

Inhaltsverzeichnis

- [1. Modul:Escape](#)
- [2. Modul:Escape/Doku](#)
- [3. Modul:Escape/testcases](#)

Modul:Escape

Inhaltsverzeichnis

- [1 Usage](#)
 - [1.1 Shortcut](#)
 - [1.2 Caveats](#)
- [2 Speed](#)
- [3 Examples](#)
 - [3.1 Template](#)
 - [3.2 Module](#)

Usage

This module is designed as an way to escape strings in a customized and efficient manner. It works by replacing characters that are preceded by your escape char (or phrase) There are two ways to call this module:

From another module:

```
local esc = require('Module:Escape')
esc:char(Vorlage:Green)
local to_escape = esc:text(Vorlage:Green)
Vorlage:Green
local result = esc:undo(to_escape)
```

From a template:

```
{{invoke:Escape|main|mode=Vorlage:Green|char=Vorlage:Green|Vorlage:Green}}
```

In a template, the most useful function is [Vorlage:Code](#).

This module is primarily intended to be used by other modules. However all functions can be called in template space using [Vorlage:Para](#) followed by arguments.

All module functions (i.e. any func. other than main()) should be called using a colon (:), e.g. [Vorlage:Code](#) or `esc:kill{{{example|\\}}}', '}' == '{{example|}'`

[Vorlage:TOC](#)
[tab](#)

This function takes only one argument: A string. All characters in this string which are preceded by the sequence set by `escape:char()` will be replaced with placeholders that can be converted back into that char by `escape:undo()`

[Vorlage:TOC](#)
[tab](#)

Takes two arguments:

1. The string that may contain placeholders set by `escape:text()`
2. Optional, a char to be placed in front of any characters that have been de-escaped. (i.e. if you need to re-escape those string with a different char)

[Vorlage:TOC](#)
[tab](#)

This is basically equivalent to calling `string.gsub()` on the string returned by `escape:text()` and feeding that result into `escape:undo()` in a single step. Takes three arguments:

1. A string
2. A sequence of characters to be removed from that string. (May use a string. gsub pattern)
3. Optional, a char to be placed in front of any characters that have been de-escaped.

[Vorlage:TOC](#)
[tab](#)

This function's primary use is to initialize the patterns to scan a string for an escape/escaped sequence. It takes two arguments, the first being the escape character and the second being a table of arguments (optional). By default, this module will escape the [Vorlage:Code](#) char. To escape the [Vorlage:Code](#) char instead, you can do [Vorlage:Code](#) (or [Vorlage:Code](#) (presuming you stored the table returned by this module in the local variable [Vorlage:Code](#)).

When called without the second argument, `char()` will return a table containing the functions. This allows, for example, `escape:char('*'):kill('1*23', '%d')` which would return '2'

For the most part, there is very little reason to set [Vorlage:Para](#) in template space since the patterns it stores are not shared with other invocations of this module. Templates should instead use the [Vorlage:Para](#) if a new escape sequence is desired.

Shortcut

If provided a second argument that is a table containing a {key = value} pair, such that the key is [Vorlage:Code](#), [Vorlage:Code](#), or [Vorlage:Code](#) and the value is a table containing the arguments that would have been passed to those functions. For `escape:undo()`, will cause the `escape:text()` and `escape:kill()`

Caveats

- When using a multi-character escape sequence, this module only marks it using the byte value of the first character. Thus, `escape:undo()` will unescape, for example, all characters escaped with 'e' and 'esc' if both were used. In practice however this shouldn't be a problem as multiple escape sequences are pretty rare unless you're transitioning between multiple code languages. (Multiple multi-char escape sequences beginning with the same character are simply bad practice anyhow.)
- Since byte values are stored as numbers, it is not recommended for you to use a number as an escape sequence (though it may work just fine).
- Placeholder byte values separated with return ('\r') characters--chosen because they are seldom used at all, and virtually never used unpaired with '\n'; moreover, it is distinct from the markers generated by [Vorlage:Tag](#) or [Vorlage:Code](#) (which use the delete char). To set a different separator char, include the key-value pair `{safeChr = Vorlage:Green}` in the table that you pass to `escape:char()`.

Speed

The following are benchmarks...

when executing the following module function:

```
function p.test_kill1500(frame)
  local esc = require('Module:Escape')
  for k = 1, 500 do
    local v = esc:kill(p.test_string2(), 'test')
  end
  return os.clock(esc)
end
```

0.04626

when repeating the following line 500 times in a template:

[Vorlage:Code](#)

0.767

All times in seconds. The module time x500 was calculated when you loaded this doc page (normally between 0.02 and 0.07). The template time x500 was recorded on Jan 15, 2015.

Examples

Template

[Modul Diskussion:Escape/testcases](#)

Module

Here's some sample output from the debug consol below the module editor:

Vorlage:Blue

test, \test, \7b 044 7btest\\ || |||

Vorlage:Blue

test, 5c 0116 5cest, 5c 055 5cb 044 7btest5c 092 5c 5c 092 5c 5c 092 5c5c 092 5c

Vorlage:Blue

test, 5c 0116 5cest, 5c 055 5cb 044 7btest5c 092 5c 5c 092 5c 5c 092 5c5c 092 5c

Vorlage:Blue

test, test, 7b 044 7btest\\ ||

Vorlage:Blue

true

Vorlage:BlueVorlage:Green

test, 5c 0116 5cest, 5c 055 5cb 044 7btest5c 092 5c 5c 092 5c 5c 092 5c5c 092 5c

Vorlage:Blue

test, \test, \,test\\ || |||

Vorlage:Blue

test, test, 7b 044 7btest\\ ||

Vorlage:Blue

test, test, {,test\\ ||

Vorlage:Blue

false

Vorlage:Blue

true

Vorlage:Blue

test { test {\{ test, \test, \{,test\\ \ \ {\

Vorlage:Blue

test test { test, test, {,test \

Vorlage:Blue

true

```

local escape = {
  char = function(self, chr, args)
    args = args or {}
    local safe = args.safeChr or string.char(13)
    chr = tostring(chr or '\\')
    self[1] = ('%s0%%s%s'):format(
      ('%x%s%s'):format(chr:byte(), safe, safe),
      ('%s%x'):format(safe, chr:byte())
    )
    if not self[self[1]] then
      self[self[1]] = {
        char = chr,
        text = ('%s(.)'):format(chr),
        undo = self[1]:format('%d+')
      }
    end
    return args.text and self:text(args.text)
      or args.undo and self:undo(args.undo, chr)
      or args.kill and self:kill(args.kill)
      or self
  end,
  exec = function(self, text, mode, newEscape)
    local target = self[self[1] or self:char() and self[1]]
    for v in text:gfind(target[mode]) do
      text = text:gsub(
        mode == 'text' and
          ('%s%s'):format(target.char, v:gsub('%W', '%%%1'))
          or self[1]:format(v),
        mode == 'text' and
          self[1]:format(v:byte())
          or (newEscape or '') .. v:char()
      )
    end
    return text
  end,
  text = function(self, text)
    return self:exec(type(text) == 'table' and text[1] or text, 'text')
  end,
  undo = function(self, text, newEscape)
    if type(text) == 'table' then
      text, newEscape = unpack(text)
    end
    return self:exec(text, 'undo', newEscape)
  end,
  kill = function(self, text, chars, newEscape)
    if type(text) == 'table' then
      text, chars, newEscape = unpack(text)
    end
    return self:undo(self:text(text):gsub(chars or '', ''), newEscape)
  end
}

function escape.main(frame)
  local args, family = {}, {frame:getParent(), frame}
  for f = 1, 2 do
    for k, v in pairs(family[f] and family[f].args or {}) do
      args[k] = args[k] or v:match('^%s*(.-)%s*$')
    end
  end
  if args.mode == 'char' then
    return escape:char(args.char or args[2], args)
  end
end

```

```
        end
        return escape[args.mode](escape:char(args.char), args)
end

return escape
```

Modul:Escape/Doku

Dies ist die Dokumentationsseite für [Modul:Escape](#)

Inhaltsverzeichnis

- [1 Usage](#)
 - [1.1 Shortcut](#)
 - [1.2 Caveats](#)
- [2 Speed](#)
- [3 Examples](#)
 - [3.1 Template](#)
 - [3.2 Module](#)

Usage

This module is designed as an way to escape strings in a customized and efficient manner. It works by replacing characters that are preceded by your escape char (or phrase) There are two ways to call this module:

From another module:

```
local esc = require('Module:Escape')
esc:char(Vorlage:Green)
local to_escape = esc:text(Vorlage:Green)
Vorlage:Green
local result = esc:undo(to_escape)
```

From a template:

```
{{invoke:Escape|main|mode=Vorlage:Green|char=Vorlage:Green|Vorlage:Green}}
```

In a template, the most useful function is [Vorlage:Code](#).

This module is primarily intended to be used by other modules. However all functions can be called in template space using [Vorlage:Para](#) followed by arguments.

All module functions (i.e. any func. other than main()) should be called using a colon (:), e.g. [Vorlage:Code](#) or `esc:kill('{{example|\\}}', ' ') == '{{example|}}`

[Vorlage:TOC](#) [tab](#)

This function takes only one argument: A string. All characters in this string which are preceded by the sequence set by `escape:char()` will be replaced with placeholders that can be converted back into that char by `escape:undo()`

[Vorlage:TOC](#) [tab](#)

Takes two arguments:

1. The string that may contain placeholders set by `escape:text()`
2. Optional, a char to be placed in front of any characters that have been de-escaped. (i.e. if you need to re-escape those string with a different char)

[Vorlage:TOC](#) [tab](#)

This is basically equivalent to calling `string.gsub()` on the string returned by `escape:text()` and feeding that result into `escape:undo()` in a single step. Takes three arguments:

1. A string
2. A sequence of characters to be removed from that string. (May use a string, `gsub` pattern)
3. Optional, a char to be placed in front of any characters that have been de-escaped.

[Vorlage:TOC](#) [tab](#)

This function's primary use is to initialize the patterns to scan a string for an escape/escaped sequence. It takes two arguments, the first being the escape character and the second being a table of arguments (optional). By default, this module will escape the [Vorlage:Code](#) char. To escape the [Vorlage:Code](#) char instead, you can do [Vorlage:Code](#) (or [Vorlage:Code](#) (presuming you stored the table returned by this module in the local variable [Vorlage:Code](#)).

When called without the second argument, `char()` will return a table containing the functions. This allows, for example, `escape:char('*'):kill('1*23', '%d')` which would return '2'

For the most part, there is very little reason to set [Vorlage:Para](#) in template space since the patterns it stores are not shared with other invocations of this module. Templates should instead use the [Vorlage:Para](#) if a new escape sequence is desired.

Shortcut

If provided a second argument that is a table containing a {key = value} pair, such that the key is [Vorlage:Code](#), [Vorlage:Code](#), or [Vorlage:Code](#) and the value is a table containing the arguments that would have been passed to those functions. For `escape:undo()`, will cause the `escape:text()` and `escape:kill()`

Caveats

- When using a multi-character escape sequence, this module only marks it using the byte value of the first character. Thus, `escape:undo()` will unescape, for example, all characters escaped with 'e' and 'esc' if both were used. In practice however this shouldn't be a problem as multiple escape sequences are pretty rare unless you're transitioning between multiple code languages. (Multiple multi-char escape sequences beginning with the same character are simply bad practice anyhow.)
- Since byte values are stored as numbers, it is not recommended for you to use a number as an escape sequence (though it may work just fine).
- Placeholder byte values separated with return ('\r') characters--chosen because they are seldom used at all, and virtually never used unpaired with '\n'; moreover, it is distinct from the markers generated by [Vorlage:Tag](#) or [Vorlage:Code](#) (which use the delete char). To set a different separator char, include the key-value pair `{safeChr = Vorlage:Green}` in the table that you pass to `escape:char()`.

Speed

The following are benchmarks...

when executing the following module function:

```
function p.test_kill500(frame)
  local esc = require('Module:Escape')
  for k = 1, 500 do
    local v = esc:kill(p.test_string2(), 'test')
  end
  return os.clock(esc)
end
```

0.05628

when repeating the following line 500 times in a template:

[Vorlage:Code](#)

0.767

All times in seconds. The module time x500 was calculated when you loaded this doc page (normally between 0.02 and 0.07). The template time x500 was recorded on Jan 15, 2015.

Examples

Template

[Modul Diskussion:Escape/testcases](#)

Module

Here's some sample output from the debug consol below the module editor:

Vorlage:Blue

test, \test, \7b 044 7btest\\ || |||

Vorlage:Blue

test, 5c 0116 5cest, 5c 055 5cb 044 7btest5c 092 5c 5c 092 5c 5c 092 5c5c 092 5c

Vorlage:Blue

test, 5c 0116 5cest, 5c 055 5cb 044 7btest5c 092 5c 5c 092 5c 5c 092 5c5c 092 5c

Vorlage:Blue

test, test, 7b 044 7btest\\ ||

Vorlage:Blue

true

Vorlage:BlueVorlage:Green

test, 5c 0116 5cest, 5c 055 5cb 044 7btest5c 092 5c 5c 092 5c 5c 092 5c5c 092 5c

Vorlage:Blue

test, \test, \,test\\ || |||

Vorlage:Blue

test, test, 7b 044 7btest\\ ||

Vorlage:Blue

test, test, {,test\\ ||

Vorlage:Blue

false

Vorlage:Blue

true

Vorlage:Blue

test { test {\{ test, \test, \{,test\\ || {\

Vorlage:Blue

test test { test, test, {,test \

Vorlage:Blue

true

Modul:Escape/testcases

Die Dokumentation für dieses Modul kann unter [Modul:Escape/testcases/Doku](#) erstellt werden

```
p = {}

function p.test_string()
    return 'test, \\test, \\{,test\\\\\\\\ \\\\ \\\\\\\\\\\\\\\\\\\\'
end

function p.test_string2()
    return 'test { test {\\{ test, \\test, \\{,test\\\\ \\\\ \\\\ {\\{\\'
end

function p.test_string_module(frame)
    return (p['test_string' .. (frame.args[1] or '')]():gsub('\\', '\\\\\\\\'))
end

function p.test_kill500(frame)
    local esc = require('Module:Escape')
    for k = 1, 500 do
        local v = esc:kill(p.test_string2(), 'test')
    end
    return os.clock(esc)
end

return p
```