Modul:Clickable button 2/Doku

Ausgabe: 10.11.2026
Letzte Änderung: 11.02.2022

Inhaltsverzeichnis

- 1. Modul:Clickable button 2/Doku
- 2. Modul:Clickable button 2

Modul:Clickable button 2/Doku

Dies ist die Dokumentationsseite für Modul:Clickable button 2

This module implements the Vorlage:Tl template.

Usage from wikitext

To use this template from wikitext, you should normally use the Vorlage:Tl template. However, it can also be used with the syntax `{{#invoke:Clickable button 2|main|`*`args`*`}}`. Please see the template page for a list of available parameters.

Usage from Lua modules

To use this module from other Lua modules, first load the module.

```
local mClickableButton2 = require('Module:Clickable button 2')
```

You can then generate a button using the luaMain function.

```
mClickableButton2.luaMain(args)
```

The *args* variable should be a table containing the arguments to pass to the module. To see the different arguments that can be specified and how they affect the module output, please refer to the Vorlage:Tl template documentation.

Modul:Clickable button 2

This module implements the Vorlage:Tl template.

Usage from wikitext

To use this template from wikitext, you should normally use the Vorlage:Tl template. However, it can also be used with the syntax `{{#invoke:Clickable button 2|main|`*`args`*`}}`. Please see the template page for a list of available parameters.

Usage from Lua modules

To use this module from other Lua modules, first load the module.

```
local mClickableButton2 = require('Module:Clickable button 2')
```

You can then generate a button using the luaMain function.

```
mClickableButton2.luaMain(args)
```

The *args* variable should be a table containing the arguments to pass to the module. To see the different arguments that can be specified and how they affect the module output, please refer to the Vorlage:Tl template documentation.

```
-- This module implements {{clickable button 2}}.

local yesno = require('Module:Yesno')

local p = {}

function p.main(frame)
	local args = require('Module:Arguments').getArgs(frame, {
		wrappers = 'Template:Clickable button 2'
	})
	return p.luaMain(args)
end

function p.luaMain(args)
	-- If first arg or a url is not provided,
	-- but we have a second arg, make a button.
	-- Otherwise, return nothing.
	if not args[1] and not args.url then
		if args[2] then
			p.nolink = true
		else
			return ''
		end
	end

	local data = p.makeLinkData(args)
	local link = p.renderLink(data)
	local trackingCategories = p.renderTrackingCategories(args)
	return link .. trackingCategories
end

function p.makeLinkData(args)
	local data = {}

	-- Get the link and display values,
	-- and find whether we are outputting
	-- a wikilink or a URL.
```

```
if args.url then
	data.isUrl = true
	data.link = args.url
	if args[1] then
		data.display = args[1]
	else if args[2] then
		data.display = args[2]
	else
		data.display = args.url
		p.urlisdisplay = true
		end
	end
else
	data.isUrl = false
	p.urlisdisplay = false
	data.link = args[1]
	if args[2] then
		data.display = args[2]
	else
		data.display = args[1]
	end
end

if yesno(args.link) == false then
	p.nolink = true
end

-- Colours
-- For the merge with {{clickable button}}
local colour = args.color and args.color:lower()

-- Classes
local class = args.class and args.class:lower()
data.classes = {}
if class == 'ui-button-green'
	or class == 'ui-button-blue'
	or class == 'ui-button-red'
then
	table.insert(
		data.classes,
		'submit ui-button ui-widget ui-state-default ui-corner-all'
			.. ' ui-button-text-only ui-button-text'
	)
else
	table.insert(data.classes, 'mw-ui-button')
end

--If class is unset,
--then let color determine class
if not class then
	if colour == 'blue' then
		class = 'mw-ui-progressive'
		else if colour == 'red' then
			class = 'mw-ui-destructive'
			else if colour == 'green' then
				class = 'mw-ui-constructive'
			end
		end
	end
end

if class then
	table.insert(data.classes, class)
end

-- Styles
do
```

```
		--[[
		-- Check whether we are on the same page as we have specified in
		-- args[1], but not if we are using a URL link, as then args[1] is only
		-- a display value. If we are currently on the page specified in
		-- args[1] make the button colour darker so that it stands out from
		-- other buttons on the page.
		--]]
		local success, linkTitle, currentTitle
		if not data.isUrl then
			currentTitle = mw.title.getCurrentTitle()
			success, linkTitle = pcall(mw.title.new, args[1])
		elseif p.urlisdisplay then
			currentTitle = mw.title.getCurrentTitle()
		end
		if success
			and linkTitle
			and mw.title.equals(currentTitle, linkTitle)
			and not p.urlisdisplay
		then
			if class == 'ui-button-blue'
				or class == 'mw-ui-progressive'
				or class == 'mw-ui-constructive'
			then
				data.backgroundColor = '#2962CB'
				data.color = '#fff'
			elseif class == 'ui-button-green' then
				data.backgroundColor = '#008B6D'
			elseif class == 'ui-button-red' or class == 'mw-ui-destructive' th
				data.backgroundColor = '#A6170F'
			else
				data.backgroundColor = '#CCC'
				data.color = '#666'
			end
		else
			if p.urlisdisplay
			then
				data.dummyLink = tostring(currentTitle)
			end
		end
		-- Add user-specified styles.
		data.style = args.style
	end

	return data
end

function p.renderLink(data)
	-- Render the display span tag.
	local display
	do
		local displaySpan = mw.html.create('span')
		for i, class in ipairs(data.classes or {}) do
			displaySpan:addClass(class)
		end
		displaySpan
			:css{
				['background-color'] = data.backgroundColor,
				color = data.color
			}
		if data.style then
			displaySpan:cssText(data.style)
		end
		displaySpan:wikitext(data.display)
		display = tostring(displaySpan)
	end

	-- Render the link
```

```
        local link
        if p.nolink then
                if p.urlisdisplay then
                        link = string.format('[[%s|%s]]', data.dummyLink, display)
                        else
                                link = string.format('%s', display)
                        end
                else if data.isUrl then
                        link = string.format('[%s %s]', data.link, display)
                else
                        link = string.format('[[%s|%s]]', data.link, display)
                end
        end

        return string.format('<span class="plainlinks clickbutton">%s</span>', link)
end

function p.renderTrackingCategories(args)
        if yesno(args.category) == false then
                return ''
        end
        local class = args.class and args.class:lower()
        if class == 'ui-button-green'
                or class == 'ui-button-blue'
                or class == 'ui-button-red'
        then
                return '[[Category:Pages using old style ui-button-color]]'
        else
                return ''
        end
end

return p
```