days	Associative array (hash): keys-value pairs. Can be initialized as: • %days = (Jan => 31, Feb => \$leap? 29 : 28, ...) • %days = ("Jan, 31, 'Feb', \$leap? 29 : 28, ...)		
Subroutine	&	&foo	& is needed to create reference to subroutine.		
Typeglob	*	*foo		See: Advanced Perl Programming , 1st Edition Section 3.2	
Scalar values			Numeric literals examples	Useful related builtin functions	
• numeric:	• integer : using the system's native format. • bigint - transparent big integer support. • bignum - transparent big number support. • floating-point : using the system's native format. • bigrat - transparent big rational number support.		my \$x = 12345; # integer my \$x = 12345.67; # floating point my \$x = 6.02e23; # scientific notation my \$x = 4_294_967_296; # underline for legibility my \$x = 0377; # octal my \$x = 0xffff; # hexadecimal my \$x = 0b1100_0010; # binary	• oct • hex • POSIX::ceil • POSIX::floor • abs	
• string	• double-quoted strings: perform backslash and variable interpolation of expression that begin with \$ (a scalar) or @ (an array). Hashes cannot be interpolated. • single-quote strings: only perform \ ' and \\ substitution (to ' and \ respectively), nothing else.				
• Quote constructs	Customary	Generic	Meaning	Interpolates?	Notes
See: • Strings in Perl: quoted, interpolated and escaped	' '	q//	Literal string	No	• Not all characters can be used as the / separator. { }, () and < > can also be used. • You can use whitespace between the quote specifier and its initial bracketing character: my \$chuck_of_code = q { if (\$condition) { print "Salut!"; } }; • It's also possible to write: s<foo>(bar) and tr(a-f)[A-F] as well as: tr(a-f) [A-F]; • Array variables are interpolated by joining all elements with the separator specified by the \$* special variable (\$LIST_SEPARATOR) .
	"""	qq//	Literal string	Yes	
	^^	qx//	Command execution	Yes	
	()	qw//	World list	No	
	//	m//	Pattern match	Yes	
	s///	s///	Pattern substitution	Yes	
	tr///	y///	Character translation	No	
	""	qr//	Regular expression	Yes	
• Character escapes	\a	Alert (bell)	\e	ESC character	\N{LATIN SMALL LETTER E WITH ACUTE} é \N{ U+E9 } é
	\b	Backspace	\033	ESC in octal	
	\e	ESC character	\o{33}	ESC in octal	
	\f	Form feed	\x7f	DEL in hexadecimal	
	\n	Newline (usually LF)	\x{263a}	Character number 0x263A	
	\r	Carriage return (Usually CR)	\cC	Control-C	
	\t	Horizontal tab			
• translation escapes	\u	Force next character to titlecase	\U	Force all following characters to uppercase. Ends at \E	\E Ends \U, \L, \F or \Q
	\l	Force next character to lowercase	\L	Force all following characters to lowercase. Ends at \E	
			\F	Force all following characters to fold case. Ends at \E	
			\Q	Backslash all following non alphanumeric characters. Ends at \E	
• bareword	In Perl, a <i>bareword</i> refers to a sequence of characters suitable for an identifier. It's not quoted. By default Perl allows barewords to behave like strings. • This is not allowed when any of use strict; or use strict "subs"; or use v5.12; is specified.				
• Here documents • Here docs @ Perl maven	Perl here-documents are a form of line oriented quoting. There are several forms of here documents, where the identifier (like EOF used below, but can be any word) must be placed at the beginning of the terminating line: • Default : <<EOF; Supports variable interpolation. • Double quotes: <<"EOF"; Supports variable interpolation. Can also be written with whitespace as in <<"EOF"; ; • Single quotes: <<'EOF'; Does not support interpolation. Can also be written with whitespace as in <<'EOF'; ; • backticks: <<'EOF'; Execute commands in a shell and return text printed on stdout. Can also be written with whitespace as in <<'EOF'; ; • indented: <<~EOF; Allows indenting the here-doc string. Can also use the ~ with the other forms: <<~\EOF, <<~"EOF" , <<~"EOF" , <<~'EOF' • They can also be stacked and text can be transformed. See the documentation.				
• Perl Regexp info, cheatsheets & regexp testers	• Regexp Tutorial • Learn PCRE in X minutes		• PCRE cheatsheet		• Debuggex regexp tester • regex101 • RegEx Pal
Perl Constants					
• Perl pragma to declare constants . ⚠ But be aware that these are still not read-only, that they inject sub-routines and have several limitations. Read the doc!! • CPAN modules for defining constants by Neil Bowers . Of particular interest: Const::Fast and Attribute::Constant for efficient read-only constants.					
Perl Variables Names		Scalar Naming Conventions		Array Naming Conventions	
👉 Case is significant in all names.	• Local variables: \$lowercase • Global variables: \$Title_Case • Constants: \$UPPER_CASE • All variables: words separated by underscores.		Similar conventions, except that array names should be plural . • @locals • @Global_Arrays • @CONSTANT_ARRAYS		
Perl Special Variables					
• Perl Variables	👉 To get information about a Perl special variable from the command line use the perlloc -v command. • To get information about \$< use: perlloc -v '\$<'				
• General variables					
default input and pattern searching space	• \$ARG • \$_		subroutine parameters		• @ARG • @_
list separator	• \$LIST_SEPARATOR • \$"		Subscript separator for multidimensional array emulation		• \$\$SUBSCRIPT_SEPARATOR • \$SUBSEP • \$;
Name of executed program	• \$PROGRAM_NAME • \$0		Name used to execute the current copy of Perl		• \$EXECUTABLE_NAME • \$^X
Perl process ID	• \$PROCESS_ID • \$PID • \$\$				
Process real GID	• \$REAL_GROUP_ID • \$GID • \$(Process effective GID		• \$EFFECTIVE_GROUP_ID • \$EGID • \$)




Process real UID	<ul style="list-style-type: none">\$REAL_USER_ID\$UID\$<		Process effective UID	<ul style="list-style-type: none">\$EFFECTIVE_USER_ID\$\$EUID\$>	
Special variables in sort	<ul style="list-style-type: none">\$a\$b		Example: by default Perl sort function sorts strings. Pass a sorting function that uses the <=> equality operator to force numerical comparisons: <code>@sorted = sort { \$a <=> \$b } @unsorted;</code>		
Current environment	%ENV		Environment variable accessed as an associative array (a hash). <ul style="list-style-type: none">See: Perl: How to access shell environment variables through Perl associative arrays.		
Perl interpreter revision, version and subversion	<ul style="list-style-type: none">\$OLD_PERL_VERSION\$]		Perl interpreter revision, version and subversion	<ul style="list-style-type: none">\$PERL_VERSION\$^V	
Maximum file descriptor	<ul style="list-style-type: none">\$SYSTEM_FD_MAX\$^F				
Fields of each line when auto-split mode is on.	@F				
Include Directories	@INC	Included filenames	%INC	Hook localization (?)	\$INC
inplace-edit extension value	<ul style="list-style-type: none">\$INPLACE_EDIT\$^I				
Package's class parent classes	@ISA				
Emergency memory pool	\$^M				
Maximum block nesting	\${^MAX_NESTED_EVAL_BEGIN_BLOCKS}				
Name of OS where this Perl was built	<ul style="list-style-type: none">\$OSNAME\$^O				
Signal handlers	%SIG				
Coderefs for various perl keywords	%{^HOOK}				
Time when program began running	<ul style="list-style-type: none">\$BASETIME\$^T				
<ul style="list-style-type: none">Variables related to regular expressions					
captured sub-patterns	\$<digit>(\$1, \$2, ...)				
Capture buffer content	@{^CAPTURE}				
String matched	<ul style="list-style-type: none">\$MATCH\$&		String matched (compiled regexp)	\${^MATCH}	
String preceding match	<ul style="list-style-type: none">\$PREMATCH\$'		String preceding match (compiled regexp)	\${^PREMATCH}	
String following match	<ul style="list-style-type: none">\$POSTMATCH\$'		String following match (compiled regexp)	{^POSTMATCH}	
Last capture group	<ul style="list-style-type: none">\$LAST_PAREN_MATCH\$+		Most recently closed capture group	<ul style="list-style-type: none">\$LAST_SUBMATCH_RESULT\$^N	
Match capture key values	<ul style="list-style-type: none">%{^CAPTURE}%LAST_PAREN_MATCH%+				
Match start offsets	<ul style="list-style-type: none">@LAST_MATCH_START@-	Match ends offsets	<ul style="list-style-type: none">@LAST_MATCH_END@+	Named captured groups	<ul style="list-style-type: none">%{^CAPTURE_ALL}%-
Last successful pattern	\${^LAST_SUCESSFUL_PATTERN}				
Result of last successful regexp assertion	<ul style="list-style-type: none">\$LAST_REGEXP_CODE_RESULT\$^R				
Maximum regexp nested group	\${^RE_COMPILE_RECURSION_LIMIT}				
regexp debug flag	\${^RE_DEBUG_FLAG}				
regexp internal optimization/memory	\${^RE_TRIE_MAXBUF}				
<ul style="list-style-type: none">Variables related to file handles	See also: <u>Perl File Handles</u>				
Name of current file read from <>	\$ARGV	Command line arguments of the script	@ARGV	Number of arguments minus one	\$#ARGV
Special file handle that iterates over command-line filenames in @ARGV	ARGV	Special file handle that points to currently open output file when doing edit-in-place processing	ARGVOUT		
Output field separator for the print operator	<ul style="list-style-type: none">IO::Handle->output_field_separator(EXPR)\$OUTPUT_FIELD_SEPARATOR\$OFS\$_,		Current line number for the last file handled accessed	<ul style="list-style-type: none">HANDLE->input_line_number(EXPR)\$INPUT_LINE_NUMBER\$NR\$.	
Input record separator (newline by default)	<ul style="list-style-type: none">IO::Handle->input_record_separator(EXPR)\$INPUT_RECORD_SEPARATOR\$RS\$/		Output record separator	<ul style="list-style-type: none">IO::Handle->output_record_separator(EXPR)\$OUTPUT_RECORD_SEPARATOR\$ORS\$\	
Auto-flush control	<ul style="list-style-type: none">HANDLE->autoflush(EXPR)\$OUTPUT_AUTOFLUSH\$!		Last read file handle	\${^LAST_FH}	
<ul style="list-style-type: none">Variables related to format					
Current value of the write() accumulator for format() lines.	<ul style="list-style-type: none">\$ACCUMULATOR\$^A				
Form feed format, defaults to \f	<ul style="list-style-type: none">IO::Handle->format_formfeed(EXPR)\$FORMAT_FORMFEED\$^L		Set of characters after which a string may be broken to fill continuation fields	<ul style="list-style-type: none">IO::Handle->format_line_break_characters EXPR\$FORMAT_LINE_BREAK_CHARACTERS\$:	

Number of lines left on the page on currently selected output channel	<ul style="list-style-type: none"> HANDLE->format_lines_left(EXPR) \$FORMAT_LINES_LEFT \$_ 	Current page length of current output channel	<ul style="list-style-type: none"> HANDLE->format_lines_per_page(EXPR) \$FORMAT_LINES_PER_PAGE \$=
Name of current top-page format of output channel	<ul style="list-style-type: none"> HANDLE->format_top_name(EXPR) \$FORMAT_TOP_NAME \$^ 	Report format name of output channel	<ul style="list-style-type: none"> HANDLE->format_name(EXPR) \$FORMAT_NAME \$~
<ul style="list-style-type: none"> Error Variables 	The variables \$@ , \$! , \$^E , and \$? contain information about different types of error conditions that may appear during execution of a Perl program. They correspond to errors detected by the Perl interpreter, C library, operating system, or an external program, respectively.		
Perl error from the last eval operator	<ul style="list-style-type: none"> \$EVAL_ERROR \$@ 	Current state of interpreter	<ul style="list-style-type: none"> \$EXCEPTIONS_BEING_CAUGHT \$^S
Current value of C errno integer variable	<ul style="list-style-type: none"> \$OS_ERROR \$ERRNO \$! 	Hash of error names to 0 or 1. set to 1 if current error is this error.	<ul style="list-style-type: none"> %OS_ERROR %ERRNO %!
OS detected error	<ul style="list-style-type: none"> \$EXTENDED_OS_ERROR \$^E 		
Status returned by last pipe close, backtick command, wait, waited, or system() call.	<ul style="list-style-type: none"> \$CHILD_ERROR \$? 	native status returned by last pipe close , backtick command, wait() or wiatpid() or system() call	\${^CHILD_ERROR_NATIVE}
Current value of warning switch	<ul style="list-style-type: none"> \$WARNING \$^W 	Current set of warning checks enabled by the use warnings pragma	\${^WARNING_BITS}
<ul style="list-style-type: none"> Variables related to the interpreter state 	These variables provide information about the current interpreter state.		
Flag associated with the -c switch	<ul style="list-style-type: none"> \$COMPILING \$^C 	The current value of the debugging flags	<ul style="list-style-type: none"> \$DEBUGGING \$^D
Current phase of the perl interpreter	\${^GLOBAL_PHASE}		
Compile-time hints for the perl interpreter. Internal use only	\$^H	Values of compiled statements	%^H
Input/Output Layers. Internal use by PerlIO only.	\${^OPEN}		
Debugging support. Internal variable.	<ul style="list-style-type: none"> \$PERLDB \$^P 		
Taint mode	\${^TAINT}	Safe locale operations availability	\${^SAFE_LOCALES}
Unicode Settings of Perl	\${^UNICODE}		
Internal UTF-8 offset caching code state	\${^UTF8CACHE}	State of UTF-8 locale detected by perl at startup.	\${^UTF8LOCALE}
<ul style="list-style-type: none"> Deprecated and removed variables: 	\$# \$* \$[\${^ENCODING} \${^WIN32_SLOPPY_STAT}		

Perl 5 Statements 🚧

Conditional statements	
Loop statements	<ul style="list-style-type: none"> <code>while (condition) { ... }</code> <code>until (condition) { ... }</code>

Perl 5 Functions 🚧

Perl Functions Perl syntax	 To get information about a Perl function from the command line use the perldoc -f command.  To get information about print use: perldoc -f print		
 Cautionary notes			
<ul style="list-style-type: none">• each keyword is broken• Use Var::Pairs instead.	<p>Do NOT use the built-in each. It is broken, as described by Damian Conway in his Modern Perl Best Practice O'Reilly course, section control structure.</p> <ul style="list-style-type: none">• each is not re-entrant:<ul style="list-style-type: none">• nested loops of each over the same hash does not work as expected and will create infinite loop since the nested loop each juts iterates from where the first loop each left it.• Exiting the loop leaves the state of the each internal pointer at the current location.<ul style="list-style-type: none">• If you use each on the same hash later it will resume from where it left, it will not start form the beginning.		
print functions	<ul style="list-style-type: none">• print• say <code>use feature qw(say);</code> or <code>use v5.10;</code> (or higher). Like print, but implicitly appends a newline at the end of the list.		