پودمان:High-use

.□□□□ □□□□ [□□□□□□□□/High-use:□□□□□□□](#) □□ □□□□□□□□ □□□□□□□ □□□□ □□□□□□□□

```
require('strict')

local p = {}
local getArgs = require('Module:Arguments').getArgs

local _fetch = require('Module:Transclusion_count')._fetch -- _fetch looks at
the 'demo' argument
local yesno = require('Module:Yesno')

local lang_obj = mw.getContentLanguage() -- this here because the language
object is used multiple places in the module
local large_count_cutoff = 100000
local approx_num_total_pages = 63000000

local user_subpage_info_page = 'Wikipedia:User pages#SUB'
local sandbox_module_page = 'Module:Sandbox'
local system_messages_cat = 'Pages used in system messages needing
protection'
local sandbox_word = 'sandbox'
local testcases_word = 'testcases'
local doc_word = 'doc'

local function count_from_args(args)
        if tonumber(args.count) then -- check if function has already been
used
                return tonumber(args.count) -- early exit if so
        end
        local count
        if yesno(args['fetch']) ~= false then
                count = _fetch(args) -- fetch transclusion count
        end
        -- use explicitly-provided count when fetch fails
        if count == nil and args[1] ~= nil and args[1] ~= '' then
                -- convert local language number string to a number
understandable by Lua
                count = mw.ustring.gsub(args[1], '+$', '')
                count = lang_obj:parseFormattedNumber(args[1])
        end
        -- in case someone writes a non-positive number
        if count and count > 0 then
                return count
        end
```

```
        return nil
end

-- Actions if there is a large (greater than or equal to 100,000)
transclusion count
local function risk_boolean(args)
        if args.risk == true or args.risk == false then
                return args.risk
        elseif args[1]  == 'risk' then
                return true
        else
                local count = count_from_args(args)
                if count and count >= large_count_cutoff then
                        return true
                end
        end
        return false
end

-- function retained for backwards compatibility
function p._risk(args)
        return risk_boolean(args) and 'risk' or ''
end

-- function retained for backwards compatibility
function p.risk(frame)
        return p._risk(getArgs(frame))
end

-- count and no_percent arguments retained for backwards compatibility
function p._num(args, count, no_percent)
        if count == nil then
                count = count_from_args(args)
        end
        args.count = count
        args.risk = risk_boolean(args)
        -- Build output string
        local return_value = ''
        if args.count == nil and args.risk then
                return 'a very large number of'
        elseif args.count == nil then
                return 'many'
        else
                -- Use 2 significant figures for smaller numbers and 3 for
larger ones
                local sigfig = 2
                if args.count >= large_count_cutoff then
                        sigfig = 3
                end
                -- Prepare to round to appropriate number of sigfigs
                local f = math.floor(math.log10(args.count)) - sigfig + 1
```

```
                -- Round and insert 'approximately' or '+' when appropriate
                if yesno(args[2]) == true or (type(args[1]) == 'string' and
(mw.ustring.sub(args[1], -1) == '+')) then
                        -- Round down
                        return_value = string.format('%s+',
lang_obj:formatNum(math.floor( (args.count / 10^(f)) ) * (10^(f))) )
                else
                        -- Round to nearest
                        return_value = string.format('approximately&amp;#x20;%s',
lang_obj:formatNum(math.floor( (args.count / 10^(f)) + 0.5) * (10^(f))) )
                end
                -- Insert percentage of pages if that is likely to be >= 1%
and when |no-percent= not set to yes
                no_percent = yesno(no_percent or args['no-percent'])
                if args.count and args.count >= approx_num_total_pages/100
and not no_percent then
                        local num_total_pages =
mw.getCurrentFrame():callParserFunction('NUMBEROFPAGES', 'R')
                        local total_percent = math.floor( ( (
args.count/num_total_pages ) * 100) + 0.5)
                        if total_percent >= 1 then
                                return_value = string.format('%s&amp;#x20;pages,
or roughly %s%% of all', return_value, total_percent)
                        end
                end
        end
        return return_value
end

-- used by [[Template:Stub documentation]] and other pages
-- count argument retained for backwards compatibility
function p.num(frame, count)
        return p._num(getArgs(frame), count)
end

-- count argument retained for backwards compatibility
function p._text(args, count)
        --[=[
                Only show the information about how this template gets
updated
                if someone is actually editing the page and maybe trying to
update the count.
        ]=]
        local bot_text = (mw.getCurrentFrame():preprocess('{{REVISIONID}}')
== '') and ("\n\n----\n'''Preview message''':" .. ' Transclusion count
updated automatically ([[Template:High-use/doc#Technical details|see
documentation]]).') or ''
        if count == nil then
                count = count_from_args(args)
        end
        args.count = count
```

```
        args.risk = risk_boolean(args)
        -- trim /doc, /sandbox and /testcases
        local title = args.title or (args.demo and args.demo ~= '' and
mw.title.new(args.demo, 'Template')) or mw.title.getCurrentTitle()
        if title.subpageText == doc_word or title.subpageText == sandbox_word
or title.subpageText == testcases_word then
                title = title.basePageTitle
        end
        -- use /testcases of base template
        local testcases_page = mw.title.new(title.prefixedText .. '/' ..
testcases_word)
        -- exists is expensive
        while testcases_page.basePageTitle.isSubpage and not
testcases_page.exists do
                testcases_page =
mw.title.new(testcases_page.basePageTitle.basePageTitle.prefixedText .. '/'
.. testcases_word)
        end
        local systemMessages = (args['system'] or '') ~= ''
        -- This retrieves the project URL automatically to simplify
localization.
        local templateCount = ('on
[https://linkcount.toolforge.org/?project=%s&page=%s#transclusions %s
pages]'):format(
                title:fullUrl():gsub('//(.-)/.*', '%1'),
                mw.uri.encode(title.fullText), p._num(args))
        local used_on_text = "'''This " .. (title:inNamespace('Module') and
'Lua module' or 'template') .. ' is used '
        if systemMessages then
                used_on_text = used_on_text .. args['system'] ..
                        ((args.count and args.count > 2000) and ("''', and "
.. templateCount) or ("''''"))
        else
                used_on_text = used_on_text .. templateCount .. "''''"
        end
        local sandbox_text = ('%s\'s [[%s/sandbox|/sandbox]] or
[[%s|/testcases]] subpages, or in your own [[%s]]. '):format(
                title:inNamespace('Module') and 'module' or 'template',
                title.fullText,
                testcases_page.fullText,
                title:inNamespace('Module') and (sandbox_module_page ..
'|module sandbox') or (user_subpage_info_page .. '|user subpage')
        )
        local infoArg = args['info'] ~= '' and args['info']
        if (systemMessages or args.risk) then
                local info = '.'
                if systemMessages then
                        info = info .. '<br />Changes to it can cause
immediate changes to the ' .. mw.site.namespaces.Project.name .. ' user
interface.'
                end
```

```
                if infoArg then
                        info = info .. '<br />' .. infoArg
                end
                sandbox_text = info .. '<br /> To avoid major disruption' ..
                        (args.count and args.count >= large_count_cutoff and
' and server load' or '') .. -- should this use args.risk?
                        ', any changes should be tested in the ' ..
sandbox_text ..
                        'The tested changes can be added to this page in a
single edit. '
        else
                sandbox_text = (infoArg and ('.<br />' .. infoArg .. ' C') or
' and c') ..
                        'hanges may be widely noticed. Test changes in the '
.. sandbox_text
        end
        local discussion_text = systemMessages and 'Please discuss changes '
or 'Consider discussing changes '
        if args[2] ~= nil and args[2] ~= '' and yesno(args[2]) == nil then
                discussion_text = string.format('%sat [[%s]]',
discussion_text, args[2])
        else
                discussion_text = string.format('%son the [[%s|talk page]]',
discussion_text, title.talkPageTitle.fullText)
        end
        return used_on_text .. sandbox_text .. discussion_text .. ' before
implementing them.' .. bot_text
end

-- used by [[Template:R from high-use template]]
-- count argument retained for backwards compatibility
function p.text(frame, count)
        return p._text(getArgs(frame), count)
end

-- nocat argument retained for backwards compatibility
function p._main(args, nocat)
        args.count = count_from_args(args)
        args.risk = risk_boolean(args)
        args.title = (args.demo and args.demo ~= '' and
mw.title.new(args.demo, 'Template')) or mw.title.getCurrentTitle()
        local image = 'Ambox warning yellow.svg'
        local type_param = 'style'
        local epilogue = ''
        if args['system'] and args['system'] ~= '' then
                image = 'Ambox important.svg'
                type_param = 'content'
                if yesno(nocat or args['nocat']) ~= true and not
args.title.isRedirect then
                        local protection_action =
(args.title:inNamespace('File') and 'upload') or 'edit'
```

```
                        local protection_level = require('Module:Effective
protection level')._main(protection_action, args.title.fullText)
                        if protection_level ~= 'sysop' and protection_level
~= 'templateeditor' and protection_level ~= 'interfaceadmin' then
                                epilogue =
mw.getCurrentFrame():expandTemplate {
                                        title = 'sandbox other',
                                        args = {
                                                [2] = '[[Category:' ..
system_messages_cat .. ']]'
                                        }
                                }
                        end
                end
        elseif args.risk then
                image = 'Ambox warning orange.svg'
                type_param = 'content'
        end
        image = '[[File:' .. image .. '|40px|alt=Warning|link=]]'
        if args['form'] == 'editnotice' then
                return mw.getCurrentFrame():expandTemplate{
                                title = 'editnotice',
                                args = {
                                                ['image'] = image,
                                                ['text'] = p._text(args),
                                                ['expiry'] = (args['expiry']
or '')
                                }
                } .. epilogue
        else
                return require('Module:Message box').main('ombox', {
                        type = type_param,
                        image = image,
                        text = p._text(args),
                        expiry = (args['expiry'] or '')
                }) .. epilogue
        end
end

function p.main(frame)
        return p._main(getArgs(frame))
end

return p
```