



Modul:ComplForColorModules

This module is used by **Module:BrewerColors** and **Module:ChartColors**.

To translate or review the translation of the module to your language, edit carefully [Data:I18n/ComplForColorModules.tab](#).

```
local p = {}

local TNTT = require "Module:TNTTools"

local I18n = 'ComplForColorModules'

--local MoreOneNoData = "Found 2 legends of \"Value not assigned\" or \"Data not
--local FoundNLegendsExpectedN = "Found $1 legends, expected $2"
--local Color = "Color"
--local Legend = "legend"

local function I18nStr (S, ...)
    return TNTT.GetMsgP (I18n, S, {...})
end

function p.ColorNameInvFromS0 (S)
    local IsInv = false
    local ColorName = ''
    local Params = {}
    if S ~= '' then
        for w in S:gmatch("[^_]+") do
            table.insert(Params, w)
        end
        ColorName = Params[1]
    end
    local PN = table.getn(Params)
    if (PN == 2) and (Params[2]=='i') then
        IsInv = true
    end
    return ColorName, IsInv, Params
end

function p.ColorNameInv (args)
    local ColorName, IsInv, Params = p.ColorNameInvFromS0(args[1])
    return ColorName, IsInv
end

function p.GetLabels(Args, N, Pos)
    local Labels = {}
    local index = Pos
    while Args[index] do
        Labels[#Labels+1] = Args[index]
        index = index + 1
    end
    local NLabels = #Labels
    local OutlineColor = Args['outline'] or ''
    if (NLabels ~= 0) and (NLabels ~= N) then
        local StartUnknown = (string.sub(Labels[1],1,2) == "--")
        local EndUnknown = (string.sub(Labels[NLabels],1,2) == "--")
        if StartUnknown and EndUnknown then
            error (I18nStr ('MoreOneNoData'))
        elseif StartUnknown or EndUnknown then
```



```
        N = N + 1
    end
    if (NLabels ~= 0) and (NLabels ~= N) then
        error (I18nStr ('FoundNLegendsExpectedN', tostring(NLabels)),
    end
end
return Labels, NLabels, OutlineColor
end

function SColor (Color)
    if string.sub(Color,1,1) == '#' then --the usual
        return string.sub(Color,2,100)
    else
        return Color
    end
end

function p.Box(Color,WriteColor)
    if WriteColor == '-' then
        WriteColor = Color
    elseif WriteColor == 'a' then
        WriteColor = Color..'ff' --Alpha channel
    end
    local TheBox = '<span style="background-color: '..Color..'"; border:1px so
    if WriteColor ~= '' then
        TheBox = TheBox..'&nbsp; '..WriteColor..' &nbsp;';
    end
    return TheBox
end

function p.TextWithTooltip (Text, Tooltip)
    if Tooltip ~= '' then
        return '<span title="'..Tooltip..'">'..Text..'</span>'
    else
        return Text
    end
end

function p.LegendColor(Color, Text, Tooltip)
    if Text == '' then
        Text = SColor(Color)
    end
    return '<li style="list-style-type: none; list-style-image: none;"><span
end

function LegendCode(Color, Text, OutlineColor)
    local SOutlineColor = ''
    if OutlineColor ~= '' then
        SOutlineColor = '|outline='..OutlineColor
    end
    return '{{'..TNTT.GetStrP(I18n,'Legend')..' '|'..Color..' '|'..Text..SOutline
end

function p.LegendText (AColors, Labels, NLabels, ColWidth, IsTemplate, OutlineCo
    if ColWidth then
        ColWidth = mw.text.trim(ColWidth)
        if #ColWidth == 0 then
            ColWidth = nil
        end
    end
    local Show, Codes = {}, {}
    local Text = ''
    local WithLabels = NLabels > 0
    if WithLabels then
```



```
        local Gray = '#b3b3b3' -- A 40% gray
        if string.sub(Labels[1],1,2) == '--' then
            table.insert(AColors, 1, Gray)
            Labels[1] = string.sub(Labels[1], 3, 1000)
        elseif string.sub(Labels[NLabels],1,2) == '--' then
            table.insert(AColors, Gray)
            Labels[NLabels] = string.sub(Labels[NLabels], 3, 1000)
        end
    end
end
for i=1, table.getn(AColors) do
    if WithLabels then Text = Labels[i] end
    table.insert(Show, p.LegendColor(AColors[i],Text, ''))
end
SShow = table.concat(Show, "\n")
if ColWidth then
    local frame = mw.getCurrentFrame()
    SShow = frame:expandTemplate{title="div col",args={colwidth=ColWidth}}
    SShow = SShow .. frame:expandTemplate{title="div col end"}
end
for i=1, table.getn(AColors) do
    if WithLabels then Text = Labels[i] end
    table.insert(Codes, ' ' .. LegendCode(AColors[i],Text,OutlineColor))
end
local SCodes = '<pre>\n'
if ColWidth then
    SCodes = SCodes .. require("Module:Template invocation").invocation
end
SCodes = SCodes .. table.concat(Codes, "\n")
if ColWidth and #ColWidth ~= 0 then
    SCodes = SCodes .. "\n{{div col end}}"
end
SCodes = SCodes .. "\n</pre>"
return SShow..' \n' .. SCodes
end
return p
```