

☐ Toggle menu
Blue Gold Program Wiki

Navigation

- [Main page](#)
- [Recent changes](#)
- [Random page](#)
- [Help about MediaWiki](#)

Tools

- [What links here](#)
- [Related changes](#)
- [Special pages](#)
- [Permanent link](#)
- [Page information](#)

Personal tools

- [Log in](#)

personal-extra

☐ Toggle search

Search

Random page

Views

- [View](#)
- [View source](#)
- [History](#)
- [PDF Export](#)

Actions

Module:Box-header

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.

Documentation for this module may be created at [Module:Box-header/doc](#)

```

local p = {}
----- Config data -----
local namedColours = mw.loadData( 'Module:Box-header/colours' )
local modes = {
    lightest = { sat=0.10, val=1.00 },
    light     = { sat=0.15, val=0.95 },
    normal    = { sat=0.40, val=0.85 },
    dark      = { sat=0.90, val=0.70 },
    darkest   = { sat=1.00, val=0.45 },
    content    = { sat=0.04, val=1.00 },
    grey      = { sat=0.00 }
}
local min_contrast_ratio_normal_text = 7 -- i.e 7:1
local min_contrast_ratio_large_text  = 4.5 -- i.e. 4.5:1

-- Template parameter aliases
-- Specify each as either a single value, or a table of values
-- Aliases are checked left-to-right, i.e. `[one] = { 'two', 'three' }`
is equivalent to using `{{{one| {{{two| {{{three}}}}} }}}` in a template
local parameterAliases = {
    ['1'] = 1,
    ['2'] = 2,
    ['colour'] = 'color'
}

----- Dependencies -----
local colourContrastModule = require('Module:Color contrast')
local hex = require( 'luabit.hex' )

----- Utility functions -----
local function getParam(args, parameter)
    if args[parameter] then
        return args[parameter]
    end
    local aliases = parameterAliases[parameter]
    if not aliases then
        return nil
    end
    if type(aliases) ~= 'table' then
        return args[aliases]
    end
    for _, alias in ipairs(aliases) do
        if args[alias] then
            return args[alias]
        end
    end
    return nil
end

local function setCleanArgs(argsTable)
    local cleanArgs = {}

```

```

    for key, val in pairs(argsTable) do
        if type(val) == 'string' then
            val = val:match('^%s*(.)%s*$')
            if val ~= '' then
                cleanArgs[key] = val
            end
        else
            cleanArgs[key] = val
        end
    end
    return cleanArgs
end

-- Merge two tables into a new table. If there are any duplicate keys, the
values from the second overwrite the values from the first.
local function mergeTables(first, second)
    local merged = {}
    for key, val in pairs(first) do
        merged[key] = val
    end
    for key, val in pairs(second) do
        merged[key] = val
    end
    return merged
end

local function toOpenTagString(selfClosedHtmlObject)
    local closedTagString = tostring(selfClosedHtmlObject)
    local openTagString = mw.ustring.gsub(closedTagString, ' />$', '>')
    return openTagString
end

local function normaliseHexTriplet(hexString)
    if not hexString then return nil end
    local hexComponent = mw.ustring.match(hexString, '^#(%x%x%x)$') or
mw.ustring.match(hexString, '^#(%x%x%x%x%x%x)$')
    if hexComponent and #hexComponent == 6 then
        return mw.ustring.upper(hexString)
    end
    if hexComponent and #hexComponent == 3 then
        local r = mw.ustring.rep(mw.ustring.sub(hexComponent, 1, 1),
2)
        local g = mw.ustring.rep(mw.ustring.sub(hexComponent, 2, 2),
2)
        local b = mw.ustring.rep(mw.ustring.sub(hexComponent, 3, 3),
2)
        return '#' .. mw.ustring.upper(r .. g .. b)
    end
    return nil
end
end

```

----- Conversions -----

```
local function decimalToPaddedHex(number)
    local prefixedHex = hex.to_hex(tonumber(number)) -- prefixed with
    '0x'
    local padding = #prefixedHex == 3 and '0' or ''
    return mw.ustr.gsub(prefixedHex, '0x', padding)
end
local function hexToDecimal(hexNumber)
    return tonumber(hexNumber, 16)
end
local function RGBtoHexTriplet(R, G, B)
    return '#' .. decimalToPaddedHex(R) .. decimalToPaddedHex(G) ..
decimalToPaddedHex(B)
end
local function hexTripletToRGB(hexTriplet)
    local R_hex, G_hex, B_hex = string.match(hexTriplet,
'(%x%x)(%x%x)(%x%x)')
    return hexToDecimal(R_hex), hexToDecimal(G_hex), hexToDecimal(B_hex)
end
local function HSVtoRGB(H, S, V) -- per [[HSL and HSV#Converting_to_RGB]]
    local C = V * S
    local H_prime = H / 60
    local X = C * ( 1 - math.abs(math.fmod(H_prime, 2) - 1) )
    local R1, G1, B1
    if H_prime <= 1 then
        R1 = C
        G1 = X
        B1 = 0
    elseif H_prime <= 2 then
        R1 = X
        G1 = C
        B1 = 0
    elseif H_prime <= 3 then
        R1 = 0
        G1 = C
        B1 = X
    elseif H_prime <= 4 then
        R1 = 0
        G1 = X
        B1 = C
    elseif H_prime <= 5 then
        R1 = X
        G1 = 0
        B1 = C
    elseif H_prime <= 6 then
        R1 = C
        G1 = 0
        B1 = X
    end
    local m = V - C
    local R = R1 + m
```

```

    local G = G1 + m
    local B = B1 + m

    local R_255 = math.floor(R*255)
    local G_255 = math.floor(G*255)
    local B_255 = math.floor(B*255)
    return R_255, G_255, B_255
end
local function RGBtoHue(R_255, G_255, B_255) -- per [[HSL and HSV#Hue and
chroma]]
    local R = R_255/255
    local G = G_255/255
    local B = B_255/255

    local M = math.max(R, G, B)
    local m = math.min(R, G, B)
    local C = M - m
    local H_prime
    if C == 0 then
        return null
    elseif M == R then
        H_prime = math.fmod(((G - B)/C + 6), 6) -- adding six before
taking mod ensures positive value
    elseif M == G then
        H_prime = (B - R)/C + 2
    elseif M == B then
        H_prime = (R - G)/C + 4
    end
    local H = 60 * H_prime
    return H
end
local function nameToHexTriplet(name)
    if not name then return nil end
    local codename = mw.uststring.gsub(mw.uststring.lower(name), ' ', '')
    return namedColours[codename]
end

----- Choose colours -----
local function calculateColours(H, S, V, minContrast)
    local bgColour = RGBtoHexTriplet(HSVtoRGB(H, S, V))
    local textColour = colourContrastModule._greatercontrast({bgColour})
    local contrast = colourContrastModule._ratio({ bgColour, textColour
})
    if contrast >= minContrast then
        return bgColour, textColour
    elseif textColour == '#FFFFFF' then
        -- make the background darker and slightly increase the
saturation
        return calculateColours(H, math.min(1, S+0.005), math.max(0,
V-0.03), minContrast)
    else

```

```

        -- make the background lighter and slightly decrease the
saturation
        return calculateColours(H, math.max(0, S-0.005), math.min(1,
V+0.03), minContrast)
    end
end

local function makeColours(hue, modeName)
    local mode = modes[modeName]
    local isGrey = not(hue)
    if isGrey then hue = 0 end

    local borderSat = isGrey and modes.grey.sat or 0.15
    local border = RGBtoHexTriplet(HSVtoRGB(hue, borderSat, 0.75))

    local titleSat = isGrey and modes.grey.sat or mode.sat
    local titleBackground, titleForeground = calculateColours(hue,
titleSat, mode.val, min_contrast_ratio_large_text)

    local contentSat = isGrey and modes.grey.sat or modes.content.sat
    local contentBackground, contentForeground = calculateColours(hue,
contentSat, modes.content.val, min_contrast_ratio_normal_text)

    return border, titleForeground, titleBackground, contentForeground,
contentBackground
end

local function findHue(colour)
    local colourAsNumber = tonumber(colour)
    if colourAsNumber and ( -1 < colourAsNumber ) and ( colourAsNumber <
360) then
        return colourAsNumber
    end

    local colourAsHexTriplet = normaliseHexTriplet(colour) or
nameToHexTriplet(colour)
    if colourAsHexTriplet then
        return RGBtoHue(hexTripletToRGB(colourAsHexTriplet))
    end

    return null
end

local function normaliseMode(mode)
    if not mode or not modes[mw.ustring.lower(mode)] or
mw.ustring.lower(mode) == 'grey' then
        return 'normal'
    end
    return mw.ustring.lower(mode)
end

----- Build output -----

```

```

local function boxHeaderOuter(args)
    local baseStyle = {
        clear = 'both',
        ['box-sizing'] = 'border-box',
        border = ( getParam(args, 'border-type') or 'solid' ) .. ' '
        .. ( getParam(args, 'titleborder') or getParam(args, 'border') or '#ababab'
        ),
        background = getParam(args, 'titlebackground') or '#bcbcbc',
        color = getParam(args, 'titleforeground') or '#000',
        padding = getParam(args, 'padding') or '.1em',
        ['text-align'] = getParam(args, 'title-align') or 'center',
        ['font-family'] = getParam(args, 'font-family') or 'sans-
serif',
        ['font-size'] = getParam(args, 'titlefont-size') or '100%',
        ['margin-bottom'] = '0px',
    }

    local tag = mw.html.create('div', {selfClosing = true})
        :addClass('box-header-title-container')
        :addClass('flex-columns-noflex')
        :css(baseStyle)
        :css('border-width', ( getParam(args, 'border-top') or
getParam(args, 'border-width') or '1' ) .. 'px ' .. ( getParam(args, 'border-
width') or '1' ) .. 'px 0')
        :css('padding-top', getParam(args, 'padding-top') or '.1em')
        :css('padding-left', getParam(args, 'padding-left') or
'.1em')
        :css('padding-right', getParam(args, 'padding-right') or
'.1em')
        :css('padding-bottom', getParam(args, 'padding-bottom') or
'.1em')
        :css('moz-border-radius', getParam(args, 'title-border-
radius') or '0')
        :css('webkit-border-radius', getParam(args, 'title-border-
radius') or '0')
        :css('border-radius', getParam(args, 'title-border-radius')
or '0')
    return toOpenTagString(tag)
end

local function boxHeaderTopLinks(args)
    local style = {
        float = 'right',
        ['margin-bottom'] = '.1em',
        ['font-size'] = getParam(args, 'font-size') or '80%',
        color = getParam(args, 'titleforeground') or '#000'
    }
    local tag = mw.html.create('div', {selfClosing = true})
        :addClass('plainlinks noprint' )
        :css(style)

```

```

        return toOpenTagString(tag)
end

local function boxHeaderEditLink(args)
    local style = {
        color = getParam(args, 'titleforeground') or '#000'
    }
    local tag = mw.html.create('span')
        :css(style)
        :wikitext('edit')
    local linktext = tostring(tag)
    local linktarget = tostring(mw.uri.fullUrl(getParam(args,
'editpage'), {action='edit', section=getParam(args, 'section')}))
    return '[' .. linktarget .. ' ' .. linktext .. ']&nbsp;'
end

local function boxHeaderViewLink(args)
    local style = {
        color = getParam(args, 'titleforeground') or '#000'
    }
    local tag = mw.html.create('span')
        :css(style)
        :wikitext('view')
    local linktext = tostring(tag)
    local linktarget = ':' .. getParam(args, 'viewpage')
    return "<b>·</b>&nbsp;[" .. linktarget .. '|' .. linktext ..
']]&nbsp;'
end

local function boxHeaderTitle(args)
    local baseStyle = {
        ['font-family'] = getParam(args, 'title-font-family') or
'sans-serif',
        ['font-size'] = getParam(args, 'title-font-size') or '100%',
        ['font-weight'] = getParam(args, 'title-font-weight') or
'bold',
        border = 'none',
        margin = '0',
        padding = '0',
        color = getParam(args, 'titleforeground') or '#000';
    }
    local tagName = getParam(args, 'SPAN') and 'span' or 'h2'
    local tag = mw.html.create(tagName)
        :css(baseStyle)
        :css('padding-bottom', '.1em')
        :wikitext(getParam(args, 'title'))
    if getParam(args, 'extra') then
        local rules = mw.text.split(getParam(args, 'extra'), ';',
true)
        for _, rule in pairs(rules) do
            local parts = mw.text.split(rule, ':', true)

```



```

        local prop = parts[1]
        local val = parts[2]
        if prop and val then
            tag:css(prop, val)
        end
    end
end
return tostring(tag)
end

local function boxBody(args)
    local baseStyle = {
        ['box-sizing'] = 'border-box',
        border = ( getParam(args, 'border-width') or '1' ) .. 'px
solid ' .. ( getParam(args, 'border') or '#ababab'),
        ['vertical-align'] = 'top';
        background = getParam(args, 'background') or '#fefeef',
        opacity = getParam(args, 'background-opacity') or '1',
        color = getParam(args, 'foreground') or '#000',
        ['text-align'] = getParam(args, 'text-align') or 'left',
        margin = '0 0 10px',
        padding = getParam(args, 'padding') or '1em',
    }
    local tag = mw.html.create('div', {selfClosing = true})
        :css(baseStyle)
        :css('border-top-width', ( getParam(args, 'border-top') or
'1' ) .. 'px')
        :css('padding-top', getParam(args, 'padding-top') or '.3em')
        :css('-moz-border-radius', getParam(args, 'border-radius') or
'0')
        :css('-webkit-border-radius', getParam(args, 'border-radius')
or '0')
        :css('border-radius', getParam(args, 'border-radius') or '0')
    return toOpenTagString(tag)
end

local function contrastCategories(args)
    local cats = ''

    local titleText = nameToHexTriplet(getParam(args, 'titleforeground'))
or normaliseHexTriplet(getParam(args, 'titleforeground')) or '#000000'
    local titleBackground = nameToHexTriplet(getParam(args,
'titlebackground')) or normaliseHexTriplet(getParam(args, 'titlebackground'))
or '#bcbcbc'
    local titleContrast = colourContrastModule._ratio({titleBackground,
titleText})
    local insufficientTitleContrast = type(titleContrast) == 'number' and
( titleContrast < min_contrast_ratio_large_text )

    local bodyText = nameToHexTriplet(getParam(args, 'foreground')) or
normaliseHexTriplet(getParam(args, 'foreground')) or '#000000'

```

```

        local bodyBackground = nameToHexTriplet(getParam(args, 'background'))
or normaliseHexTriplet(getParam(args, 'background')) or '#fefeef'
        local bodyContrast = colourContrastModule._ratio({bodyBackground,
bodyText})
        local insufficientBodyContrast = type(bodyContrast) == 'number' and (
bodyContrast < min_contrast_ratio_normal_text )

        if insufficientTitleContrast and insufficientBodyContrast then
            return '[[Category:Box-header with insufficient title
contrast]][[Category:Box-header with insufficient body contrast]]'
        elseif insufficientTitleContrast then
            return '[[Category:Box-header with insufficient title
contrast]]'
        elseif insufficientBodyContrast then
            return '[[Category:Box-header with insufficient body
contrast]]'
        else
            return ''
        end
    end
end

----- Main functions / entry points -----

-- Entry point for templates (manually-specified colours)
function p.boxHeader(frame)
    local parent = frame.getParent(frame)
    local parentArgs = parent.args
    local page = parentArgs.editpage
    if not parentArgs.editpage or parentArgs.editpage == '' then
        page = parent:preprocess('{{FULLPAGENAME}}')
    end
    local output = p._boxHeader(parentArgs, page)
    if mw.ustring.find(output, '{') then
        return frame:preprocess(output)
    end
    return output
end

-- Entry point for modules (manually-specified colours)
function p._boxHeader(_args, page)
    local args = setCleanArgs(_args)
    if page and not args.editpage then
        args.editpage = page
    end
    if not args.title then
        args.title = '{{{title}}}'
    end
    local output = {}
    table.insert(output, boxHeaderOuter(args))
    if not getParam(args, 'EDITLINK') then
        table.insert(output, boxHeaderTopLinks(args))
    end
end

```

```

        if not getParam(args, 'noedit') then
            table.insert(output, boxHeaderEditLink(args))
        end
        if getParam(args, 'viewpage') then
            table.insert(output, boxHeaderViewLink(args))
        end
        if getParam(args, 'top') then
            table.insert(output, getParam(args, 'top') ..
'&nbsp;')
        end
        table.insert(output, '</div>')
    end
    table.insert(output, boxHeaderTitle(args))
    table.insert(output, '</div>')
    table.insert(output, boxBody(args))
    if not getParam(args, 'TOC') then
        table.insert(output, '__NOTOC__')
    end
    if not getParam(args, 'EDIT') then
        table.insert(output, '__NOEDITSECTION__')
    end
    table.insert(output, contrastCategories(args))

    return table.concat(output)
end

```

-- Entry point for templates (automatically calculated colours)

```

function p.autoColour(frame)
    local parent = frame.getParent(frame)
    local parentArgs = parent.args
    local colourParam = getParam(parentArgs, 'colour')
    local generatedColour = nil
    if not colourParam or colourParam == '' then
        -- convert the root page name into a number and use that
        local root = parent:preprocess('{{ROOTPAGENAME}}')
        local rootStart = mw.ustring.sub(root, 1, 12)
        local digitsFromRootStart = mw.ustring.gsub(rootStart, ".",
function(s) return math.fmod(string.byte(s, 2) or string.byte(s, 1), 10) end)
        local numberFromRoot = tonumber(digitsFromRootStart, 10)
        generatedColour = math.fmod(numberFromRoot, 360)
    end
    local output = p._autoColour(parent.args, generatedColour)
    if mw.ustring.find(output, '{') then
        return frame:preprocess(output)
    end
    return output
end
end

```

-- Entry point for modules (automatically calculated colours)

```

function p._autoColour(_args, generatedColour)
    local args = setCleanArgs(_args)

```

```

    local hue = generatedColour or findHue(getParam(args, 'colour'))
    local mode = normaliseMode(getParam(args, 'mode'))
    local border, titleForeground, titleBackground, contentForeground,
contentBackground = makeColours(hue, mode)
    local boxTemplateArgs = mergeTables(args, {
        title = getParam(args, '1') or '{{{1}}}',
        editpage = getParam(args, '2') or '',
        noedit = getParam(args, '2') and '' or 'yes',
        border = border,
        titleforeground = titleForeground,
        titlebackground = titleBackground,
        foreground = contentForeground,
        background = contentBackground
    })
    return p._boxHeader(boxTemplateArgs)
end
return p

```

Retrieved from "<https://www.bluegoldwiki.com/index.php?title=Module:Box-header&oldid=2415>"

Namespaces

- [Module](#)
- [Discussion](#)

Variants

This page was last edited on 26 April 2020, at 05:11.

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.

- [Privacy policy](#)
- [About Blue Gold Program Wiki](#)
- [Disclaimers](#)

Developed and maintained by Big Blue Communications for Blue Gold Program



[Blue Gold Program Wiki](#)