# Modul:Navbar

Vorlage:Lua

This is a Lua implementation of Vorlage:Tl. It is used in Module:Navbox.

```
local p = {}
local cfg = mw.loadData('Module:Navbar/configuration')

local function get_title_arg(is_collapsible, template)
        local title_arg = 1
        if is_collapsible then title_arg = 2 end
        if template then title_arg = 'template' end
        return title_arg
end

local function choose_links(template, args)
        -- The show table indicates the default displayed items.
        -- view, talk, edit, hist, move, watch
        -- TODO: Move to configuration.
        local show = {true, true, true, false, false, false}
        if template then
                show[2] = false
                show[3] = false
                local index = {t = 2, d = 2, e = 3, h = 4, m = 5, w = 6,
                        talk = 2, edit = 3, hist = 4, move = 5, watch = 6}
                -- TODO: Consider removing TableTools dependency.
                for _, v in ipairs(require ('Module:TableTools').compressSparseAr
                        local num = index[v]
                        if num then show[num] = true end
                end
        end

        local remove_edit_link = args.noedit
        if remove_edit_link then show[3] = false end

        return show

end

local function add_link(link_description, ul, is_mini, font_style)
        local l
        if link_description.url then
                l = {'[', '', ']'}
        else
                l = {'[[', '|', ']]'}
        end
        ul:tag('li')
                :addClass('nv-' .. link_description.full)
                :wikitext(l[1] .. link_description.link .. l[2])
                :tag(is_mini and 'abbr' or 'span')
                        :attr('title', link_description.html_title)
                        :cssText(font_style)
                        :wikitext(is_mini and link_description.mini or link_descr
                        :done()
                :wikitext(l[3])
                :done()
end
```

```
local function make_list(title_text, has_brackets, displayed_links, is_mini, font

        local title = mw.title.new(mw.text.trim(title_text), cfg.title_namespace)
        if not title then
                error(cfg.invalid_title .. title_text)
        end
        local talkpage = title.talkPageTitle and title.talkPageTitle.fullText or

        -- TODO: Get link_descriptions and show into the configuration module.
        -- link_descriptions should be easier...
        local link_descriptions = {
                { ['mini'] = 'v', ['full'] = 'view', ['html_title'] = 'View this
                        ['link'] = title.fullText, ['url'] = false },
                { ['mini'] = 't', ['full'] = 'talk', ['html_title'] = 'Discuss th
                        ['link'] = talkpage, ['url'] = false },
                { ['mini'] = 'e', ['full'] = 'edit', ['html_title'] = 'Edit this
                        ['link'] = title:fullUrl('action=edit'), ['url'] = true },
                { ['mini'] = 'h', ['full'] = 'hist', ['html_title'] = 'History of
                        ['link'] = title:fullUrl('action=history'), ['url'] = tru
                { ['mini'] = 'm', ['full'] = 'move', ['html_title'] = 'Move this
                        ['link'] = mw.title.new('Special:Movepage'):fullUrl('targ
                { ['mini'] = 'w', ['full'] = 'watch', ['html_title'] = 'Watch thi
                        ['link'] = title:fullUrl('action=watch'), ['url'] = true
        }

        local ul = mw.html.create('ul')
        if has_brackets then
                ul:addClass(cfg.classes.brackets)
                        :cssText(font_style)
        end

        for i, _ in ipairs(displayed_links) do
                if displayed_links[i] then add_link(link_descriptions[i], ul, is_
        end
        return ul:done()

end

function p._navbar(args)

        -- TODO: We probably don't need both fontstyle and fontcolor...
        local font_style = args.fontstyle
        local font_color = args.fontcolor
        local is_collapsible = args.collapsible
        local is_mini = args.mini
        local is_plain = args.plain

        local collapsible_class = nil
        if is_collapsible then
                collapsible_class = cfg.classes.collapsible
                if not is_plain then is_mini = 1 end
                if font_color then
                        font_style = (font_style or '') .. '; color: ' .. font_co
                end
        end

        local navbar_style = args.style
        local div = mw.html.create():tag('div')
        div
                :addClass(cfg.classes.navbar)
                :addClass(cfg.classes.plainlinks)
                :addClass(cfg.classes.horizontal_list)
                :addClass(collapsible_class) -- we made the determination earlier
                :cssText(navbar_style)
```

```
        if is_mini then div:addClass(cfg.classes.mini) end

        local box_text = (args.text or cfg.box_text) .. ' '
         -- the concatenated space guarantees the box text is separated
        if not (is_mini or is_plain) then
                div
                        :tag('span')
                                :addClass(cfg.classes.box_text)
                                :cssText(font_style)
                                :wikitext(box_text)
        end

        local template = args.template
        local displayed_links = choose_links(template, args)
        local has_brackets = args.brackets
        local title_arg = get_title_arg(is_collapsible, template)
        local title_text = args[title_arg] or (':' .. mw.getCurrentFrame():getPar
        local list = make_list(title_text, has_brackets, displayed_links, is_mini
        div:node(list)

        if is_collapsible then
                local title_text_class
                if is_mini then
                        title_text_class = cfg.classes.collapsible_title_mini
                else
                        title_text_class = cfg.classes.collapsible_title_full
                end
                div:done()
                        :tag('div')
                        :addClass(title_text_class)
                        :cssText(font_style)
                        :wikitext(args[1])
        end

        return mw.getCurrentFrame():extensionTag{
                name = 'templatestyles', args = { src = cfg.templatestyles }
        } .. tostring(div:done())
end

function p.navbar(frame)
        return p._navbar(require('Module:Arguments').getArgs(frame))
end

return p
```