☐ Toggle menu
Blue Gold Program Wiki

Navigation

- Main page
- Recent changes
- Random page
- Help about MediaWiki

Tools

- What links here
- Related changes
- Special pages
- Permanent link
- Page information

Personal tools

- Log in

personal-extra

☐ Toggle search
Search

Random page

Views

- View
- View source
- History
- PDF Export

Actions

Module:Sidebar

From Blue Gold Program Wiki
The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

40px **This Lua module is used on 266,000+ pages, or roughly 14927% of all pages.**
To avoid major disruption and server load, any changes should be tested in the module's /sandbox or /testcases subpages, or in your own module sandbox. The tested changes can be added to this page in a single edit. Consider discussing changes on the talk page before implementing them.

This module is subject to page protection. It is a highly visible module in use by a very large number of pages, or is substituted very frequently. Because vandalism or mistakes would affect many pages, and even trivial editing might cause substantial load on the servers, it is protected from editing.

This module implements the templates {{sidebar}} and {{sidebar with collapsible lists}}. See the individual template pages for documentation.

```
--
-- This module implements {{Sidebar}}
--
require('Module:No globals')
local cfg = mw.loadData('Module:Sidebar/configuration')

local p = {}

local getArgs = require('Module:Arguments').getArgs

--[[
Categorizes calling templates and modules with a 'style' parameter of any
sort
for tracking to convert to TemplateStyles.

TODO after a long cleanup: Catch sidebars in other namespaces than Template
and Module.
TODO would probably want to remove /log and /archive as CS1 does
]]
local function categorizeTemplatesWithInlineStyles(args)
        local title = mw.title.getCurrentTitle()
        if title.namespace ~= 10 and title.namespace ~= 828 then return ''
end
        for _, pattern in ipairs
(cfg.i18n.pattern.uncategorized_conversion_titles) do
                if title.text:match(pattern) then return '' end
        end
        for key, _ in pairs(args) do
                if mw.ustring.find(key, cfg.i18n.pattern.style_conversion) or
key == 'width' then
                        return cfg.i18n.category.conversion
                end
        end
end

--[[
For compatibility with the original {{sidebar with collapsible lists}}
implementation, which passed some parameters through {{#if}} to trim their
whitespace. This also triggered the automatic newline behavior.
]]
-- See ([[meta:Help:Newlines and spaces#Automatic newline]])
```

```
local function trimAndAddAutomaticNewline(s)
	s = mw.ustring.gsub(s, "^%s*(.-)%s*$", "%1")
	if mw.ustring.find(s, '^[#*:;]') or mw.ustring.find(s, '^{|') then
		return '\n' .. s
	else
		return s
	end
end

--[[
Finds whether a sidebar has a subgroup sidebar.
]]
local function hasSubgroup(s)
	if mw.ustring.find(s, cfg.i18n.pattern.subgroup) then
		return true
	else
		return false
	end
end

--[[
Main sidebar function. Takes the frame, args, and an optional
collapsibleClass.
The collapsibleClass is and should be used only for sidebars with collapsible
lists, as in p.collapsible.
]]
function p.sidebar(frame, args, collapsibleClass)
	if not args then
		args = getArgs(frame)
	end
	local root = mw.html.create()
	local child = args.child and mw.text.trim(args.child) ==
cfg.i18n.child_yes

	root = root:tag('table')
	if not child then
		root
			:addClass(cfg.i18n.class.sidebar)
			-- force collapsibleclass to be sidebar-collapse
otherwise output nothing
			:addClass(collapsibleClass == cfg.i18n.class.collapse
and cfg.i18n.class.collapse or nil)
			:addClass('nomobile')
			:addClass(args.float == cfg.i18n.float_none and
cfg.i18n.class.float_none or nil)
			:addClass(args.float == cfg.i18n.float_left and
cfg.i18n.class.float_left or nil)
			:addClass(args.wraplinks ~= cfg.i18n.wrap_true and
cfg.i18n.class.wraplinks or nil)
			:addClass(args.bodyclass or args.class)
			:css('width', args.width or nil)
```

```
                        :cssText(args.bodystyle or args.style)

                if args.outertitle then
                        root
                                :tag('caption')
                                        :addClass(cfg.i18n.class.outer_title)
                                        :addClass(args.outertitleclass)
                                        :cssText(args.outertitlestyle)
                                        :wikitext(args.outertitle)
                end

                if args.topimage then
                        local imageCell = root:tag('tr'):tag('td')

                        imageCell
                                :addClass(cfg.i18n.class.top_image)
                                :addClass(args.topimageclass)
                                :cssText(args.topimagestyle)
                                :wikitext(args.topimage)

                        if args.topcaption then
                                imageCell
                                        :tag('div')
:addClass(cfg.i18n.class.top_caption)
:cssText(args.topcaptionstyle)
                                                :wikitext(args.topcaption)
                        end
                end

                if args.pretitle then
                        root
                                :tag('tr')
                                        :tag('td')
                                                :addClass(args.topimage and
cfg.i18n.class.pretitle_with_top_image
                                                        or
cfg.i18n.class.pretitle)
                                                :addClass(args.pretitleclass)
                                                :cssText(args.basestyle)
                                                :cssText(args.pretitlestyle)
                                                :wikitext(args.pretitle)
                end
        else
                root
                        :addClass(cfg.i18n.class.subgroup)
                        :addClass(args.bodyclass or args.class)
                        :cssText(args.bodystyle or args.style)
        end

        if args.title then
                if child then
```

```
                        root
                                :wikitext(args.title)
                elseroot
                                :tag('tr')
                                        :tag('th')
                                                :addClass(args.pretitle and
cfg.i18n.class.title_with_pretitle
                                                        or
cfg.i18n.class.title)
                                                :addClass(args.titleclass)
                                                :cssText(args.basestyle)
                                                :cssText(args.titlestyle)
                                                :wikitext(args.title)
                end
        end

        if args.image then
                local imageCell = root:tag('tr'):tag('td')

                imageCell
                        :addClass(cfg.i18n.class.image)
                        :addClass(args.imageclass)
                        :cssText(args.imagestyle)
                        :wikitext(args.image)

                if args.caption then
                        imageCell
                                :tag('div')
                                        :addClass(cfg.i18n.class.caption)
                                        :cssText(args.captionstyle)
                                        :wikitext(args.caption)
                end
        end

        if args.above then
                root
                        :tag('tr')
                                :tag('td')
                                        :addClass(cfg.i18n.class.above)
                                        :addClass(args.aboveclass)
                                        :cssText(args.abovestyle)
                                        :newline() -- newline required for
bullet-points to work
                                        :wikitext(args.above)
        end

        local rowNums = {}
        for k, v in pairs(args) do
                k = '' .. k
                local num = k:match('^heading(%d+)$') or
```

```
k:match('^content(%d+)$')
                if num then table.insert(rowNums, tonumber(num)) end
        end
        table.sort(rowNums)
        -- remove duplicates from the list (e.g. 3 will be duplicated if both
heading3
        -- and content3 are specified)
        for i = #rowNums, 1, -1 do
                if rowNums[i] == rowNums[i - 1] then
                        table.remove(rowNums, i)
                end
        end

        for i, num in ipairs(rowNums) do
                local heading = args['heading' .. num]
                if heading then
                        root
                                :tag('tr')
                                        :tag('th')
:addClass(cfg.i18n.class.heading)
                                                :addClass(args.headingclass)
                                                :addClass(args['heading' ..
num .. 'class'])
                                                :cssText(args.basestyle)
                                                :cssText(args.headingstyle)
                                                :cssText(args['heading' ..
num .. 'style'])
                                                :newline()
                                                :wikitext(heading)
                end

                local content = args['content' .. num]
                if content then
                        root
                                :tag('tr')
                                        :tag('td')
:addClass(hasSubgroup(content) and cfg.i18n.class.content_with_subgroup
                                                        or
cfg.i18n.class.content)
                                                :addClass(args.contentclass)
                                                :addClass(args['content' ..
num .. 'class'])
                                                :cssText(args.contentstyle)
                                                :cssText(args['content' ..
num .. 'style'])
                                                :newline()
                                                :wikitext(content)
                                                :done()
                                        -- Without a linebreak after the
</td>, a nested list like
                                        -- "* {{hlist| ...}}" doesn't parse
```

```
correctly.
                                        :newline()
                end
        end

        if args.below then
                root
                        :tag('tr')
                                :tag('td')
                                        :addClass(cfg.i18n.class.below)
                                        :addClass(args.belowclass)
                                        :cssText(args.belowstyle)
                                        :newline()
                                        :wikitext(args.below)
        end

        if not child then
                if args.navbar ~= cfg.i18n.navbar_none and args.navbar ~=
cfg.i18n.navbar_off and
                        (args.name or
frame:getParent():getTitle():gsub(cfg.i18n.pattern.sandbox, '') ~=
                        cfg.i18n.title_not_to_add_navbar) then
                        root
                                :tag('tr')
                                        :tag('td')
:addClass(cfg.i18n.class.navbar)
                                                :cssText(args.navbarstyle)
:wikitext(require('Module:Navbar')._navbar{
                                                        args.name,
                                                        mini = 1,
                                                        fontstyle =
args.navbarfontstyle
                                                })
                end
        end
        local base_templatestyles = frame:extensionTag{
                name = 'templatestyles', args = { src =
cfg.i18n.templatestyles }
        }
        local templatestyles = ''
        if args['templatestyles'] and args['templatestyles'] ~= '' then
                templatestyles = frame:extensionTag{
                        name = 'templatestyles', args = { src =
args['templatestyles'] }
                }
        end
        local child_templatestyles = ''
        if args['child templatestyles'] and args['child templatestyles'] ~=
'' then
                child_templatestyles = frame:extensionTag{
                        name = 'templatestyles', args = { src = args['child
```

```
templatestyles'] }
                }
        end
        local grandchild_templatestyles = ''
        if args['grandchild templatestyles'] and args['grandchild
templatestyles'] ~= '' then
                grandchild_templatestyles = frame:extensionTag{
                        name = 'templatestyles', args = { src =
args['grandchild templatestyles'] }
                }
        end

        return table.concat({
                base_templatestyles,
                templatestyles,
                child_templatestyles,
                grandchild_templatestyles,
                tostring(root),
                (child and cfg.i18n.category.child or ''),
                categorizeTemplatesWithInlineStyles(args)
        })
end

local function list_title(args, is_centered_list_titles, num)
        local title_text = trimAndAddAutomaticNewline(args['list' .. num ..
'title']
                or cfg.i18n.default_list_title)

        local title
        if is_centered_list_titles then
                -- collapsible can be finicky, so provide some CSS/HTML to
support
                title = mw.html.create('div')
                        :addClass(cfg.i18n.class.list_title_centered)
                        :wikitext(title_text)
        else
                title = mw.html.create()
                        :wikitext(title_text)
        end
        local title_container = mw.html.create('div')
                :addClass(cfg.i18n.class.list_title)
                -- don't /need/ a listnumtitleclass because you can do
                -- .templateclass .listnumclass .sidebar-list-title
                :addClass(args.listtitleclass)
                :cssText(args.basestyle)
                :cssText(args.listtitlestyle)
                :cssText(args['list' .. num .. 'titlestyle'])
                :node(title)
                :done()
        return title_container
end
```

```
--[[
Main entry point for sidebar with collapsible lists.
Does the work of creating the collapsible lists themselves and including them
into the args.
]]
function p.collapsible(frame)
	local args = getArgs(frame)
	if not args.name and
frame:getParent():getTitle():gsub(cfg.i18n.pattern.collapse_sandbox, '') ==
		cfg.i18n.collapse_title_not_to_add_navbar then
		args.navbar = cfg.i18n.navbar_none
	end

	local contentArgs = {}
	local is_centered_list_titles
	if args['centered list titles'] and args['centered list titles'] ~=
'' then
		is_centered_list_titles = true
	else
		is_centered_list_titles = false
	end

	for k, v in pairs(args) do
		local num = string.match(k, '^list(%d+)$')
		if num then
			local expand = args.expanded and
				(args.expanded == 'all' or args.expanded ==
args['list' .. num .. 'name'])
			local row = mw.html.create('div')
			row
				:addClass(cfg.i18n.class.list)
				:addClass('mw-collapsible')
				:addClass((not expand) and 'mw-collapsed' or
nil)
				:addClass(args['list' .. num .. 'class'])
				:cssText(args.listframestyle)
				:cssText(args['list' .. num .. 'framestyle'])
				:node(list_title(args,
is_centered_list_titles, num))
				:tag('div')
:addClass(cfg.i18n.class.list_content)
					:addClass('mw-collapsible-content')
					-- don't /need/ a listnumstyleclass
because you can do
					-- .templatename .listnumclass
.sidebar-list
					:addClass(args.listclass)
					:cssText(args.liststyle)
					:cssText(args['list' .. num ..
'style'])
:wikitext(trimAndAddAutomaticNewline(args['list' .. num]))
```

```
                        contentArgs['content' .. num] = tostring(row)
                end
        end

        for k, v in pairs(contentArgs) do
                args[k] = v
        end

        return p.sidebar(frame, args, cfg.i18n.class.collapse)
end

return p
```


Namespaces

- Module
- Discussion

Variants

Categories:

- Pages with script errors
- Pages with broken file links
- Modules subject to page protection

Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.