

# Modul:Shortcut

---

This module makes a box showing the shortcut links to a page.

Inhaltsverzeichnis	
1 Usage .....	1
1.1 From wikitext .....	1
1.2 From Lua .....	1
2 Technical details .....	1

## Usage

---

### From wikitext

---

From wikitext, this module should be called from a template, usually Vorlage:TI. Please see the template page for documentation. However, it can also be called using the syntax `{{#invoke:shortcut|main|arguments}}`.

### From Lua

---

To use this module from Lua, first load it.

```
local mShortcut = require('Module:Shortcut')
```

Then you can create shortcut boxes with the following syntax:

```
mShortcut._main(shortcuts, options, frame, cfg)
```

- *shortcuts* is an array of shortcut page names. (required)
- *options* is a table of options. The following keys are supported:
  - *msg* - a message to leave after the list of shortcuts.
  - *category* - if set to false (or a value regarded as false by Module:Yesno, such as "no"), categories are suppressed.
- *frame* a frame object. This is optional, and only intended to be used internally.
- *cfg* a table of config values. This is optional, and is only intended for testing.

## Technical details

---

This module has a configuration file at Module:Shortcut/config. It can be used to translate this module into different languages or to change details like category names.



```
-- This module implements {{shortcut}}.

-- Set constants
local CONFIG_MODULE = 'Module:Shortcut/config'

-- Load required modules
local checkType = require('libraryUtil').checkType
local yesno = require('Module:Yesno')

local p = {}

local function message(msg, ...)
    return mw.message.newRawMessage(msg, ...):plain()
end

local function makeCategoryLink(cat)
    return string.format('[[:%s:%s]]', mw.site.namespaces[14].name, cat)
end

function p._main(shortcuts, options, frame, cfg)
    checkType('_main', 1, shortcuts, 'table')
    checkType('_main', 2, options, 'table', true)
    options = options or {}
    frame = frame or mw.getCurrentFrame()
    cfg = cfg or mw.loadData(CONFIG_MODULE)
    local templateMode = options.template and yesno(options.template)
    local redirectMode = options.redirect and yesno(options.redirect)
    local isCategorized = not options.category or yesno(options.category) ~=

    -- Validate shortcuts
    for i, shortcut in ipairs(shortcuts) do
        if type(shortcut) ~= 'string' or #shortcut < 1 then
            error(message(cfg['invalid-shortcut-error'], i), 2)
        end
    end

    -- Make the list items. These are the shortcuts plus any extra lines such
    -- as options.msg.
    local listItems = {}
    for i, shortcut in ipairs(shortcuts) do
        local templatePath, prefix
        if templateMode then
            -- Namespace detection
            local titleObj = mw.title.new(shortcut, 10)
            if titleObj.namespace == 10 then
                templatePath = titleObj.fullText
            else
                templatePath = shortcut
            end
            prefix = options['pre' .. i] or options.pre or ''
        end
        if options.target and yesno(options.target) then
            listItems[i] = templateMode
                and string.format("&#123;&#123;%s[[%s|%s]]&#125;&#125;"
                    or string.format("[[:%s]]", shortcut)
        else
            listItems[i] = frame:expandTemplate{
                title = 'No redirect',
                args = templateMode and {templatePath, shortcut}
            }
            if templateMode then
                listItems[i] = string.format("&#123;&#123;%s&#125;&#125;"
                    or string.format("[[:%s]]", shortcut)
            end
        end
    end
end
```

```
end
table.insert(listItems, options.msg)

-- Return an error if we have nothing to display
if #listItems < 1 then
    local msg = cfg['no-content-error']
    msg = string.format('<strong class="error">%s</strong>', msg)
    if isCategorized and cfg['no-content-error-category'] then
        msg = msg .. makeCategoryLink(cfg['no-content-error-cate
    end
    return msg
end

local root = mw.html.create()
root:wikitext(frame:extensionTag{ name = 'templatestyles', args = { src =
-- Anchors
local anchorDiv = root
    :tag('div')
        :addClass('module-shortcutanchordiv')
for i, shortcut in ipairs(shortcuts) do
    local anchor = mw.uri.anchorEncode(shortcut)
    anchorDiv:tag('span'):attr('id', anchor)
end

-- Shortcut heading
local shortcutHeading
do
    local nShortcuts = #shortcuts
    if nShortcuts > 0 then
        local headingMsg = options['shortcut-heading'] or
            redirectMode and cfg['redirect-heading'] or
            cfg['shortcut-heading']
        shortcutHeading = message(headingMsg, nShortcuts)
        shortcutHeading = frame:preprocess(shortcutHeading)
    end
end

-- Shortcut box
local shortcutList = root
    :tag('div')
        :addClass('module-shortcutboxplain plainlist noprint')
        :attr('role', 'note')
if options.float and options.float:lower() == 'left' then
    shortcutList:addClass('module-shortcutboxleft')
end
if options.clear and options.clear ~= '' then
    shortcutList:css('clear', options.clear)
end
if shortcutHeading then
    shortcutList
        :tag('div')
            :addClass('module-shortcutlist')
            :wikitext(shortcutHeading)
end
local list = shortcutList:tag('ul')
for i, item in ipairs(listItems) do
    list:tag('li'):wikitext(item)
end
return tostring(root)
end

function p.main(frame)
    local args = require('Module:Arguments').getArgs(frame)
```

```
-- Separate shortcuts from options
local shortcuts, options = {}, {}
for k, v in pairs(args) do
    if type(k) == 'number' then
        shortcuts[k] = v
    else
        options[k] = v
    end
end

-- Compress the shortcut array, which may contain nils.
local function compressArray(t)
    local nums, ret = {}, {}
    for k in pairs(t) do
        nums[#nums + 1] = k
    end
    table.sort(nums)
    for i, num in ipairs(nums) do
        ret[i] = t[num]
    end
    return ret
end

shortcuts = compressArray(shortcuts)

return p._main(shortcuts, options, frame)
end

return p
```