Modul:ChartColors

Used by **Vorlage:TI** and **Vorlage:TI**

To translate or review the translation of the module to your language, edit carefully Data:I18n/ChartColors.tab.

Function ShowColors

Code	Result
{{#invoke:ChartColors|ShowColors|Categ20|5}}	**Lua-Fehler: bad argument #1 to "get" (not a valid title)**
{{ChartColors|Categ20|9}}	**Lua-Fehler: bad argument #1 to "get" (not a valid title)**
{{#invoke:ChartColors|ShowColors|Categ20|3|-}}	**Lua-Fehler: bad argument #1 to "get" (not a valid title)**
{{#invoke:ChartColors|ShowColors|Categ20|3|a}}	**Lua-Fehler: bad argument #1 to "get" (not a valid title)**
{{#invoke:ChartColors|ShowColors|Plotter|5}}	**Lua-Fehler: bad argument #1 to "get" (not a valid title)**
{{ChartColors|Plotter|20}}	**Lua-Fehler: bad argument #1 to "get" (not a valid title)**

Function Legends

```
{{#invoke:ChartColors|Legends|Categ20|5|2}}
```

produces:

Lua-Fehler: expandTemplate: template "div col" does not exist

```
{{#invoke:ChartColors|Legends|Categ20|3|1|One|Two|Three}}
```

produces:

Lua-Fehler: expandTemplate: template "div col" does not exist

```
local p = {}

local ARG = require "Module:Arguments"
local CFCM = require "Module:ComplForColorModules"
```

```
local TNTT = require "Module:TNTTools"
--local SD = require "Module:SimpleDebug"

local I18n = 'ChartColors'

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

--local RS_ColorNameInvalid = 'El nom del color (%s) no és vàlid (ha de ser Categ
--local RS_ColorNumInvalid = 'El nombre de colors hauria d'estar entre 1 i %d (ar
--local RS_StartNumInvalid = 'El número del primer color hauria d'estar entre 1 i

local category20 = {
        '#1f77b4', '#aec7e8', '#ff7f0e', '#ffbb78', '#2ca02c', '#98df8a', '#d6272
        '#8c564b', '#c49c94', '#e377c2', '#f7b6d2', '#7f7f7f', '#c7c7c7', '#bcbd2
}
local plotter = {--from Module:Plotter/DefaultColors
        'red', 'blue', 'green', 'yellow', 'fuchsia', 'aqua', 'brown', 'orange',
        '#F0A3FF', '#191919', '#005C31', 'honeydew', '#808080', 'khaki', 'lime',
        '#5EF1F2', 'turquoise', '#E0FF66', 'violet', '#FFFF80', '#FF5005',
}

function p.GetColors (ColorName, IsInv, StartN, N, CallError)
        local SelColors = {}
        local ColorNameL = string.lower(ColorName)
        local Palet0 = {}
        if ColorNameL == 'categ20' then
                Palet0 = category20
        elseif ColorNameL == 'plotter' then
                Palet0 = plotter
        elseif CallError then
                error (I18nStr ('ColorNameInvalid',ColorName))
        end
        MaxN = table.getn(Palet0)
        if (N < 0) or (N > MaxN) then
                error (I18nStr ('ColorNumInvalid',tostring(MaxN),tostring(N)))
        else
                Palet = {}
                if IsInv then
                        for i=MaxN, 1, -1 do
                                table.insert(Palet, Palet0[i])
                        end
                else
                        Palet = Palet0
                end
                if StartN > MaxN then
                        error (I18nStr ('StartNumInvalid',tostring(MaxN),tostring
                end
                local Step = 1
                local NEnd = N
                if (ColorNameL == 'categ20') and (N<=10) then
                        Step = 2
                        if StartN == 1 then
                                NEnd = N*2
                        end
                end
                if StartN == 1 then
                        for i=1, NEnd, Step do
                                table.insert(SelColors, Palet[i])
                        end
                else
                        Count = 0
                        for i=StartN, MaxN, Step do
```

```
                                table.insert(SelColors, Palet[i])
                                Count = Count + 1
                                if Count == NEnd then
                                        break
                                end
                        end
                        if Count < NEnd then
                                for i=1, MaxN, Step do
                                        table.insert(SelColors, Palet[i])
                                        Count = Count + 1
                                        if Count == NEnd then
                                                break
                                        end
                                end
                        end
                end
                return SelColors
        end
end --GetColors

function p.ColorNameInvStartFromS (S)
        local StartN = 1
        local ParamsA = {}
        local ColorName, IsInv
    ColorName, IsInv, ParamsA = CFCM.ColorNameInvFromS0 (S)
        local PN = table.getn(ParamsA)
        if PN > 1 then
                if ParamsA[2]=='i' then
                        IsInv = true
                else
                        StartN = tonumber (ParamsA[2])
                end
                if PN == 3 then
                        StartN = tonumber (ParamsA[3])
                end
        end
        return ColorName, IsInv, StartN
end --ColorNameInvStartFromS

function ColorNameInv (args)
        local S = args[1] or ''
    local ColorName, IsInv, ParamsA = CFCM.ColorNameInvFromS0 (S)
    local StartN = 1
        ColorName, IsInv, StartN = p.ColorNameInvStartFromS (S)
        return ColorName, IsInv, StartN
end --ColorNameInv

function p.ShowColors(frame)
        local args = ARG.getArgs(frame,{
                removeBlanks = false
        });
        local ColorName, IsInv, StartN = ColorNameInv (args)
        local N = tonumber(args[2])
        local WriteColor = args[3] or ""
        local ColorFound = {}
        ColorFound = p.GetColors (ColorName, IsInv, StartN, N, true)
        local boxes = {}
        for i=1, table.getn(ColorFound) do
                table.insert(boxes, CFCM.Box(ColorFound[i],WriteColor))
        end
        return table.concat(boxes, " ")
end --ShowColors

function p.Legends(frame)
```

```
        local args = ARG.getArgs(frame,{
                removeBlanks = false
        })
        local Nargs = require("Module:TableTools").length(args)
        local ColorName, IsInv, StartN = ColorNameInv (args)
        local N = tonumber(args[2])
        local ColWidth = args[3]
        local ColorFound = {}
        local Labels = {}
        local NLabels = 0
        local IsTemplate = true
        ColorFound = p.GetColors (ColorName, IsInv, StartN, N, true)
        Labels, NLabels, OutlineColor = CFCM.GetLabels (args, N, 4)
        return CFCM.LegendText (ColorFound, Labels, NLabels, ColWidth, IsTemplate
end

return p
```