



Inhaltsverzeichnis

1. Modul:Color contrast/colors/Doku	2
2. Modul:Color contrast	3
3. Modul:Color contrast/colors	8



Modul:Color contrast/colors/Doku

Dies ist die Dokumentationsseite für Modul:Color contrast/colors

This module contains the **relative luminance** of **HTML colors**. It is used in **Module:Color contrast**.

Modul:Color contrast

Vorlage:Lua

This module is used primarily by

- [Vorlage:TI](#)
- [Vorlage:TI / Vorlage:TI](#)
- [Vorlage:TI](#)
- [Vorlage:TI](#)
- [Vorlage:TI](#)
- [Vorlage:TI](#)

It is also used for tracking within

- [Module:Navbar](#)
- [Module:Userbox](#)
- [Module:Episode list](#)

and for documentation in

- [Module:College color](#)

The formulas used are [stated in WCAG 2.x specifications](#). **WCAG** is the main guideline for creating accessible interfaces on the web.

Usage

To use this module, you may use one of the above listed templates or invoke the module directly

- To compute relative luminescence:
`{{ColorToLum|color}}` or `{{#invoke:Color contrast|lum|color}}`
 - To compute a contrast ratio between two colors:
`{{Color contrast ratio|color1|color2|error=?}}` or `{{#invoke:Color contrast|ratio|color1|color2|error=?}}`
 - To determine which of two colors (color2a and color2b) has the greater contrast ratio with a particular color (color1):
`{{Greater color contrast ratio|color1|color2a|color2b}}` or `{{#invoke:Color contrast|greatercontrast|color1|color2a|color2b}}`
 - To compute the contrast ratio between the background and text colors specified in a css style string:
`{{#invoke:Color contrast|styleratio|css style statement string|default background color|default text color}}`
-



```
--
-- This module implements
-- {{Color contrast ratio}}
-- {{Greater color contrast ratio}}
-- {{ColorToLum}}
-- {{RGBColorToLum}}
--
local p = {}
local HTMLcolor = mw.loadData( 'Module:Color contrast/colors' )

local function sRGB (v)
    if (v <= 0.03928) then
        v = v / 12.92
    else
        v = math.pow((v+0.055)/1.055, 2.4)
    end
    return v
end

local function rgbdec2lum(R, G, B)
    if ( 0 <= R and R < 256 and 0 <= G and G < 256 and 0 <= B and B < 256 ) then
        return 0.2126 * sRGB(R/255) + 0.7152 * sRGB(G/255) + 0.0722 * sRGB(B/255)
    else
        return ''
    end
end

local function hsl2lum(h, s, l)
    if ( 0 <= h and h < 360 and 0 <= s and s <= 1 and 0 <= l and l <= 1 ) then
        local c = (1 - math.abs(2*l - 1))*s
        local x = c*(1 - math.abs( math.fmod(h/60, 2) - 1) )
        local m = l - c/2

        local r, g, b = m, m, m
        if( 0 <= h and h < 60 ) then
            r = r + c
            g = g + x
        elseif( 60 <= h and h < 120 ) then
            r = r + x
            g = g + c
        elseif( 120 <= h and h < 180 ) then
            g = g + c
            b = b + x
        elseif( 180 <= h and h < 240 ) then
            g = g + x
            b = b + c
        elseif( 240 <= h and h < 300 ) then
            r = r + x
            b = b + c
        elseif( 300 <= h and h < 360 ) then
            r = r + c
            b = b + x
        end
        return rgbdec2lum(255*r, 255*g, 255*b)
    else
        return ''
    end
end

local function color2lum(c)
    if (c == nil) then
```

```
        return ''
    end

    -- html '#' entity
    c = c:gsub("&#35;", "#")

    -- whitespace
    c = c:match( '^%s*(.)[%s;]*$' )

    -- unstrip nowiki strip markers
    c = mw.text.unstripNoWiki(c)

    -- lowercase
    c = c:lower()

    -- first try to look it up
    local L = HTMLcolor[c]
    if (L ~= nil) then
        return L
    end

    -- convert from hsl
    if mw.usttring.match(c, '^hsl%([%s]*[0-9][0-9%.]*[%s]*,[%s]*[0-9][0-9%.]*%9
        local h, s, l = mw.usttring.match(c, '^hsl%([%s]*([0-9][0-9%.]*)[%s]*
        return hsl2lum(tonumber(h), tonumber(s)/100, tonumber(l)/100)
    end

    -- convert from rgb
    if mw.usttring.match(c, '^rgb%([%s]*[0-9][0-9]*[%s]*,[%s]*[0-9][0-9]*[%s]*
        local R, G, B = mw.usttring.match(c, '^rgb%([%s]*([0-9][0-9]*)[%s]*
        return rgbdec2lum(tonumber(R), tonumber(G), tonumber(B))
    end

    -- convert from rgb percent
    if mw.usttring.match(c, '^rgb%([%s]*[0-9][0-9%.]*%[%s]*,[%s]*[0-9][0-9%.]*
        local R, G, B = mw.usttring.match(c, '^rgb%([%s]*([0-9][0-9%.]*)%[%s]*
        return rgbdec2lum(255*tonumber(R)/100, 255*tonumber(G)/100, 255*
    end

    -- remove leading # (if there is one) and whitespace
    c = mw.usttring.match(c, '^[%s#]*([a-f0-9]*)[%s]*$')

    -- split into rgb
    local cs = mw.text.split(c or '', '')
    if( #cs == 6 ) then
        local R = 16*tonumber('0x' .. cs[1]) + tonumber('0x' .. cs[2])
        local G = 16*tonumber('0x' .. cs[3]) + tonumber('0x' .. cs[4])
        local B = 16*tonumber('0x' .. cs[5]) + tonumber('0x' .. cs[6])

        return rgbdec2lum(R, G, B)
    elseif ( #cs == 3 ) then
        local R = 16*tonumber('0x' .. cs[1]) + tonumber('0x' .. cs[1])
        local G = 16*tonumber('0x' .. cs[2]) + tonumber('0x' .. cs[2])
        local B = 16*tonumber('0x' .. cs[3]) + tonumber('0x' .. cs[3])

        return rgbdec2lum(R, G, B)
    end

    -- failure, return blank
    return ''
end

-- This exports the function for use in other modules.
-- The colour is passed as a string.
```

```
function p._lum(color)
    return color2lum(color)
end

function p._greatercontrast(args)
    local bias = tonumber(args['bias'] or '0') or 0
    local css = (args['css'] and args['css'] ~= '') and true or false
    local v1 = color2lum(args[1] or '')
    local c2 = args[2] or '#FFFFFF'
    local v2 = color2lum(c2)
    local c3 = args[3] or '#000000'
    local v3 = color2lum(c3)
    local ratio1 = -1;
    local ratio2 = -1;
    if (type(v1) == 'number' and type(v2) == 'number') then
        ratio1 = (v2 + 0.05)/(v1 + 0.05)
        ratio1 = (ratio1 < 1) and 1/ratio1 or ratio1
    end
    if (type(v1) == 'number' and type(v3) == 'number') then
        ratio2 = (v3 + 0.05)/(v1 + 0.05)
        ratio2 = (ratio2 < 1) and 1/ratio2 or ratio2
    end

    if css then
        local c1 = args[1] or ''
        if mw.ustring.match(c1, '^[[A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]$') or
            mw.ustring.match(c1, '^[[A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]' .. c1)
        then
            c1 = '#' .. c1
        end
        if mw.ustring.match(c2, '^[[A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]$') or
            mw.ustring.match(c2, '^[[A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]' .. c2)
        then
            c2 = '#' .. c2
        end
        if mw.ustring.match(v3, '^[[A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]$') or
            mw.ustring.match(v3, '^[[A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]' .. c3)
        then
            c3 = '#' .. c3
        end
        return 'background-color:' .. c1 .. '; color:' .. ((ratio1 > 0) and c2 or c3)
    end

    return (ratio1 > 0) and (ratio2 > 0) and ((ratio1 + bias > ratio2) and c2 or c3)
end

function p._ratio(args)
    local v1 = color2lum(args[1])
    local v2 = color2lum(args[2])
    if (type(v1) == 'number' and type(v2) == 'number') then
        -- v1 should be the brighter of the two.
        if v2 > v1 then
            v1, v2 = v2, v1
        end
        return (v1 + 0.05)/(v2 + 0.05)
    else
        return args['error'] or '?'
    end
end

function p._styleratio(args)
    local style = (args[1] or ''):lower()
    local bg, fg = 'white', 'black'
    local lum_bg, lum_fg = 1, 0

    if args[2] then
        local lum = color2lum(args[2])
    end
end
```

```
        if lum ~= '' then bg, lum_bg = args[2], lum end
    end
    if args[3] then
        local lum = color2lum(args[3])
        if lum ~= '' then fg, lum_fg = args[3], lum end
    end

    local slist = mw.text.split(mw.ustring.gsub(mw.ustring.gsub(style or '',
for k = 1,#slist do
    local s = slist[k]
    local k,v = s:match( '^[%s]*([^:]-):([^:]-)[%s;]*$' )
    k = k or ''
    v = v or ''
    if (k:match('^[%s]*(background)[%s]*$') or k:match('^[%s]*(backgr
        local lum = color2lum(v)
        if( lum ~= '' ) then bg, lum_bg = v, lum end
    elseif (k:match('^[%s]*(color)[%s]*$')) then
        local lum = color2lum(v)
        if( lum ~= '' ) then bg, lum_fg = v, lum end
    end
end
if lum_bg > lum_fg then
    return (lum_bg + 0.05)/(lum_fg + 0.05)
else
    return (lum_fg + 0.05)/(lum_bg + 0.05)
end
end

--[[
Use {{#invoke:Color contrast|somecolor}} directly or
{{#invoke:Color contrast}} from a wrapper template.

Parameters:
-- |1=          - required; A color to check.
--]]
function p.lum(frame)
    local color = frame.args[1] or frame:getParent().args[1]
    return p._lum(color)
end

function p.ratio(frame)
    local args = frame.args[1] and frame.args or frame:getParent().args
    return p._ratio(args)
end

function p.styleratio(frame)
    local args = frame.args[1] and frame.args or frame:getParent().args
    return p._styleratio(args)
end

function p.greatercontrast(frame)
    local args = frame.args[1] and frame.args or frame:getParent().args
    return p._greatercontrast(args)
end

return p
```

Modul:Color contrast/colors

This module contains the **relative luminance** of **HTML colors**. It is used in **Module:Color contrast**.

```
return {
  aliceblue           = 0.92880068253475,
  antiquewhite       = 0.84646951707754,
  aqua               = 0.7874,
  aquamarine         = 0.8078549208338,
  azure              = 0.97265264954166,
  beige              = 0.8988459998705,
  bisque             = 0.80732327372979,
  black              = 0,
  blanchetalmond    = 0.85084439608156,
  blue               = 0.0722,
  blueviolet         = 0.12622014321946,
  brown              = 0.098224287876511,
  burlywood          = 0.51559844533893,
  cadetblue          = 0.29424681085422,
  chartreuse         = 0.76032025902623,
  chocolate          = 0.23898526114557,
  coral              = 0.37017930872924,
  cornflowerblue     = 0.30318641994179,
  cornsilk           = 0.93562110372965,
  crimson            = 0.16042199953026,
  cyan               = 0.7874,
  darkblue           = 0.018640801980939,
  darkcyan           = 0.20329317839046,
  darkgoldenrod      = 0.27264703559993,
  darkgray           = 0.39675523072563,
  darkgreen          = 0.091143429047575,
  darkgrey           = 0.39675523072563,
  darkkhaki          = 0.45747326349994,
  darkmagenta        = 0.07353047651207,
  darkolivegreen     = 0.12651920884889,
  darkorange         = 0.40016167026524,
  darkorchid         = 0.13413142174857,
  darkred            = 0.054889674531132,
  darksalmon         = 0.40541471563381,
  darkseagreen       = 0.43789249325969,
  darkslateblue      = 0.065792846227988,
  darkslategray      = 0.067608151928044,
  darkslategrey      = 0.067608151928044,
  darkturquoise      = 0.4874606277449,
  darkviolet         = 0.10999048339343,
  deeppink           = 0.23866895828276,
  deepskyblue        = 0.44481603395575,
  dimgray            = 0.14126329114027,
  dimgrey            = 0.14126329114027,
  dodgerblue         = 0.27442536991456,
  firebrick          = 0.10724525535015,
  floralwhite        = 0.95922484825004,
  forestgreen        = 0.18920812076002,
  fuchsia            = 0.2848,
  gainsboro          = 0.71569350050648,
  ghostwhite         = 0.94311261886323,
  gold               = 0.69860877428159,
  goldenrod          = 0.41919977809569,
  gray               = 0.2158605001139,
```



green	= 0.15438342968146,
greenyellow	= 0.80609472611453,
grey	= 0.2158605001139,
honeydew	= 0.96336535554782,
hotpink	= 0.34658438169715,
indianred	= 0.21406134963884,
indigo	= 0.03107561486337,
ivory	= 0.99071270600615,
khaki	= 0.77012343394121,
lavender	= 0.80318750514521,
lavenderblush	= 0.90172748631046,
lawngreen	= 0.73905893124963,
lemonchiffon	= 0.94038992245622,
lightblue	= 0.63709141280807,
lightcoral	= 0.35522120733135,
lightcyan	= 0.94587293494829,
lightgoldenrodyellow	= 0.93348351018297,
lightgray	= 0.65140563741982,
lightgreen	= 0.69091979956865,
lightgrey	= 0.65140563741982,
lightpink	= 0.58566152734898,
lightsalmon	= 0.4780675225206,
lightseagreen	= 0.35050145117042,
lightskyblue	= 0.56195637618331,
lightslategray	= 0.23830165007287,
lightslategrey	= 0.23830165007287,
lightsteelblue	= 0.53983888284666,
lightyellow	= 0.98161818392882,
lime	= 0.7152,
limegreen	= 0.44571042246098,
linen	= 0.88357340984379,
magenta	= 0.2848,
maroon	= 0.045891942324215,
mediumaquamarine	= 0.49389703310801,
mediumblue	= 0.044077780212328,
mediumorchid	= 0.21639251153773,
mediumpurple	= 0.22905858091648,
mediumseagreen	= 0.34393112338131,
mediumslateblue	= 0.20284629471622,
mediumspringgreen	= 0.70704308194184,
mediumturquoise	= 0.5133827926448,
mediumvioletred	= 0.14371899849357,
midnightblue	= 0.02071786635086,
mintcream	= 0.97834604947588,
mistyrose	= 0.82183047859185,
moccasin	= 0.80083000991567,
navajowhite	= 0.76519682342785,
navy	= 0.015585128108224,
oldlace	= 0.91900633405549,
olive	= 0.20027537200568,
olivedrab	= 0.22593150951929,
orange	= 0.4817026703631,
orangered	= 0.25516243753416,
orchid	= 0.31348806761439,
palegoldenrod	= 0.78792647887614,
palegreen	= 0.77936759006353,
paleturquoise	= 0.76436077921714,
palevioletred	= 0.28754994117889,
papayawhip	= 0.87797100199835,
peachpuff	= 0.74905589878251,
peru	= 0.30113074877936,
pink	= 0.63271070702466,
plum	= 0.45734221587969,
powderblue	= 0.68254586500605,



purple	= 0.061477070432439,
rebeccapurple	= 0.07492341159447,
red	= 0.2126,
rosybrown	= 0.32319457649407,
royalblue	= 0.16663210743188,
saddlebrown	= 0.097922285020521,
salmon	= 0.36977241527596,
sandybrown	= 0.46628543696283,
seagreen	= 0.19734199706275,
seashell	= 0.92737862206922,
sienna	= 0.13697631337098,
silver	= 0.52711512570581,
skyblue	= 0.55291668518184,
slateblue	= 0.14784278062136,
slategray	= 0.20896704076536,
slategrey	= 0.20896704076536,
snow	= 0.96533341834849,
springgreen	= 0.73052306068529,
steelblue	= 0.20562642207625,
tan	= 0.48237604163921,
teal	= 0.16996855778968,
thistle	= 0.56818401093733,
tomato	= 0.30638612719415,
turquoise	= 0.5895536427578,
violet	= 0.40315452986676,
wheat	= 0.74909702820482,
white	= 1,
whitesmoke	= 0.91309865179342,
yellow	= 0.9278,
yellowgreen	= 0.50762957208707,

}