Navigation

Tools

Personal tools

personal-extra

Random page

Views

Actions

Module:Navbox

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 approximately 3,930,000 pages, or roughly 220539% 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.

40x40px 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.

30px This module depends on the following other modules:

- Module:Navbar
- Module:Color contrast
- Module:Arguments

Related pages

- Template (talk)
- Template sandbox
- Template testcases
- Template doc
- Template subpages
- Module (talk)
- Module sandbox
- Module testcases
- Module doc
- Module subpages

This module implements the {{Navbox}} template. Please see the template page for usage instructions.

Tracking/maintenance categories[edit source]

- Category:Navigational boxes without horizontal lists (2)
- Category:Navboxes using background colours (1)
- Category:Potentially illegible navboxes (0)
- Category:Navboxes using borders (0)

```
--
-- This module implements {{Navbox}}
--

local p = {}

local navbar = require('Module:Navbar')._navbar
local getArgs -- lazily initialized

local args
local border
local listnums
local ODD_EVEN_MARKER = '\127_ODDEVEN_\127'
local RESTART_MARKER = '\127_ODDEVEN0_\127'
local REGEX_MARKER = '\127_ODDEVEN(%d?)_\127'

local function striped(wikitext)
        -- Return wikitext with markers replaced for odd/even striping.
        -- Child (subgroup) navboxes are flagged with a category that is
```

```
removed
        -- by parent navboxes. The result is that the category shows all
pages
        -- where a child navbox is not contained in a parent navbox.
        local orphanCat = '[[Category:Navbox orphans]]'
        if border == 'subgroup' and args.orphan ~= 'yes' then
                -- No change; striping occurs in outermost navbox.
                return wikitext .. orphanCat
        end
        local first, second = 'odd', 'even'
        if args.evenodd then
                if args.evenodd == 'swap' then
                        first, second = second, first
                else
                        first = args.evenodd
                        second = first
                end
        end
        local changer
        if first == second then
                changer = first
        else
                local index = 0
                changer = function (code)
                        if code == '0' then
                                -- Current occurrence is for a group before a
nested table.
                                -- Set it to first as a valid although
pointless class.
                                -- The next occurrence will be the first row
after a title
                                -- in a subgroup and will also be first.
                                index = 0
                                return first
                        end
                        index = index + 1
                        return index % 2 == 1 and first or second
                end
        end
        local regex = orphanCat:gsub('([%[%]])', '%%%1')
        return (wikitext:gsub(regex, ''):gsub(REGEX_MARKER, changer))  -- ()
omits gsub count
end

local function processItem(item, nowrapitems)
        if item:sub(1, 2) == '{|' then
                -- Applying nowrap to lines in a table does not make sense.
                -- Add newlines to compensate for trim of x in |parm=x in a
template.
                return '\n' .. item ..'\n'
        end
```

```
        if nowrapitems == 'yes' then
                local lines = {}
                for line in (item .. '\n'):gmatch('([^\n]*)\n') do
                        local prefix, content =
line:match('^([*:;#]+)%s*(.*)')
                        if prefix and not content:match('^<span
class="nowrap">') then
                                line = prefix .. '<span class="nowrap">' ..
content .. '</span>'
                        end
                        table.insert(lines, line)
                end
                item = table.concat(lines, '\n')
        end
        if item:match('^[*:;#]') then
                return '\n' .. item ..'\n'
        end
        return item
end

local function renderNavBar(titleCell)

        if args.navbar ~= 'off' and args.navbar ~= 'plain' and not (not
args.name and mw.getCurrentFrame():getParent():getTitle():gsub('/sandbox$',
'') == 'Template:Navbox') then
                titleCell:wikitext(navbar{
                        args.name,
                        mini = 1,
                        fontstyle = (args.basestyle or '') .. ';' ..
(args.titlestyle or '') .. ';background:none transparent;border:none;-moz-
box-shadow:none;-webkit-box-shadow:none;box-shadow:none; padding:0;'
                })
        end

end

--
--   Title row
--
local function renderTitleRow(tbl)
        if not args.title then return end

        local titleRow = tbl:tag('tr')

        if args.titlegroup then
                titleRow
                        :tag('th')
                                :attr('scope', 'row')
                                :addClass('navbox-group')
                                :addClass(args.titlegroupclass)
                                :cssText(args.basestyle)
```

```
                                :cssText(args.groupstyle)
                                :cssText(args.titlegroupstyle)
                                :wikitext(args.titlegroup)
        end

        local titleCell = titleRow:tag('th'):attr('scope', 'col')

        if args.titlegroup then
                titleCell
                        :css('border-left', '2px solid #fdfdfd')
                        :css('width', '100%')
        end

        local titleColspan = 2
        if args.imageleft then titleColspan = titleColspan + 1 end
        if args.image then titleColspan = titleColspan + 1 end
        if args.titlegroup then titleColspan = titleColspan - 1 end

        titleCell
                :cssText(args.basestyle)
                :cssText(args.titlestyle)
                :addClass('navbox-title')
                :attr('colspan', titleColspan)

        renderNavBar(titleCell)

        titleCell
                :tag('div')
                        -- id for aria-labelledby attribute
                        :attr('id', mw.uri.anchorEncode(args.title))
                        :addClass(args.titleclass)
                        :css('font-size', '114%')
                        :css('margin', '0 4em')
                        :wikitext(processItem(args.title))
end

--
--   Above/Below rows
--

local function getAboveBelowColspan()
        local ret = 2
        if args.imageleft then ret = ret + 1 end
        if args.image then ret = ret + 1 end
        return ret
end

local function renderAboveRow(tbl)
        if not args.above then return end

        tbl:tag('tr')
```

```
                :tag('td')
                        :addClass('navbox-abovebelow')
                        :addClass(args.aboveclass)
                        :cssText(args.basestyle)
                        :cssText(args.abovestyle)
                        :attr('colspan', getAboveBelowColspan())
                        :tag('div')
                                -- id for aria-labelledby attribute, if no
title
                                :attr('id', args.title and nil or
mw.uri.anchorEncode(args.above))
                                :wikitext(processItem(args.above,
args.nowrapitems))
end

local function renderBelowRow(tbl)
        if not args.below then return end

        tbl:tag('tr')
                :tag('td')
                        :addClass('navbox-abovebelow')
                        :addClass(args.belowclass)
                        :cssText(args.basestyle)
                        :cssText(args.belowstyle)
                        :attr('colspan', getAboveBelowColspan())
                        :tag('div')
                                :wikitext(processItem(args.below,
args.nowrapitems))
end

--
--   List rows
--
local function renderListRow(tbl, index, listnum)
        local row = tbl:tag('tr')

        if index == 1 and args.imageleft then
                row
                        :tag('td')
                                :addClass('noviewer')
                                :addClass('navbox-image')
                                :addClass(args.imageclass)
                                :css('width', '1px')                --
Minimize width
                                :css('padding', '0px 2px 0px 0px')
                                :cssText(args.imageleftstyle)
                                :attr('rowspan', #listnums)
                                :tag('div')
:wikitext(processItem(args.imageleft))
        end
```

```
        if args['group' .. listnum] then
                local groupCell = row:tag('th')

                -- id for aria-labelledby attribute, if lone group with no
title or above
                if listnum == 1 and not (args.title or args.above or
args.group2) then
                        groupCell
                                :attr('id', mw.uri.anchorEncode(args.group1))
                end

                groupCell
                        :attr('scope', 'row')
                        :addClass('navbox-group')
                        :addClass(args.groupclass)
                        :cssText(args.basestyle)
                        :css('width', args.groupwidth or '1%') -- If
groupwidth not specified, minimize width

                groupCell
                        :cssText(args.groupstyle)
                        :cssText(args['group' .. listnum .. 'style'])
                        :wikitext(args['group' .. listnum])
        end

        local listCell = row:tag('td')

        if args['group' .. listnum] then
                listCell
                        :css('text-align', 'left')
                        :css('border-left-width', '2px')
                        :css('border-left-style', 'solid')
        else
                listCell:attr('colspan', 2)
        end

        if not args.groupwidth then
                listCell:css('width', '100%')
        end

        local rowstyle  -- usually nil so cssText(rowstyle) usually adds
nothing
        if index % 2 == 1 then
                rowstyle = args.oddstyle
        else
                rowstyle = args.evenstyle
        end

        local listText = args['list' .. listnum]
        local oddEven = ODD_EVEN_MARKER
        if listText:sub(1, 12) == '</div><table' then
```

```
                -- Assume list text is for a subgroup navbox so no automatic
striping for this row.
                oddEven = listText:find('<th[^>]*"navbox%-title"') and
RESTART_MARKER or 'odd'
        end
        listCell
                :css('padding', '0px')
                :cssText(args.liststyle)
                :cssText(rowstyle)
                :cssText(args['list' .. listnum .. 'style'])
                :addClass('navbox-list')
                :addClass('navbox-' .. oddEven)
                :addClass(args.listclass)
                :addClass(args['list' .. listnum .. 'class'])
                :tag('div')
                        :css('padding', (index == 1 and args.list1padding) or
args.listpadding or '0em 0.25em')
                        :wikitext(processItem(listText, args.nowrapitems))

        if index == 1 and args.image then
                row
                        :tag('td')
                                :addClass('noviewer')
                                :addClass('navbox-image')
                                :addClass(args.imageclass)
                                :css('width', '1px')                --
Minimize width
                                :css('padding', '0px 0px 0px 2px')
                                :cssText(args.imagestyle)
                                :attr('rowspan', #listnums)
                                :tag('div')
                                        :wikitext(processItem(args.image))
        end
end


--
--   Tracking categories
--

local function needsHorizontalLists()
        if border == 'subgroup' or args.tracking == 'no' then
                return false
        end
        local listClasses = {
                ['plainlist'] = true, ['hlist'] = true, ['hlist hnum'] =
true,
                ['hlist hwrap'] = true, ['hlist vcard'] = true, ['vcard
hlist'] = true,
                ['hlist vevent'] = true,
        }
```

```
        return not (listClasses[args.listclass] or
listClasses[args.bodyclass])
end

local function hasBackgroundColors()
        for _, key in ipairs({'titlestyle', 'groupstyle', 'basestyle',
'abovestyle', 'belowstyle'}) do
                if tostring(args[key]):find('background', 1, true) then
                        return true
                end
        end
end

local function hasBorders()
        for _, key in ipairs({'groupstyle', 'basestyle', 'abovestyle',
'belowstyle'}) do
                if tostring(args[key]):find('border', 1, true) then
                        return true
                end
        end
end

local function isIllegible()
        local styleratio = require('Module:Color contrast')._styleratio

        for key, style in pairs(args) do
                if tostring(key):match("style$") then
                        if styleratio{mw.text.unstripNoWiki(style)} < 4.5
then
                                return true
                        end
                end
        end
        return false
end

local function getTrackingCategories()
        local cats = {}
        if needsHorizontalLists() then table.insert(cats, 'Navigational boxes
without horizontal lists') end
        if hasBackgroundColors() then table.insert(cats, 'Navboxes using
background colours') end
        if isIllegible() then table.insert(cats, 'Potentially illegible
navboxes') end
        if hasBorders() then table.insert(cats, 'Navboxes using borders') end
        return cats
end

local function renderTrackingCategories(builder)
        local title = mw.title.getCurrentTitle()
        if title.namespace ~= 10 then return end -- not in template space
```

```
        local subpage = title.subpageText
        if subpage == 'doc' or subpage == 'sandbox' or subpage == 'testcases'
then return end

        for _, cat in ipairs(getTrackingCategories()) do
                builder:wikitext('[[Category:' .. cat .. ']]')
        end
end

--
--   Main navbox tables
--
local function renderMainTable()
        local tbl = mw.html.create('table')
                :addClass('nowraplinks')
                :addClass(args.bodyclass)

        if args.title and (args.state ~= 'plain' and args.state ~= 'off')
then
                if args.state == 'collapsed' then args.state = 'mw-collapsed'
end
                tbl
                        :addClass('mw-collapsible')
                        :addClass(args.state or 'autocollapse')
        end

        tbl:css('border-spacing', 0)
        if border == 'subgroup' or border == 'none' then
                tbl
                        :addClass('navbox-subgroup')
                        :cssText(args.bodystyle)
                        :cssText(args.style)
        else  -- regular navbox - bodystyle and style will be applied to the
wrapper table
                tbl
                        :addClass('navbox-inner')
                        :css('background', 'transparent')
                        :css('color', 'inherit')
        end
        tbl:cssText(args.innerstyle)

        renderTitleRow(tbl)
        renderAboveRow(tbl)
        for i, listnum in ipairs(listnums) do
                renderListRow(tbl, i, listnum)
        end
        renderBelowRow(tbl)

        return tbl
end
```

```
function p._navbox(navboxArgs)
	args = navboxArgs
	listnums = {}

	for k, _ in pairs(args) do
		if type(k) == 'string' then
			local listnum = k:match('^list(%d+)$')
			if listnum then table.insert(listnums,
tonumber(listnum)) end
		end
	end
	table.sort(listnums)

	border = mw.text.trim(args.border or args[1] or '')
	if border == 'child' then
		border = 'subgroup'
	end

	-- render the main body of the navbox
	local tbl = renderMainTable()

	-- render the appropriate wrapper around the navbox, depending on the
border param
	local res = mw.html.create()
	if border == 'none' then
		local nav = res:tag('div')
			:attr('role', 'navigation')
			:node(tbl)
		-- aria-labelledby title, otherwise above, otherwise lone
group
		if args.title or args.above or (args.group1 and not
args.group2) then
			nav:attr('aria-labelledby',
mw.uri.anchorEncode(args.title or args.above or args.group1))
		else
			nav:attr('aria-label', 'Navbox')
		end
	elseif border == 'subgroup' then
		-- We assume that this navbox is being rendered in a list
cell of a parent navbox, and is
		-- therefore inside a div with padding:0em 0.25em. We start
with a </div> to avoid the
		-- padding being applied, and at the end add a <div> to
balance out the parent's </div>
		res
			:wikitext('</div>')
			:node(tbl)
			:wikitext('<div>')
	else
		local nav = res:tag('div')
			:attr('role', 'navigation')
```

```
                        :addClass('navbox')
                        :addClass(args.navboxclass)
                        :cssText(args.bodystyle)
                        :cssText(args.style)
                        :css('padding', '3px')
                        :node(tbl)
                -- aria-labelledby title, otherwise above, otherwise lone
group
                if args.title or args.above or (args.group1 and not
args.group2) then
                        nav:attr('aria-labelledby',
mw.uri.anchorEncode(args.title or args.above or args.group1))
                else
                        nav:attr('aria-label', 'Navbox')
                end
        end

        if (args.nocat or 'false'):lower() == 'false' then
                renderTrackingCategories(res)
        end
        return striped(tostring(res))
end

function p.navbox(frame)
        if not getArgs then
                getArgs = require('Module:Arguments').getArgs
        end
        args = getArgs(frame, {wrappers = {'Template:Navbox'}})

        -- Read the arguments in the order they'll be output in, to make
references number in the right order.
        local _
        _ = args.title
        _ = args.above
        for i = 1, 20 do
                _ = args["group" .. tostring(i)]
                _ = args["list" .. tostring(i)]
        end
        _ = args.below

        return p._navbox(args)
end

return p
```

actions taken to prevent or repair the deterioration of water management infrastructure and to keep the physical components of a water management system in such a state that they can serve their intended function.

Namespaces

- Module
- Discussion

Variants

Categories:

- Pages with script errors
- Pages with broken file links
- Modules subject to page protection
- Sidebars with styles needing conversion
- Modules that add a tracking category

This page was last edited on 17 November 2020, at 10:35.

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.

- Privacy policy
- About Blue Gold Program Wiki
- Disclaimers

Developed and maintained by Big Blue Communications for Blue Gold Program

- Powered by MediaWiki

Blue Gold Program Wiki