Inhaltsverzeichnis

Modul:Hatnote

[Vorlage:Lua](#)

This is a meta-module that provides various functions for making [hatnotes](#). It implements the [Vorlage:Tl](#) template, for use in hatnotes at the top of pages, and the [Vorlage:Tl](#) template, which is used to format a wikilink for use in hatnotes. It also contains a number of helper functions for use in other Lua hatnote modules.

Use from wikitext

The functions in this module cannot be used directly from #invoke, and must be used through templates instead. Please see [Template:Hatnote](#) and [Template:Format link](#) for documentation.

Use from other Lua modules

To load this module from another Lua module, use the following code.

```
local mHatnote = require('Module:Hatnote')
```

You can then use the functions as documented below.

Hatnote

```
mHatnote._hatnote(s, options)
```

Formats the string *s* as a hatnote. This encloses *s* in the tags [Vorlage:Tag](#). Options are provided in the *options* table. Options include:

- *options.extraclasses* - a string of extra classes to provide
- *options.selfref* - if this is not nil or false, adds the class "selfref", used to denote self-references to Wikipedia (see [Template:Selfref](#)))

The CSS of the hatnote class is defined in [Module:Hatnote/styles.css](#).

Example 1

```
mHatnote._hatnote('This is a hatnote.')
```

Produces: Vorlage:Tag

Displays as: Vorlage:Hatnote

Example 2

```
mHatnote._hatnote('This is a hatnote.', {extraclasses = 'boilerplate seealso', se
```

Produces: Vorlage:Tag

Displayed as: Vorlage:Hatnote

Find namespace id

```
mHatnote.findNamespaceId(link, removeColon)
```

Finds the namespace id of the string *link*, which should be a valid page name, with or without the section name. This function will not work if the page name is enclosed with square brackets. When trying to parse the namespace name, colons are removed from the start of the link by default. This is helpful if users have specified colons when they are not strictly necessary. If you do not need to check for initial colons, set *removeColon* to false.

Examples

    Vorlage:Code → 0
    Vorlage:Code → 14
    Vorlage:Code → 14
    Vorlage:Code → 0 (the namespace is detected as ":Category", rather than "Category")

Make wikitext error

```
mHatnote.makeWikitextError(msg, helpLink, addTrackingCategory)
```

Formats the string *msg* as a red wikitext error message, with optional link to a help page *helpLink*. Normally this function also adds Vorlage:Clc. To suppress categorization, pass `false` as third parameter of the function (`addTrackingCategory`).

Examples:

    Vorlage:Code → **Error: an error has occurred.**
    Vorlage:Code → **Error: an error has occurred (help).**

Examples

For an example of how this module is used in other Lua modules, see Module:Main

```
--------------------------------------------------------------------------------
--                            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 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

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.findNamespaceId(link, removeColon)
        -- Finds the namespace id (namespace number) of a link or a pagename. Thi
        -- function will not work if the link is enclosed in double brackets. Col
        -- 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 categor
        -- is added.
```

```lua
            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

    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

    --------------------------------------------------------------------------------
    -- Hatnote
    --
    -- Produces standard hatnote text. Implements the {{hatnote}} template.
    --------------------------------------------------------------------------------

    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
```

```lua
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(inline == 1 and 'hatnote-inline' or 'hatnote')
                :addClass('navigation-not-searchable')
                :addClass(extraclasses)
                :addClass(options.selfref and 'selfref')
                :wikitext(s)

        return mw.getCurrentFrame():extensionTag{
                name = 'templatestyles', args = { src = 'Module:Hatnote/styles.cs
        } .. tostring(hatnote)
end

return p
```

Modul:Hatnote/styles.css

```css
/* {{pp|small=y}} */
.hatnote {
        font-style: italic;
}

/* Limit structure CSS to divs because of [[Module:Hatnote inline]] */
div.hatnote {
        /* @noflip */
        padding-left: 1.6em;
        margin-bottom: 0.5em;
}

.hatnote i {
        font-style: normal;
}

/* The templatestyles element inserts a link element before hatnotes.
 * TODO: Remove link if/when WMF resolves T200206 */
.hatnote + link + .hatnote {
        margin-top: -0.5em;
}
```