# پودمان:Hatnote

.□□□□ □□□□ □□□□□□□□/Hatnote:□□□□□□ □□ □□□□□□□□ □□□□□□ □□□ □□□□□□□□

```
------------------------------------------------------------------------------
--                                Module:Hatnote
--
--
--
-- This module produces hatnote links and links to related articles. It
--
-- implements the {{hatnote}} and {{format link}} meta-templates and includes
--
-- helper functions for other Lua hatnote modules.
--
------------------------------------------------------------------------------

local libraryUtil = require('libraryUtil')
local checkType = libraryUtil.checkType
local checkTypeForNamedArg = libraryUtil.checkTypeForNamedArg
local mArguments -- lazily initialise [[Module:Arguments]]
local yesno -- lazily initialise [[Module:Yesno]]
local formatLink -- lazily initialise [[Module:Format link]] ._formatLink

local p = {}

------------------------------------------------------------------------------
-- Helper functions
------------------------------------------------------------------------------

local function getArgs(frame)
        -- Fetches the arguments from the parent frame. Whitespace is trimmed
and
        -- blanks are removed.
        mArguments = require('Module:Arguments')
        return mArguments.getArgs(frame, {parentOnly = true})
end

local function removeInitialColon(s)
        -- Removes the initial colon from a string, if present.
        return s:match('^:?(.*)')
end
```

```
function p.defaultClasses(inline)
	-- Provides the default hatnote classes as a space-separated string;
useful
	-- for hatnote-manipulation modules like [[Module:Hatnote group]].
	return
		(inline == 1 and 'hatnote-inline' or 'hatnote') .. ' ' ..
		'navigation-not-searchable'
end

function p.disambiguate(page, disambiguator)
	-- Formats a page title with a disambiguation parenthetical,
	-- i.e. "Example" → "Example (disambiguation)".
	checkType('disambiguate', 1, page, 'string')
	checkType('disambiguate', 2, disambiguator, 'string', true)
	disambiguator = disambiguator or 'disambiguation'
	return mw.ustring.format('%s (%s)', page, disambiguator)
end

function p.findNamespaceId(link, removeColon)
	-- Finds the namespace id (namespace number) of a link or a pagename.
This
	-- function will not work if the link is enclosed in double brackets.
Colons
	-- are trimmed from the start of the link by default. To skip colon
	-- trimming, set the removeColon parameter to false.
	checkType('findNamespaceId', 1, link, 'string')
	checkType('findNamespaceId', 2, removeColon, 'boolean', true)
	if removeColon ~= false then
		link = removeInitialColon(link)
	end
	local namespace = link:match('^(.-):')
	if namespace then
		local nsTable = mw.site.namespaces[namespace]
		if nsTable then
			return nsTable.id
		end
	end
	return 0
end

function p.makeWikitextError(msg, helpLink, addTrackingCategory, title)
	-- Formats an error message to be returned to wikitext. If
	-- addTrackingCategory is not false after being returned from
	-- [[Module:Yesno]], and if we are not on a talk page, a tracking
category
	-- is added.
	checkType('makeWikitextError', 1, msg, 'string')
	checkType('makeWikitextError', 2, helpLink, 'string', true)
	yesno = require('Module:Yesno')
	title = title or mw.title.getCurrentTitle()
	-- Make the help link text.
```

```
        local helpText
        if helpLink then
                helpText = ' ([[' .. helpLink .. '|help]])'
        else
                helpText = ''
        end
        -- Make the category text.
        local category
        if not title.isTalkPage -- Don't categorise talk pages
                and title.namespace ~= 2 -- Don't categorise userspace
                and yesno(addTrackingCategory) ~= false -- Allow opting out
        then
                category = 'Hatnote templates with errors'
                category = mw.ustring.format(
                        '[[%s:%s]]',
                        mw.site.namespaces[14].name,
                        category
                )
        else
                category = ''
        end
        return mw.ustring.format(
                '<strong class="error">Error: %s%s.</strong>%s',
                msg,
                helpText,
                category
        )
end

local curNs = mw.title.getCurrentTitle().namespace
p.missingTargetCat =
        --Default missing target category, exported for use in related
modules
        ((curNs ==  0) or (curNs == 14)) and
        'Articles with hatnote templates targeting a nonexistent page' or nil

function p.quote(title)
        --Wraps titles in quotation marks. If the title starts/ends with a
quotation
        --mark, kerns that side as with {{-'}}
        local quotationMarks = {
                ["'"]=true, ['"']=true, ['“']=true, ["‘"]=true, ['”']=true,
["’"]=true
        }
        local quoteLeft, quoteRight = -- Test if start/end are quotation
marks
                quotationMarks[string.sub(title,  1,  1)],
                quotationMarks[string.sub(title, -1, -1)]
        if quoteLeft or quoteRight then
                title = mw.html.create("span"):wikitext(title)
        end
```

```
        if quoteLeft  then title:css("padding-left",  "0.15em") end
        if quoteRight then title:css("padding-right", "0.15em") end
        return '"' .. tostring(title) .. '"'
end

-------------------------------------------------------------------------------
---
-- Hatnote
--
-- Produces standard hatnote text. Implements the {{hatnote}} template.
-------------------------------------------------------------------------------
---
p[''] = function (frame) return p.hatnote(frame:newChild{ title =
"Template:Hatnote" }) end

function p.hatnote(frame)
        local args = getArgs(frame)
        local s = args[1]
        if not s then
                return p.makeWikitextError(
                        'no text specified',
                        'Template:Hatnote#Errors',
                        args.category
                )
        end
        return p._hatnote(s, {
                extraclasses = args.extraclasses,
                selfref = args.selfref
        })
end

function p._hatnote(s, options)
        checkType('_hatnote', 1, s, 'string')
        checkType('_hatnote', 2, options, 'table', true)
        options = options or {}
        local inline = options.inline
        local hatnote = mw.html.create(inline == 1 and 'span' or 'div')
        local extraclasses
        if type(options.extraclasses) == 'string' then
                extraclasses = options.extraclasses
        end

        hatnote
                :attr('role', 'note')
                :addClass(p.defaultClasses(inline))
                :addClass(extraclasses)
                :addClass(options.selfref and 'selfref' or nil)
                :wikitext(s)

        return mw.getCurrentFrame():extensionTag{
                name = 'templatestyles', args = { src =
```

```
'Module:Hatnote/styles.css' }
        } .. tostring(hatnote)
end

return p
```