



# Inhaltsverzeichnis

---

# Modul:String/Doku

---

**Dies ist die Dokumentationsseite für Modul:String**

Vorlage:Lmd

This module is intended to provide access to basic string functions.

Most of the functions provided here can be invoked with named parameters, unnamed parameters, or a mixture. If named parameters are used, Mediawiki will automatically remove any leading or trailing whitespace from the parameter. Depending on the intended use, it may be advantageous to either preserve or remove such whitespace.

Inhaltsverzeichnis	
1 Global options .....	2
2 len .....	3
3 sub .....	3
4 sublength .....	4
5 match .....	4
6 pos .....	6
7 str_find .....	6
8 find .....	7
9 replace (gsub) .....	8
10 rep .....	9
11 escapePattern .....	9
12 count .....	9
13 join .....	10
14 endswith .....	11
15 See also .....	11

## Global options

---

### **ignore\_errors**

If set to 'true' or 1, any error condition will result in an empty string being returned rather than an error message.

### **error\_category**

If an error occurs, specifies the name of a category to include with the error message. The default category is [Vorlage:Clc](#).

### **no\_category**

If set to 'true' or 1, no category will be added if an error is generated.

Unit tests for this module are available at [Module:String/testcases](#).

## len

This function returns the length of the target string.

Usage:

```
{{#invoke:String|len|target_string}}
```

OR

```
{{#invoke:String|len|s= target_string }}
```

Parameters:

**s**

The string whose length to report

Examples:

- `{{#invoke:String|len| abcdefghi }}` → 11
- `{{#invoke:String|len|s= abcdefghi }}` → 9

## sub

This function returns a substring of the target string at specified inclusive, one-indexed indices.

Usage:

```
{{#invoke:String|sub|target_string|start_index|end_index}}
```

OR

```
{{#invoke:String|sub|s= target_string |i= start_index |j= end_index }}
```

Parameters:

**s**

The string to return a subset of

**i**

The first index of the substring to return, defaults to 1.

**j**

The last index of the string to return, defaults to the last character.

The first character of the string is assigned an index of 1. If either *i* or *j* is a negative value, it is interpreted the same as selecting a character by counting from the end of the string. Hence, a value of -1 is the same as selecting the last character of the string.

If the requested indices are out of range for the given string, an error is reported. To avoid error messages, use **Vorlage:MI** instead.

Examples:

- `"{{#invoke:String|sub| abcdefghi }}"` → " abcdefghi "



- `"{{#invoke:String|sub|s= abcdefghi }}"` → "abcdefghi"
- `"{{#invoke:String|sub| abcdefghi | 3 }}"` → "bcdefghi "
- `"{{#invoke:String|sub|s= abcdefghi |i= 3 }}"` → "cdefghi"
- `"{{#invoke:String|sub| abcdefghi | 3 | 4 }}"` → "bc"
- `"{{#invoke:String|sub|s= abcdefghi |i= 3 |j= 4 }}"` → "cd"

## sublength

---

This function implements the features of [Vorlage:TI](#) and is kept in order to maintain these older templates. It returns a substring of the target string starting at a specified index and of a specified length.

Usage:

```
{{#invoke:String|sublength|s= target_string |i= start_index |len= length }}
```

Parameters:

**s**

The string

**i**

The starting index of the substring to return. The first character of the string is assigned an index of 0.

**len**

The length of the string to return, defaults to the last character.

Examples:

- `{{#invoke:String|sublength|s= abcdefghi }}"` → abcdefghi
- `{{#invoke:String|sublength|s= abcdefghi |i= 3 }}"` → defghi
- `{{#invoke:String|sublength|s= abcdefghi |i= 3 |len= 4 }}"` → defg

## match

---

This function returns a substring from the source string that matches a specified pattern.

Usage:

```
{{#invoke:String|match|source_string|pattern_string|start_index|match_number|plain_flag|nomatch_output}}
```

OR

```
{{#invoke:String|match|s= source_string |pattern= pattern_string |start= start_index |match= match_number |plain= plain_flag |nomatch= nomatch_output }}
```

Parameters:

**s**

The string to search

## pattern

The pattern or string to find within the string

## start

The index within the source string to start the search. The first character of the string has index 1. Defaults to 1.

## match

In some cases it may be possible to make multiple matches on a single string. This specifies which match to return, where the first match is `match= 1`. If a negative number is specified then a match is returned counting from the last match. Hence `match = -1` is the same as requesting the last match. Defaults to 1.

## plain

Boolean flag indicating that pattern should be understood as plain text and not as a [Scribunto ustring pattern](#) (a unicode-friendly [Lua-style regular expression](#)). Defaults to false (to change: `plain=true`)

## nomatch

If no match is found, output the "nomatch" value rather than an error.

## ignore\_errors

If no match is found and `ignore_errors=true`, output an empty string rather than an error.

If the `match_number` or `start_index` are out of range for the string being queried, then this function generates an error. An error is also generated if no match is found. If one adds the parameter `ignore_errors=true`, then the error will be suppressed and an empty string will be returned on any failure.

For information on constructing [Lua patterns](#), a form of [regular expression](#), see:

- [Scribunto patterns](#)
- [Scribunto Unicode string patterns](#)

Examples:

- `{{#invoke:String|match| abc123def456 |%d+}} → 123`
- `{{#invoke:String|match|s= abc123def456 |pattern= %d+ }} → 123`
- `{{#invoke:String|match| abc123def456 |%d+|6}} → 23`
- `{{#invoke:String|match|s= abc123def456 |pattern= %d+ |start= 6 }} → 3`
- `{{#invoke:String|match|s= abc123def456 |pattern= %d+ |start= 6 |match= 2 }} → 456`
- `{{#invoke:String|match|s= abc123%d+ |pattern= %d+ }} → 123`
- `{{#invoke:String|match|s= abc123%d+ |pattern= %d+ |plain= true }} → %d+`
- `{{#invoke:String|match|s= abc |pattern= %d }} → String Module Error: Match not found`
- `{{#invoke:String|match|s= abc |pattern= %d |nomatch= No numeric characters in string }} → No numeric characters in string`
- `{{#invoke:String|match|s= abc |pattern= %d |ignore_errors= true }} →`
- `{{#invoke:String|match|s= 0012001200 |pattern= 0*(%d*) }} → 12001200`

## pos

This function returns a single character from the target string at position pos.

Usage:

```
{{#invoke:String|pos|target_string|index_value}}
```

OR

```
{{#invoke:String|pos|target=target_string |pos=index_value }}
```

Parameters:

### **target**

The string to search

### **pos**

The index for the character to return

The first character has an index value of 1.

If one requests a negative value, this function will select a character by counting backwards from the end of the string. In other words pos = -1 is the same as asking for the last character.

A requested value of zero, or a value greater than the length of the string returns an error.

Examples:

- ```
{{#invoke:String|pos| abcdefghi | 4 }}
```

 → c
- ```
{{#invoke:String|pos|target= abcdefghi |pos= 4 }}
```

 → d

## str\_find

This function duplicates the behavior of [Vorlage:TI](#), including all of its quirks. This is provided in order to support existing templates, but is NOT RECOMMENDED for new code and templates. New code is recommended to use the "find" function instead.

Returns the first index in "source" that is a match to "target". Indexing is 1-based, and the function returns -1 if the "target" string is not present in "source".

Important Note: If the "target" string is empty / missing, this function returns a value of "1", which is generally unexpected behavior, and must be accounted for separately.

Usage:

```
{{#invoke:String|str_find|source_string|target_string}}
```

OR

```
{{#invoke:String|str_find|source=source_string |target=target_string }}
```

Parameters:

### **source**

The string to search



## target

The string to find within source

Examples:

- `{{#invoke:String|str_find| abc123def }} → 1`
- `{{#invoke:String|str_find|source= abc123def }} → 1`
- `{{#invoke:String|str_find| abc123def |123}} → 5`
- `{{#invoke:String|str_find|source= abc123def |target= 123 }} → 4`
- `{{#invoke:String|str_find| abc123def |not}} → -1`

## find

---

This function allows one to search for a target string or pattern within another string.

Usage:

```
{{#invoke:String|find|source_string|target_string|start_index|plain_flag}}
```

OR

```
{{#invoke:String|find|source= source_string |target= target_string |start=  
start_index |plain= plain_flag }}
```

Parameters:

### source

The string to search

### target

The string or pattern to find within source

### start

The index within the source string to start the search, defaults to 1

### plain

Boolean flag indicating that target should be understood as plain text and not as a [Scribunto ustring pattern](#) (a unicode-friendly [Lua-style regular expression](#)); defaults to true

This function returns the first index  $\geq$  "start" where "target" can be found within "source". Indices are 1-based. If "target" is not found, then this function returns 0. If either "source" or "target" are missing / empty, this function also returns 0.

This function should be safe for UTF-8 strings.

Examples:

- `{{#invoke:String|find|abc123def|12}} → 4`
- `{{#invoke:String|find|source=abc123def|target=12}} → 4`
- `{{#invoke:String|find|source=abc123def|target=pqr}} → 0`
- `{{#invoke:String|find| abc123def |123}} → 5`



- `{{#invoke:String|find|source= abc123def |target= 123 }} → 4`
- `{{#invoke:String|find|source=abc123def|target=%d |start=3 |plain=false }} → 4`

When using unnamed parameters, preceding and trailing spaces are kept and counted:

- `{{#invoke:String|find| abc123def |c|false}} → 5`
- `{{#invoke:String|find|source= abc123def |target=c|plain=false}} → 3`
- `{{#invoke:string|find|abc 123 def|s|plain=false}} → 4`

Testing for the presence of a string:

- `Vorlage:Pf` → Didn't find needle

[Vorlage:Anchor](#)

## replace (gsub)

---

This function allows one to replace a target string or pattern within another string. To Lua programmers: this function works internally by calling [Vorlage:Code](#).

Usage:

```
{{#invoke:String|replace|source_str|pattern_string|replace_string|replacement_count|plain_flag}}
```

OR

```
{{#invoke:String|replace|source= source_string |pattern= pattern_string |replace= replace_string |count= replacement_count |plain= plain_flag }}
```

Parameters:

### source

The string to search

### pattern

The string or pattern to find within source

### replace

The replacement text

### count

The number of occurrences to replace; defaults to all

### plain

Boolean flag indicating that pattern should be understood as plain text and not as a [Scribunto ustring pattern](#) (a unicode-friendly [Lua-style regular expression](#)); defaults to true

Examples:

- `"{{#invoke:String|replace| abc123def456 |123|XYZ}}"` → " abcXYZdef456 "
- `"{{#invoke:String|replace|source= abc123def456 |pattern= 123 |replace= XYZ }}"`  
→ "abcXYZdef456"
- `"{{#invoke:String|replace| abc123def456 |%d+|XYZ|1|false}}"` → " abcXYZdef456 "



- `"{{#invoke:String|replace|source= abc123def456 |pattern= %d+ |replace= XYZ |count=1 |plain= false }}"` → "abcXYZdef456"
- `"{{#invoke:String|replace|source= abc123def456 |pattern= %d+ |replace= XYZ |plain= false }}"` → "abcXYZdefXYZ"
- `{{#invoke:String|replace|source= 0012001200 |pattern= ^0* |plain= false }}` → 12001200

## rep

---

Repeats a string *n* times. A simple function to pipe `string.rep` to templates.

Usage:

```
{{#invoke:String|rep|source|count}}
```

Parameters:

### source

The string to repeat

### count

The number of repetitions.

Examples:

- `"{{#invoke:String|rep|hello|3}}"` → "hellohellohello"
- `"{{#invoke:String|rep| hello | 3 }}"` → " hello hello hello "

## escapePattern

---

In a [Lua pattern](#), changes a *class character* into a *literal character*. For example: in a pattern, character `.` catches "any character"; `escapePattern` will convert it to `%.`, catching just the literal character ".".

Usage:

- `{{#invoke:String|escapePattern|pattern_string}}`

Parameters:

### pattern\_string

The pattern string to escape

Examples:

- `"{{#invoke:String|escapePattern|A.D. }}"` → "A%.D%."
- `"{{#invoke:String|escapePattern|10% }}"` → "10%%"

## count

---

Counts the number of times a given pattern appears in the arguments that get passed on to this module. Counts disjoint matches only.

Usage:

```
{{#invoke:String|count|source_str|pattern_string|plain_flag}}
```

OR

```
{{#invoke:String|count|source= source_string |pattern= pattern_string|plain= plain_flag }}
```

Parameters:

### source\_string

The string to count occurrences in

### pattern

The string or pattern to count occurrences of within source

### plain

Boolean flag indicating that pattern should be understood as plain text and not as a [Scribunto ustring pattern](#) (a unicode-friendly [Lua-style regular expression](#)); defaults to true

Examples:

- Count of 'a': "

```
{{#invoke:String|count|aabbcc|a}}
```

" → "2"
- Count occurrences of 'aba': "

```
{{#invoke:String|count|ababababab|aba}}
```

" → "2"
- Count of "either 'a' or 'c' ": "

```
{{#invoke:String|count|aabbcc|[ac]|plain=false}}
```

" → "4"
- Count of "not 'a' ": "

```
{{#invoke:String|count|aaabaaac|^[a]|plain=false}}
```

" → "2"
- Count of "starts with 'a' ": "

```
{{#invoke:String|count|aaabaaac|^[a]|plain=false}}
```

" → "1"

## join

---

Joins all strings passed as arguments into one string, treating the first argument as a separator

Usage:

```
{{#invoke:String|join|separator|string1|string2|...}}
```

Parameters:

### separator

String that separates each string being joined together

Note that leading and trailing spaces are *not* stripped from the separator.

### string1/string2/...

Strings being joined together

Examples:

- "

```
{{#invoke:String|join|x|foo|bar|baz}}
```

" → "fooxbarxbaz"
- "

```
{{#invoke:String|join||a|b|c|d|e|f|g}}
```

" → "abcdefg"
- "

```
{{#invoke:String|join|,|a|b|c|d|e|f|g}}
```

" → "a,b,c,d,e,f,g"
- "

```
{{#invoke:String|join|, |a|b|c|d|e|f|g}}
```

" → "a, b, c, d, e, f, g"
- "

```
{{#invoke:String|join|-|a|b|c|d|e|f|g}}
```

" → "a - b - c - d - e - f - g"



The preceding example uses the html entity &ndash; but the unicode character also works.

## endswith

---

**Vorlage:For** Usage:

```
{{#invoke:String|endswith|source_str|search_string}}
```

OR

```
{{#invoke:String|endswith|source= source_string |pattern= search_string}}
```

Returns "yes" if the source string ends with the search string. Use named parameters to have the strings trimmed before use. Despite the parameter name, *search\_string* is not a Lua pattern, it is interpreted literally.

- `"{{#invoke:String|endswith|xxxyyy|y}}"` → "yes"
- `"{{#invoke:String|endswith|xxxyyy|z}}"` → ""

## See also

---

**Vorlage:String** handling templates