https://fa.wikimarefat.ir/view/%D9%BE%D9%88%D8%AF%D9%85%D8%A7%D9%86:Format_link

پودمان:Format link

.□□□□ □□□□ □□□□□□□□/Format link:□□□□□□ □□ □□□□□□□□ □□□□□□ □□□ □□□□□□□□

```
------------------------------------------------------------------------------
---
-- Format link
--
-- Makes a wikilink from the given link and display values. Links are escaped
-- with colons if necessary, and links to sections are detected and displayed
-- with " § " as a separator rather than the standard MediaWiki "#". Used in
-- the {{format link}} template.
------------------------------------------------------------------------------
---
local libraryUtil = require('libraryUtil')
local checkType = libraryUtil.checkType
local checkTypeForNamedArg = libraryUtil.checkTypeForNamedArg
local mArguments -- lazily initialise [[Module:Arguments]]
local mError -- lazily initialise [[Module:Error]]
local yesno -- lazily initialise [[Module:Yesno]]

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

local function maybeItalicize(s, shouldItalicize)
	-- Italicize s if s is a string and the shouldItalicize parameter is
true.
	if s and shouldItalicize then
```

```
                return '<i>' .. s .. '</i>'
        else
                return s
        end
end

local function parseLink(link)
        -- Parse a link and return a table with the link's components.
        -- These components are:
        -- - link: the link, stripped of any initial colon (always present)
        -- - page: the page name (always present)
        -- - section: the page name (may be nil)
        -- - display: the display text, if manually entered after a pipe (may
be nil)
        link = removeInitialColon(link)

        -- Find whether a faux display value has been added with the {{!}}
magic
        -- word.
        local prePipe, display = link:match('^(.-)|(.*)$')
        link = prePipe or link

        -- Find the page, if it exists.
        -- For links like [[#Bar]], the page will be nil.
        local preHash, postHash = link:match('^(.-)#(.*)$')
        local page
        if not preHash then
                -- We have a link like [[Foo]].
                page = link
        elseif preHash ~= '' then
                -- We have a link like [[Foo#Bar]].
                page = preHash
        end

        -- Find the section, if it exists.
        local section
        if postHash and postHash ~= '' then
                section = postHash
        end
        return {
                link = link,
                page = page,
                section = section,
                display = display,
        }
end

local function formatDisplay(parsed, options)
        -- Formats a display string based on a parsed link table (matching
the
        -- output of parseLink) and an options table (matching the input
```

```
options for
        -- _formatLink).
        local page = maybeItalicize(parsed.page, options.italicizePage)
        local section = maybeItalicize(parsed.section,
options.italicizeSection)
        if (not section) then
                return page
        elseif (not page) then
                return mw.ustring.format('§ %s', section)
        else
                return mw.ustring.format('%s § %s', page, section)
        end
end

local function missingArgError(target)
        mError = require('Module:Error')
        return mError.error{message =
                'Error: no link or target specified! ([[' .. target ..
'#Errors|help]])'
        }
end

------------------------------------------------------------------------
---
-- Main functions
------------------------------------------------------------------------
---

function p.formatLink(frame)
        -- The formatLink export function, for use in templates.
        yesno = require('Module:Yesno')
        local args = getArgs(frame)
        local link = args[1] or args.link
        local target = args[3] or args.target
        if not (link or target) then
                return missingArgError('Template:Format link')
        end

        return p._formatLink{
                link = link,
                display = args[2] or args.display,
                target = target,
                italicizePage = yesno(args.italicizepage),
                italicizeSection = yesno(args.italicizesection),
                categorizeMissing = args.categorizemissing
        }
end

function p._formatLink(options)
        -- The formatLink export function, for use in modules.
        checkType('_formatLink', 1, options, 'table')
```

```
	local function check(key, expectedType) --for brevity
		checkTypeForNamedArg(
			'_formatLink', key, options[key], expectedType or
'string', true
		)
	end
	check('link')
	check('display')
	check('target')
	check('italicizePage', 'boolean')
	check('italicizeSection', 'boolean')
	check('categorizeMissing')

	-- Normalize link and target and check that at least one is present
	if options.link == '' then options.link = nil end
	if options.target == '' then options.target = nil end
	if not (options.link or options.target) then
		return missingArgError('Module:Format link')
	end

	local parsed = parseLink(options.link)
	local display = options.display or parsed.display
	local catMissing = options.categorizeMissing
	local category = ''

	-- Find the display text
	if not display then display = formatDisplay(parsed, options) end

	-- Handle the target option if present
	if options.target then
		local parsedTarget = parseLink(options.target)
		parsed.link = parsedTarget.link
		parsed.page = parsedTarget.page
	end

	-- Test if page exists if a diagnostic category is specified
	if catMissing and (mw.ustring.len(catMissing) > 0) then
		local title = nil
		if parsed.page then title = mw.title.new(parsed.page) end
		if title and (not title.isExternal) then
			local success, exists = pcall(function() return
title.exists end)
			if success and not exists then
				category =
mw.ustring.format('[[Category:%s]]', catMissing)
			end
		end
	end
	-- Format the result as a link
	if parsed.link == display then
		return mw.ustring.format('[[:%s]]%s', parsed.link, category)
```

```
        else
                return mw.ustring.format('[[:%s|%s]]%s', parsed.link,
display, category)
        end
end

--------------------------------------------------------------------------
---
-- Derived convenience functions
--------------------------------------------------------------------------
---

function p.formatPages(options, pages)
        -- Formats an array of pages using formatLink and the given options
table,
        -- and returns it as an array. Nil values are not allowed.
        local ret = {}
        for i, page in ipairs(pages) do
                ret[i] = p._formatLink{
                        link = page,
                        categorizeMissing = options.categorizeMissing,
                        italicizePage = options.italicizePage,
                        italicizeSection = options.italicizeSection
                }
        end
        return ret
end

return p
```