

Help:Extension:ParserFunctions

From MediaWiki.org

The ParserFunctions extension provides ten additional parser functions to supplement the "magic words", which are already present in MediaWiki. All the parser functions provided by this extension take the form:

```
{{ #functionname: argument 1 | argument 2 | argument 3 ... }}
```

Contents

- 1 #expr:
- 2 #if:
- 3 #ifeq:
- 4 #iferror:
- 5 #ifexpr:
- 6 #ifexist:
- 7 #rel2abs:
- 8 #switch:
- 9 #time:
- 10 #titleparts:
- 11 General points
 - 11.1 Substitution
 - 11.2 Tables
 - 11.3 Stripping whitespace
- 12 See also

#expr:

This function evaluates a mathematical expression and returns the calculated value.

```
{{#expr: expression }}
```

The available operators are listed to the right, in order of precedence. See Help:Calculation for more details of the function of each operator. The accuracy and format of the result returned will vary depending on the operating system of the server running the wiki, and the number format of the site language.

When evaluating using boolean algebra, zero evaluates to *false* and any nonzero value, positive or negative, evaluates to *true*:

```
{{#expr: 1 and -1 }} → 1
```

An empty input expression returns an empty string. Invalid expressions return one of several error messages, which can be caught using the `#iferror: function`:

Type	Operators
Grouping (parentheses)	()
Numbers	1234.5 2.4E5 e (2.718) pi (3.142)
Logic	or
	and
	= != <> > < <= >=
Round	round
Binary	+ - mod
	* / div ^

```

{{#expr: }} →
{{#expr: 1+ }} → Expression error: Missing operand for +
{{#expr: 1 foo 2 }} → Expression error: Unrecognised word "foo"

```

Unary	not	ceil
	trunc	floor
	abs	ln
	cos	tan
	asin	atan
	e	+ -

Warning: Some expressions may invoke floating-point errors when used with very large or very small numbers:

```

{{#expr: 20060618093259 mod 10000}} → 3259 in most cases, but may occasionally give -6357. This varies with the specification and configuration of the server running the wiki. See bug 6356.

```

#if:

```

{{#if: test string | value if true | value if false }}

```

This function tests whether the first parameter is 'non-empty'. It evaluates to false if the test string is empty or contains only whitespace characters (space, newline, etc).

```

{{#if: | yes | no}} → no
{{#if: string | yes | no}} → yes
{{#if:      | yes | no}} → no
{{#if:
| yes | no}} → no

```

The test string is always interpreted as pure text, so mathematical expressions are not evaluated:

```

{{#if: 1==2 | yes | no}} → yes

```

Either or both the return values may be omitted:

```

{{#if: foo | yes }} → yes
{{#if: | yes }} →
{{#if: foo | | no}} →

```

See Help:Parser functions in templates for more examples of this parser function.

#ifeq:

This parser function compares two strings and determines whether they are identical.

```

{{#ifeq: string 1 | string 2 | value if true | value if false }}

```

If both strings are valid numerical values, the strings are compared numerically:

```

{{#ifeq: 01 | 1 | yes | no}} → yes
{{#ifeq: 0 | -0 | yes | no}} → yes

```

Otherwise the comparison is made as text; this comparison is case sensitive:

```

{{#ifeq: foo | bar | yes | no}} → no
{{#ifeq: foo | Foo | yes | no}} → no
{{#ifeq: "01" | "1" | yes | no}} → no

```

Warning: Text inside `<nowiki>` tags is hashed within parser functions, resulting in errors:

```
{{#ifeq: <nowiki>foo</nowiki> | <nowiki>foo</nowiki> | yes | no}} → no
```

#iferror:

This function takes an input string and returns one of two results; the function evaluates to `true` if the input string contains an HTML object with `class="error"`, as generated by other parser functions such as `#expr:`, `#time:` and `#rel2abs:`, template errors such as loops and recursions, and other 'failsoft' parser errors.

```
{{#iferror: test string | value if error | value if correct }}
```

One or both of the return strings can be omitted. If the `correct` string is omitted, the `test string` is returned if it is not erroneous. If the `error` string is also omitted, an empty string is returned on an error:

```
{{#iferror: {{#expr: 1 + 2 }} | error | correct }} → correct  
{{#iferror: {{#expr: 1 + X }} | error | correct }} → error  
{{#iferror: {{#expr: 1 + 2 }} | error }} → 3  
{{#iferror: {{#expr: 1 + X }} | error }} → error  
{{#iferror: {{#expr: 1 + 2 }} }} → 3  
{{#iferror: {{#expr: 1 + X }} }} →
```

#ifexpr:

This function evaluates a mathematical expression and returns one of two strings depending on the boolean value of the result:

```
{{#ifexpr: expression | value if true | value if false }}
```

The `expression` input is evaluated exactly as for `#expr:` above, with the same operators being available. The output is then evaluated as a boolean expression. This function is equivalent to one using `#ifeq:` and `#expr:` only:

```
{{#ifeq: {{#expr: expression }} | 0 | value if false | value if true }}
```

An empty input expression evaluates to false:

```
{{#ifexpr: | yes | no}} → no
```

Either or both the return values may be omitted; no output is given when the appropriate branch is left empty:

```
{{#ifexpr: 1 > 0 | yes }} → yes  
{{#ifexpr: 1 < 0 | yes }} →  
{{#ifexpr: 1 > 0 | | no}} →  
{{#ifexpr: 1 > 0 }} →
```

#ifexist:

This function takes an input string, interprets it as a page title, and returns one of two values depending on whether or not the page exists on the local wiki.

```
{{#ifexist: page title | value if exists | value if doesn't exist }}
```

The function evaluates to `true` if the page exists, whether it contains content, is visibly blank (contains meta-data such as category links or magic words, but no visible content), is blank, or is a redirect. Only pages


that are redlinked evaluate to false, including if the page used to exist but has been deleted.

```
{{#ifexist: Help:Extension:ParserFunctions | exists | doesn't exist }} → exists
{{#ifexist: XXXHelp:Extension:ParserFunctionsXXX | exists | doesn't exist }} →
doesn't exist
```

The function evaluates to true for system messages that have been customised, and for special pages that are defined by the software.

```
{{#ifexist: Special:Watchlist | exists | doesn't exist }} → exists
{{#ifexist: Special:CheckUser | exists | doesn't exist }} → exists (because the CheckUser
extension is installed on this wiki)
{{#ifexist: MediaWiki:Copyright | exists | doesn't exist }} → exists (because
MediaWiki:Copyright has been customised)
```

`#ifexist:` is considered an "expensive parser function", only a limited number of which can be included on any one page (including functions inside transcluded templates). When this limit is exceeded, the page is categorised into Category:Pages with too many expensive parser function calls, and any further `#ifexist:` functions automatically return false, whether the target page exists or not.

 **Tip for wiki admins:** Configure the maximum number of allowed expensive parser functions using the `$wgExpensiveParserFunctionLimit` variable.

If a page checks a target using `#ifexist:`, then that page will appear in the Special:WhatLinksHere list for the target page. So if the code `{{#ifexist:Foo}}` were included live on this page (Help:Extension:ParserFunctions), Special:WhatLinksHere/Foo will list Help:Extension:ParserFunctions.

On wikis using a shared media repository, `#ifexist:` can be used to check if a file has been uploaded to the repository, but not to the wiki itself:

```
{{#ifexist: File:Example.png | exists | doesn't exist }} → doesn't exist
{{#ifexist: Image:Example.png | exists | doesn't exist }} → doesn't exist
{{#ifexist: Media:Example.png | exists | doesn't exist }} → exists
```

If a local description page has been created for the file, the result is **exists** for all of the above.

#rel2abs:

This function converts a relative file path into an absolute filepath.

```
{{#rel2abs: path }}
{{#rel2abs: path | base path }}
```

Within the `path` input, the following syntax is valid:

- `.` → the current level
- `..` → "go up one level"
- `/foo` → "go down one level into the subdirectory /foo"

If the `base path` is not specified, the full page name of the page will be used instead:

```
{{#rel2abs: ./quok | Help:Foo/bar/baz }} → Help:Foo/bar/baz/quok
{{#rel2abs: ../quok | Help:Foo/bar/baz }} → Help:Foo/bar/quok
{{#rel2abs: ../. | Help:Foo/bar/baz }} → Help:Foo/bar
```

Invalid syntax, such as `/.` or `./.`, is ignored. Since no more than two consecutive full stops are permitted,

sequences such as these can be used to separate successive statements:

```
{{#rel2abs: ../quok/. | Help:Foo/bar/baz }} → Help:Foo/bar/quok
{{#rel2abs: ../../quok | Help:Foo/bar/baz }} → Help:Foo/quok
{{#rel2abs: ../../../../quok | Help:Foo/bar/baz }} → quok
{{#rel2abs: ../../../.././quok | Help:Foo/bar/baz }} → Error: Invalid depth in path:
"Help:Foo/bar/baz/../.././quok" (tried to access a node above the root node)
```

#switch:

This function compares one input value against several test cases, returning an associated string if a match is found.

```
{{#switch: comparison string
| case = result
| case = result
| ...
| case = result
| default result
}}
```

The *default result* is returned if no *case* string matches the *comparison string*. In this syntax, the default result must be the last parameter and must not contain a raw equals sign. Alternatively, the default result may be explicitly declared with a case string of "#default"; default results declared in this way may be placed anywhere within the function:

```
{{#switch: test | foo = Foo | #default = Bar | baz = Baz }} → Bar
```

If *default* parameter is entirely omitted, an empty string will be returned as default result. It is possible to have 'fall through' values, where several *case* strings return the same *result* string. This minimises duplication.

```
{{#switch: comparison string
| case1 = result1
| case2
| case3
| case4 = result2
| case5 = result3
| case6
| case7 = result4
| default result
}}
```


Here cases 2, 3 and 4 all return *result2*; cases 6 and 7 both return *result4*

As with `#ifeq:`, the comparison is made numerically if both the comparison string and the case string being tested are numeric; or as case-sensitive string otherwise. A *case* string may be empty:

```
{{#switch: | = Nothing | foo = Foo | Something }} → Nothing
```

Once a match is found, subsequent *cases* are ignored:

```
{{#switch: b | f = Foo | b = Bar | b = Baz | }} → Bar
```

 **Warning:** "Case" strings cannot contain raw equals signs:

```
{{#switch: 1=2 | 1=2 = raw | 1<nowiki>=</nowiki>2 = nowiki | 1&#61;2 = html |
foo }} → foo
```

#time:

This parser function takes a date and/or time construct and formats it according to the syntax given. A date/time object can be specified; the default is the value of the magic word `{{CURRENTTIMESTAMP}}` – that is, the time the page was last rendered into HTML.

```
{{#time: format string }}
{{#time: format string | date/time object
}}
```


The list of accepted formatting codes is given in the table to the right. Any character in the formatting string that is not recognised is passed through unaltered. There are also two ways to escape characters within the formatting string:

1. A backslash followed by a formatting character is interpreted as a single literal character
2. characters enclosed in double quotes are considered literal characters, and the quotes are removed

In addition, the digraph `xx` is interpreted as a single literal `"x"`.

```
{{#time: Y-m-d }} → 2009-06-05
{{#time: [[Y]] m d }} → 2009 06 05
{{#time: [[Y (year)]] }} → 2009 (09epmFri,
05 Jun 2009 15:12:53 +0000)
{{#time: [[Y "(year)"]] }} → 2009 (year)
{{#time: i's" }} → 12'53"
```

The *date/time object* can be in any format accepted by PHP's `strtotime()` (<http://uk3.php.net/manual/en/function.strtotime.php>) function. Both absolute (eg 20 December 2000) and relative (eg +20 hours) times are accepted.

 **Warning:** The range of acceptable input is January 1 0100 → December 31 9999. Values outside this range will be misinterpreted:

```
{{#time: d F Y | 15 April 0099 }} →
15 April 1999
{{#time: d F Y | 15 April 10000 }}
→ Error: invalid time
```

Full or partial absolute dates can be specified; the function will 'fill in' parts of the date that are not specified using the *current* values:

```
{{#time: Y | January 1 }} → 2009
```

A four-digit number is interpreted as hours and minutes if possible, and otherwise as year:

Code	Description	Current output
Year		
Y	4-digit year.	2009
y	2-digit year.	09
L	1 or 0 whether it's a leap year or not	0
o ¹	ISO-8601 year number. ²	2009 ³
<small>¹ Requires PHP 5.1.0 and newer and rev:45208 ² This has the same value as Y, except that if the ISO week number (W) belongs to the previous or next year, that year is used instead. ³ Will output literal <i>o</i> if ¹ not fulfilled</small>		
Month		
n	Month index, not zero-padded.	6
m	Month index, zero-padded.	06
M	An abbreviation of the month name, in the site language.	Jun
F	The full month name in the site language.	June
Week		
W	ISO 8601 week number, zero-padded.	23
Day		
j	Day of the month, not zero-padded.	5
d	Day of the month, zero-padded.	05
z	Day of the year (January 1 = 0)	155
D	An abbreviation for the day of the week. Rarely internationalised.	Fri
l	The full weekday name. Rarely internationalised.	Friday
N	ISO 8601 day of the week.	5
w	number of the day of the week (Monday = 1).	5

{{#time: Y m d H:i:s | 1959 }} → **2009 06 05**

19:59:00 Input is treated as a time rather than a year.

{{#time: Y m d H:i:s | 1960 }} → **1960 06 05**


15:12:53 Since 19:60 is no valid time, 1960 is treated as a year.

A six-digit number is interpreted as hours, minutes and seconds if possible, but otherwise as an error (not, for instance, a year and month):

{{#time: Y m d H:i:s | 195909 }} → **2009 06 05**

19:59:09 Input is treated as a time rather than a year+month code.

{{#time: Y m d H:i:s | 196009 }} → **Error: invalid time** Although 19:60:09 is no valid time, 196009 is not interpreted as September 1960.

 **Warning:** The fill-in feature is not consistent; some parts are filled in using the current values, others are not:

{{#time: Y m d H:i:s | January 1 }}
→ **2009 01 01 00:00:00**

{{#time: Y m d H:i:s | February 2007 }}
→ **2007 02 01 00:00:00** Goes to the start of the month, not the current day.

The function performs a certain amount of date mathematics:

{{#time: d F Y | January 0 2008 }} → **31 December 2007**

{{#time: d F | January 32 }} → **01 February**

{{#time: d F | February 29 2008 }} → **29 February**

{{#time: d F | February 29 2007 }} → **01 March**

The function recognizes a large number of placenames and time zones (full list):

{{#time:c|December 7, 1941 8:43AM HST }} → **1941-12-07T18:43:00+00:00**

{{#time:c|December 8, 1941 12:30PM Asia/Manila }} → **1941-12-08T04:30:00+00:00**

Hour		
a	"am" during the morning (00:00:00 → 11:59:59), "pm" otherwise (12:00:00 → 23:59:59)	pm
A	Uppercase version of a above.	PM
g	Hour in 12-hour format, not zero-padded.	3
h	Hour in 12-hour format, zero-padded.	03
G	Hour in 24-hour format, not zero-padded.	15
H	Hour in 24-hour format, zero-padded.	15
Minutes and seconds		
i	Minutes past the hour, zero-padded.	12
s	Seconds past the minute, zero-padded.	53
U	Seconds since January 1 1970 00:00:00 GMT.	1244214773
Miscellaneous		
L	1 if this year is a leap year in the Gregorian calendar, 0 otherwise	0
t	Number of days in the current month.	30
c	ISO 8601 formatted date, equivalent to Y-m-dTH:i:s+00:00.	2009-06-05T15:12:53+00:00
r	RFC 2822 formatted date, equivalent to D, j M Y H:i:s +0000, with weekday name and month name not internationalised.	Fri, 05 Jun 2009 15:12:53 +0000
Non-Gregorian calendars		
Iranian		
xij	Day of the month	15
xiF	Full month name	Khordad
xin	Month index	3
xiY	Full year	1388
Hebrew		

#titleparts:

This function separates a pagetitle into segments based on slashes, then returns some of those segments as output.

```
{{#titleparts: pagename | number of segments to return | first segment to return }}
```

If the `number of segments` parameter is not specified, it defaults to "0", which returns *all* the segments. If the `first segment` parameter is not specified or is "0", it defaults to "1":

```
{{#titleparts: Talk:Foo/bar/baz/quok }} → Talk:Foo/bar/baz/quok
{{#titleparts: Talk:Foo/bar/baz/quok | 1 }} → Talk:Foo
{{#titleparts: Talk:Foo/bar/baz/quok | 2 }} → Talk:Foo/bar
{{#titleparts: Talk:Foo/bar/baz/quok | 2 | 2 }} → bar/baz
```

Negative values are accepted for both values. Negative values for `number of segments` effectively 'strips' segments from the end of the string. Negative values for `first segment` translates to "add this value to the total number of segments", loosely equivalent to "count from the right":

```
{{#titleparts: Talk:Foo/bar/baz/quok | -1 }} → Talk:Foo/bar/baz
{{#titleparts: Talk:Foo/bar/baz/quok | | -1 }} → quok
{{#titleparts: Talk:Foo/bar/baz/quok | -1 | 2 }} → bar/baz Strips one segment from the end of the string, then returns the second segment and beyond
```

The string is split a maximum of 25 times; further slashes are ignored. The string is also limited to 255 characters, as it is treated as a page title:

```
{{#titleparts: a/b/c/d/e/f/g/h/i/j/k/l/m/n/o/p/q/r/s/t/u/v/w/x/y/z/aa/bb/cc/dd/ee | 1 | 25 }} → y/z/aa/bb/cc/dd/ee
```

General points

Substitution

Parser functions can be substituted by prefixing the hash character with **subst::**:

```
{{subst:#ifexist: Help:Extension:ParserFunctions | [[Help:Extension:ParserFunctions]] | Help:Extension:ParserFunctions }} → the code [[Help:Extension:ParserFunctions]] will be inserted in the wikitext since the page Help:Extension:ParserFunctions exists.
```

Warning: The results of substituted parser functions are undefined if the expressions contain *unsubstituted* volatile code such as variables or other parser functions. For consistent results, all the volatile code in the expression to be evaluated must be substituted. See Help:Substitution.

xjJ	Day of the month	13
xjF	Full month name	Sivan
xjX	Genitive form of the month name	Sivan
xjN	Month number	9
xjY	Full year	5769
Thai solar		
xkY	Full year	2552
Flags		
xn	Format the next numeric code as a raw ASCII number.	In the Hindi language, <code>{{#time:H, xnH}}</code> produces ०६, 06
xN	Like <code>xn</code> , but as a toggled flag, which endures until the end of the string or until the next appearance of <code>xN</code> in the string.	
xr	Format the next number as a roman numeral. Only works for numbers up to 3000.	<code>{{#time:xrY}}</code> → MMIX
xg	Before a month flag (<code>n, m, M, F</code>), output the genitive form if the site language distinguishes between genitive and nominative forms.	