# Inhaltsverzeichnis


- 1. Modul:Uses TemplateStyles/Doku
- 2. Modul:Uses TemplateStyles


# Modul:Uses TemplateStyles/Doku

**Dies ist die Dokumentationsseite für Modul:Uses TemplateStyles**

Vorlage:Lua

Implements Vorlage:Tlx.

[[Category:Module documentation pages{{#translation:}}]]

# Modul:Uses TemplateStyles

Vorlage:Lua

Implements Vorlage:Tlx.

[[Category:Global modules{{#translation:}}]]

---

```
-- This module implements the {{Uses TemplateStyles}} template.
local yesno = require('Module:Yesno')
local mList = require('Module:List')
local mTableTools = require('Module:TableTools')
local mMessageBox = require('Module:Message box')
local TNT = require('Module:TNT')

local p = {}

local function format(msg, ...)
	return TNT.format('I18n/Uses TemplateStyles', msg, ...)
end

local function getConfig()
	return mw.loadData('Module:Uses TemplateStyles/config')
end

function p.main(frame)
	local origArgs = frame:getParent().args
	local args = {}
```

```
	for k, v in pairs(origArgs) do
		v = v:match('^%s*(.-)%s*$')
		if v ~= '' then
			args[k] = v
		end
	end
	return p._main(args)
end

function p._main(args, cfg)
	local tStyles = mTableTools.compressSparseArray(args)
	local box = p.renderBox(tStyles)
	local trackingCategories = p.renderTrackingCategories(args, tStyles)
	return box .. trackingCategories
end

function p.renderBox(tStyles)
	local boxArgs = {}
	if #tStyles < 1 then
		boxArgs.text = string.format('<strong class="error">%s</strong>', format('error-en
	else
		local cfg = getConfig()
		local tStylesLinks = {}
		for i, ts in ipairs(tStyles) do
			local link = string.format('[[:%s]]', ts)
			local sandboxLink = nil
			local tsTitle = mw.title.new(ts)
			if tsTitle and cfg['sandbox_title'] then
				local tsSandboxTitle = mw.title.new(string.format(
					'%s:%s/%s/%s', tsTitle.nsText, tsTitle.baseText, cfg['sand
				if tsSandboxTitle and tsSandboxTitle.exists then
					sandboxLink = format('sandboxlink', link, ':' .. tsSandbox
				end
			end
			tStylesLinks[i] = sandboxLink or link
		end
		local tStylesList = mList.makeList('bulleted', tStylesLinks)
		boxArgs.text = format(
			mw.title.getCurrentTitle():inNamespaces(828,829) and 'header-module' or 'h
			'\n' .. tStylesList
	end
	boxArgs.type = 'notice'
	boxArgs.small = true
	boxArgs.image = string.format('[[File:Farm-Fresh css add.svg|32px|alt=%s]]', format('logo-
	return mMessageBox.main('mbox', boxArgs)
end

function p.renderTrackingCategories(args, tStyles, titleObj)
	if yesno(args.nocat) then
		return ''
	end
```

```
	local cfg = getConfig()

	local cats = {}

	-- Error category
	if #tStyles < 1 and cfg['error_category'] then
		cats[#cats + 1] = cfg['error_category']
	end

	-- TemplateStyles category
	titleObj = titleObj or mw.title.getCurrentTitle()
	if (titleObj.namespace == 10 or titleObj.namespace == 828)
		and not cfg['subpage_blacklist'][titleObj.subpageText]
	then
		local category = args.category or cfg['default_category']
		if category then
			cats[#cats + 1] = category
		end
		if not yesno(args.noprotcat) and (cfg['protection_conflict_category'] or cfg['padl
			local currentProt = titleObj.protectionLevels["edit"] and titleObj.protect
			local addedLevelCat = false
			local addedPadlockCat = false
			for i, ts in ipairs(tStyles) do
				local tsTitleObj = mw.title.new(ts)
				local tsProt = tsTitleObj.protectionLevels["edit"] and tsTitleObj.
				if cfg['protection_conflict_category'] and tsProt ~= currentProt a
					addedLevelCat = true
					cats[#cats + 1] = cfg['protection_conflict_category']
				end
				if cfg['padlock_pattern'] and tsProt and not addedPadlockCat then
					local content = tsTitleObj:getContent()
					if not content:find(cfg['padlock_pattern']) then
						cats[#cats + 1] = cfg['missing_padlock_category']
						addedPadlockCat = true
					end
				end
			end
		end
	end
	for i, cat in ipairs(cats) do
		cats[i] = string.format('[[Category:%s]]', cat)
	end
	return table.concat(cats)
end

return p
```