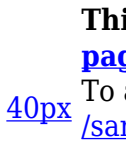
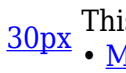
Module:Color contrast

From Blue Gold Program Wiki
The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

40x40px This module is rated as ready for general use. It has reached a mature form and is thought to be bug-free and ready for use wherever appropriate. It is ready to mention on help pages and other Wikipedia resources as an option for new users to learn. To reduce server load and bad output, it should be improved by sandbox testing rather than repeated trial-and-error editing.


This module is subject to page protection. It is a highly visible module in use by a very large number of pages, or is substituted very frequently. Because vandalism or mistakes would affect many pages, and even trivial editing might cause substantial load on the servers, it is protected from editing.


This Lua module is used on approximately 371,000 pages, or roughly 20819% of all pages.
To avoid major disruption and server load, any changes should be tested in the module's /sandbox or /testcases subpages, or in your own module sandbox. The tested changes can be added to this page in a single edit. Consider discussing changes on the talk page before implementing them.


This module depends on the following other modules:
• Module:Color contrast/colors

This module is used primarily by

{{Color contrast ratio}}
{{ColorToLum}} / {{RGBColorToLum}}
{{Color contrast conformance}}
{{Ensure AAA contrast ratio}}
{{Ensure AA contrast ratio}}
{{Greater color contrast ratio}}

It is also used for tracking within

Module:Navbox
Module:Userbox
Module:Episode list

and for documentation in

Module:College color

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.ustring.match(c,'^hsl%([%s]*[0-9][0-9%.]*[%s]*,[%s]*[0-9][0-9%.]*%%[%s]*,[
```

```
%s]*[0-9][0-9%.]*%%[%s]*%)$') then
                local h, s, l =
mw.ustring.match(c,'^hsl%([%s]*([0-9][0-9%.]*)[%s]*,[%s]*([0-9][0-9%.]*)%%[%s
]*,[%s]*([0-9][0-9%.]*)%%[%s]*%)$')
                return hsl2lum(tonumber(h), tonumber(s)/100, tonumber(l)/100)
        end

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

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

        -- remove leading # (if there is one) and whitespace
        c = mw.ustring.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]$') or
                        mw.ustring.match(c1, '^[A-Fa-f0-9][A-Fa-f0-9][A-Fa-
f0-9][A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]$') then
                                c1 = '#' .. c1
                end
                if mw.ustring.match(c2, '^[A-Fa-f0-9][A-Fa-f0-9][A-Fa-
f0-9]$') or
                        mw.ustring.match(c2, '^[A-Fa-f0-9][A-Fa-f0-9][A-Fa-
f0-9][A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]$') then
                                c2 = '#' .. c2
                end
                if mw.ustring.match(v3, '^[A-Fa-f0-9][A-Fa-f0-9][A-Fa-
f0-9]$') or
                        mw.ustring.match(v3, '^[A-Fa-f0-9][A-Fa-f0-9][A-Fa-
f0-9][A-Fa-f0-9][A-Fa-f0-9][A-Fa-f0-9]$') then
                                c3 = '#' .. c3
                end
```

```
                return 'background-color:' .. c1 .. '; color:' .. ((ratio1 >
0) and (ratio2 > 0) and ((ratio1 + bias > ratio2) and c2 or c3) or '') .. ';'
        end

        return (ratio1 > 0) and (ratio2 > 0) and ((ratio1 + bias > ratio2)
and c2 or c3) or ''
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])
                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
'', '&#[Xx]23;', '#'), '&#35;', '#'), ';')
        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]*(background%-color)[%s]*$')) then
                        local lum = color2lum(v)
                        if( lum ~= '' ) then bg, lum_bg = v, lum end
                elseif (k:match('^[%s]*(color)[%s]*$')) 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
```


Namespaces

- Module
- Discussion

Variants

Categories:

- Pages with script errors
- Pages with broken file links
- Modules for general use

- Modules subject to page protection

This page was last edited on 19 February 2020, at 09:52.

Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.