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

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:Ml 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:Tl 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:Tl, 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 >= "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

```
--[[

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.

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
        [Category:Errors reported by Module String].

    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/tests.
]]

local str = {}

--[[
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

If invoked using named parameters, Mediawiki will automatically remove any leadir
trailing whitespace from the target string.
]]
function str.len( frame )
        local new_args = str._getParameters( frame.args, {'s'} )
        local s = new_args['s'] or ''
        return mw.ustring.len( s )
end

--[[
sub

This function returns a substring of the target string at specified 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 fist 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.
]]
function str.sub( frame )
        local new_args = str._getParameters( frame.args, { 's', 'i', 'j' } )
        local s = new_args['s'] or ''
        local i = tonumber( new_args['i'] ) or 1
        local j = tonumber( new_args['j'] ) or -1

        local len = mw.ustring.len( s )

        -- Convert negatives for range checking
        if i < 0 then
                i = len + i + 1
        end
        if j < 0 then
                j = len + j + 1
        end

        if i > len or j > len or i < 1 or j < 1 then
                return str._error( 'String subset index out of range' )
        end
        if j < i then
                return str._error( 'String subset indices out of order' )
        end

        return mw.ustring.sub( s, i, j )
end

--[[
This function implements that features of {{str sub old}} and is kept in order
```

```
to maintain these older templates.
]]
function str.sublength( frame )
        local i = tonumber( frame.args.i ) or 0
        local len = tonumber( frame.args.len )
        return mw.ustring.sub( frame.args.s, i + 1, len and ( i + len ) )
end

--[[
_match

This function returns a substring from the source string that matches a
specified pattern. It is exported for use in other modules

Usage:
strmatch = require("Module:String")._match
sresult = strmatch( s, pattern, start, match, plain, nomatch )

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: A flag indicating that the pattern should be understood as plain
        text.  Defaults to false.
    nomatch: If no match is found, output the "nomatch" value rather than an erro

For information on constructing Lua patterns, a form of [regular expression], see

* http://www.lua.org/manual/5.1/manual.html#5.4.1
* http://www.mediawiki.org/wiki/Extension:Scribunto/Lua_reference_manual#Patterns
* http://www.mediawiki.org/wiki/Extension:Scribunto/Lua_reference_manual#Ustring_

]]
-- This sub-routine is exported for use in other modules
function str._match( s, pattern, start, match_index, plain_flag, nomatch )
        if s == '' then
                return str._error( 'Target string is empty' )
        end
        if pattern == '' then
                return str._error( 'Pattern string is empty' )
        end
        start = tonumber(start) or 1
        if math.abs(start) < 1 or math.abs(start) > mw.ustring.len( s ) then
                return str._error( 'Requested start is out of range' )
        end
        if match_index == 0 then
                return str._error( 'Match index is out of range' )
        end
        if plain_flag then
                pattern = str._escapePattern( pattern )
        end

        local result
        if match_index == 1 then
                -- Find first match is simple case
                result = mw.ustring.match( s, pattern, start )
        else
                if start > 1 then
```

```
                        s = mw.ustring.sub( s, start )
                end

                local iterator = mw.ustring.gmatch(s, pattern)
                if match_index > 0 then
                        -- Forward search
                        for w in iterator do
                                match_index = match_index - 1
                                if match_index == 0 then
                                        result = w
                                        break
                                end
                        end
                else
                        -- Reverse search
                        local result_table = {}
                        local count = 1
                        for w in iterator do
                                result_table[count] = w
                                count = count + 1
                        end

                        result = result_table[ count + match_index ]
                end
        end

        if result == nil then
                if nomatch == nil then
                        return str._error( 'Match not found' )
                else
                        return nomatch
                end
        else
                return result
        end
end

--[[
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|plai
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: A flag indicating that the pattern should be understood as plain
        text.  Defaults to false.
    nomatch: If no match is found, output the "nomatch" value rather than an erro

If invoked using named parameters, Mediawiki will automatically remove any leadin
```

```
trailing whitespace from each string.  In some circumstances this is desirable, i
other cases one may want to preserve the whitespace.

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

For information on constructing Lua patterns, a form of [regular expression], see

* http://www.lua.org/manual/5.1/manual.html#5.4.1
* http://www.mediawiki.org/wiki/Extension:Scribunto/Lua_reference_manual#Patterns
* http://www.mediawiki.org/wiki/Extension:Scribunto/Lua_reference_manual#Ustring_

]]
-- This is the entry point for #invoke:String|match
function str.match( frame )
        local new_args = str._getParameters( frame.args, {'s', 'pattern', 'start'
        local s = new_args['s'] or ''
        local start = tonumber( new_args['start'] ) or 1
        local plain_flag = str._getBoolean( new_args['plain'] or false )
        local pattern = new_args['pattern'] or ''
        local match_index = math.floor( tonumber(new_args['match']) or 1 )
        local nomatch = new_args['nomatch']

        return str._match( s, pattern, start, match_index, plain_flag, nomatch )
end

--[[
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

If invoked using named parameters, Mediawiki will automatically remove any leadin
trailing whitespace from the target string.  In some circumstances this is desira
other cases one may want to preserve the whitespace.

The first character has an index value of 1.

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

A requested value of zero, or a value greater than the length of the string retur
]]
function str.pos( frame )
        local new_args = str._getParameters( frame.args, {'target', 'pos'} )
        local target_str = new_args['target'] or ''
        local pos = tonumber( new_args['pos'] ) or 0

        if pos == 0 or math.abs(pos) > mw.ustring.len( target_str ) then
                return str._error( 'String index out of range' )
        end

        return mw.ustring.sub( target_str, pos, pos )
end
```

```
--[[
str_find

This function duplicates the behavior of {{str_find}}, including all of its quirk
This is provided in order to support existing templates, but is NOT RECOMMENDED f
new code and templates.  New code is recommended to use the "find" function inste

Returns the first index in "source" that is a match to "target".  Indexing is 1-b
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
value of "1", which is generally unexpected behavior, and must be accounted for
separately.
]]
function str.str_find( frame )
        local new_args = str._getParameters( frame.args, {'source', 'target'} )
        local source_str = new_args['source'] or ''
        local target_str = new_args['target'] or ''

        if target_str == '' then
                return 1
        end

        local start = mw.ustring.find( source_str, target_str, 1, true )
        if start == nil then
                start = -1
        end

        return start
end

--[[
find

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

Usage:
{{#invoke:String|find|source_str|target_string|start_index|plain_flag}}
OR
{{#invoke:String|find|source=source_str|target=target_str|start=start_index|plain

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 Lua style regular expression, defaults to true

If invoked using named parameters, Mediawiki will automatically remove any leadin
trailing whitespace from the parameter.  In some circumstances this is desirable,
other cases one may want to preserve the whitespace.

This function returns the first index >= "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.
]]
function str.find( frame )
        local new_args = str._getParameters( frame.args, {'source', 'target', 'st
        local source_str = new_args['source'] or ''
        local pattern = new_args['target'] or ''
        local start_pos = tonumber(new_args['start']) or 1
        local plain = new_args['plain'] or true

        if source_str == '' or pattern == '' then
                return 0
        end

        plain = str._getBoolean( plain )

        local start = mw.ustring.find( source_str, pattern, start_pos, plain )
        if start == nil then
                start = 0
        end

        return start
end

--[[
replace

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

Usage:
{{#invoke:String|replace|source_str|pattern_string|replace_string|replacement_cou
OR
{{#invoke:String|replace|source=source_string|pattern=pattern_string|replace=repl
   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 occurences to replace, defaults to all.
    plain: Boolean flag indicating that pattern should be understood as plain
        text and not as a Lua style regular expression, defaults to true
]]
function str.replace( frame )
        local new_args = str._getParameters( frame.args, {'source', 'pattern', 'r
        local source_str = new_args['source'] or ''
        local pattern = new_args['pattern'] or ''
        local replace = new_args['replace'] or ''
        local count = tonumber( new_args['count'] )
        local plain = new_args['plain'] or true

        if source_str == '' or pattern == '' then
                return source_str
        end
        plain = str._getBoolean( plain )

        if plain then
                pattern = str._escapePattern( pattern )
                replace = mw.ustring.gsub( replace, "%%", "%%%%" ) --Only need to
        end

        local result

        if count ~= nil then
                result = mw.ustring.gsub( source_str, pattern, replace, count )
        else
                result = mw.ustring.gsub( source_str, pattern, replace )
        end
        return result
end

--[[
    simple function to pipe string.rep to templates.
]]
function str.rep( frame )
        local repetitions = tonumber( frame.args[2] )
        if not repetitions then
                return str._error( 'function rep expects a number as second param
        end
        return string.rep( frame.args[1] or '', repetitions )
end

--[[
escapePattern

This function escapes special characters from a Lua string pattern. See [1]
for details on how patterns work.

[1] https://www.mediawiki.org/wiki/Extension:Scribunto/Lua_reference_manual#Patte

Usage:
{{#invoke:String|escapePattern|pattern_string}}

Parameters
    pattern_string: The pattern string to escape.
]]
function str.escapePattern( frame )
        local pattern_str = frame.args[1]
        if not pattern_str then
                return str._error( 'No pattern string specified' )
        end
        local result = str._escapePattern( pattern_str )
        return result
end

--[[
count
This function counts the number of occurrences of one string in another.
]]
function str.count(frame)
        local args = str._getParameters(frame.args, {'source', 'pattern', 'plain'
        local source = args.source or ''
        local pattern = args.pattern or ''
        local plain = str._getBoolean(args.plain or true)
        if plain then
                pattern = str._escapePattern(pattern)
        end
        local _, count = mw.ustring.gsub(source, pattern, '')
        return count
end

--[[
endswith
This function determines whether a string ends with another string.
]]
function str.endswith(frame)
        local args = str._getParameters(frame.args, {'source', 'pattern'})
        local source = args.source or ''
        local pattern = args.pattern or ''
        if pattern == '' then
                -- All strings end with the empty string.
                return "yes"
        end
        if mw.ustring.sub(source, -mw.ustring.len(pattern), -1) == pattern then
                return "yes"
        else
                return ""
        end
end

--[[
join

Join all non empty arguments together; the first argument is the separator.
Usage:
{{#invoke:String|join|sep|one|two|three}}
]]
function str.join(frame)
        local args = {}
        local sep
        for _, v in ipairs( frame.args ) do
                if sep then
                        if v ~= '' then
                                table.insert(args, v)
                        end
                else
                        sep = v
                end
        end
        return table.concat( args, sep or '' )
end

--[[
Helper function that populates the argument list given that user may need to use
named and unnamed parameters.  This is relevant because named parameters are not
identical to unnamed parameters due to string trimming, and when dealing with st
we sometimes want to either preserve or remove that whitespace depending on the 
]]
function str._getParameters( frame_args, arg_list )
        local new_args = {}
        local index = 1
        local value

        for _, arg in ipairs( arg_list ) do
                value = frame_args[arg]
                if value == nil then
                        value = frame_args[index]
                        index = index + 1
                end
                new_args[arg] = value
        end

        return new_args
end

--[[
Helper function to handle error messages.
]]
function str._error( error_str )
        local frame = mw.getCurrentFrame()
        local error_category = frame.args.error_category or 'Errors reported by 
        local ignore_errors = frame.args.ignore_errors or false
        local no_category = frame.args.no_category or false

        if str._getBoolean(ignore_errors) then
                return ''
        end

        local error_str = '<strong class="error">String Module Error: ' .. error_
        if error_category ~= '' and not str._getBoolean( no_category ) then
                error_str = '[[Category:' .. error_category .. ']]' .. error_str
        end

        return error_str
end

--[[
Helper Function to interpret boolean strings
]]
function str._getBoolean( boolean_str )
        local boolean_value

        if type( boolean_str ) == 'string' then
                boolean_str = boolean_str:lower()
                if boolean_str == 'false' or boolean_str == 'no' or boolean_str =
                                or boolean_str == '' then
                        boolean_value = false
                else
                        boolean_value = true
                end
        elseif type( boolean_str ) == 'boolean' then
                boolean_value = boolean_str
        else
                error( 'No boolean value found' )
        end
        return boolean_value
end

--[[
Helper function that escapes all pattern characters so that they will be treated
as plain text.
]]
function str._escapePattern( pattern_str )
        return mw.ustring.gsub( pattern_str, "([%(%)%.%%%+%-%*%?%[%^%$%]])", "%%
end

return str
```