

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:List

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:List/doc](#)

```

-- This module outputs different kinds of lists. At the moment, bulleted,
-- unbulleted, horizontal, ordered, and horizontal ordered lists are
supported.

local libUtil = require('libraryUtil')
local checkType = libUtil.checkType
local mTableTools = require('Module:TableTools')

local p = {}

local listTypes = {
    ['bulleted'] = true,
    ['unbulleted'] = true,
    ['horizontal'] = true,
    ['ordered'] = true,
    ['horizontal_ordered'] = true
}

function p.makeListData(listType, args)
    -- Constructs a data table to be passed to p.renderList.
    local data = {}

    -- Classes
    data.classes = {}
    if listType == 'horizontal' or listType == 'horizontal_ordered' then
        table.insert(data.classes, 'hlist hlist-separated')
    elseif listType == 'unbulleted' then
        table.insert(data.classes, 'plainlist')
    end
    table.insert(data.classes, args.class)

    -- Main div style
    data.style = args.style

    -- Indent for horizontal lists
    if listType == 'horizontal' or listType == 'horizontal_ordered' then
        local indent = tonumber(args.indent)
        indent = indent and indent * 1.6 or 0
        if indent > 0 then
            data.marginLeft = indent .. 'em'
        end
    end

    -- List style types for ordered lists
    -- This could be "1, 2, 3", "a, b, c", or a number of others. The
list style
    -- type is either set by the "type" attribute or the "list-style-
type" CSS
    -- property.
    if listType == 'ordered' or listType == 'horizontal_ordered' then
        data.liststyleType = args.list_style_type or args['list-
style-type']
    end
end

```

```

        data.type = args['type']

        -- Detect invalid type attributes and attempt to convert them
to
        -- list-style-type CSS properties.
        if data.type
            and not data.listStyleType
            and not tostring(data.type):find('^%s*[1AaIi]%s*$')
        then
            data.listStyleType = data.type
            data.type = nil
        end
    end
    -- List tag type
    if listType == 'ordered' or listType == 'horizontal_ordered' then
        data.listTag = 'ol'
    else
        data.listTag = 'ul'
    end
    end

    -- Start number for ordered lists
    data.start = args.start
    if listType == 'horizontal_ordered' then
        -- Apply fix to get start numbers working with horizontal
ordered lists.
        local startNum = tonumber(data.start)
        if startNum then
            data.counterReset = 'listitem ' .. tostring(startNum
- 1)
        end
    end
    end

    -- List style
    -- ul_style and ol_style are included for backwards compatibility.
No
    -- distinction is made for ordered or unordered lists.
    data.listStyle = args.list_style

    -- List items
    -- li_style is included for backwards compatibility. item_style was
included
    -- to be easier to understand for non-coders.
    data.itemStyle = args.item_style or args.li_style
    data.items = {}
    for i, num in ipairs(mTableTools.numKeys(args)) do
        local item = {}
        item.content = args[num]
        item.style = args['item' .. tostring(num) .. '_style']
            or args['item_style' .. tostring(num)]
        item.value = args['item' .. tostring(num) .. '_value']
            or args['item_value' .. tostring(num)]
    end
end

```

```

        table.insert(data.items, item)
    end
    return data
end

function p.renderList(data)
    -- Renders the list HTML.
    -- Return the blank string if there are no list items.
    if type(data.items) ~= 'table' or #data.items < 1 then
        return ''
    end
    -- Render the main div tag.
    local root = mw.html.create('div')
    for i, class in ipairs(data.classes or {}) do
        root.addClass(class)
    end
    root.css[['margin-left'] = data.marginLeft]
    if data.style then
        root.cssText(data.style)
    end

    -- Render the list tag.
    local list = root:tag(data.listTag or 'ul')
    list
        :attr{start = data.start, type = data.type}
        :css{
            ['counter-reset'] = data.counterReset,
            ['list-style-type'] = data.listStyleType
        }
    if data.listStyle then
        list.cssText(data.listStyle)
    end

    -- Render the list items
    for i, t in ipairs(data.items or {}) do
        local item = list:tag('li')
        if data.itemStyle then
            item.cssText(data.itemStyle)
        end
        if t.style then
            item.cssText(t.style)
        end
        item
            :attr{value = t.value}
            :wikitext(t.content)
    end

    return tostring(root)
end

function p.renderTrackingCategories(args)

```

```

local isDeprecated = false -- Tracks deprecated parameters.
for k, v in pairs(args) do
    k = tostring(k)
    if k:find('^item_style%d+$') or k:find('^item_value%d+$')
then
        isDeprecated = true
        break
    end
end
local ret = ''
if isDeprecated then
    ret = ret .. '[[Category:List templates with deprecated
parameters]]'
end
return ret
end

function p.makeList(listType, args)
    if not listType or not listTypes[listType] then
        error(string.format(
            "bad argument #1 to 'makeList' ('%s' is not a valid
list type)",
            tostring(listType)
        ), 2)
    end
    checkType('makeList', 2, args, 'table')
    local data = p.makeListData(listType, args)
    local list = p.renderList(data)
    local trackingCategories = p.renderTrackingCategories(args)
    return list .. trackingCategories
end

for listType in pairs(listTypes) do
    p[listType] = function (frame)
        local mArguments = require('Module:Arguments')
        local origArgs = mArguments.getArgs(frame, {
            valueFunc = function (key, value)
                if not value or not mw.ustring.find(value, '%S') then
                    if mw.ustring.find(value, '^%s*[%*#;:]') then
                        return value
                    else
                        return value:match('^%s*(.-)%s*$')
                    end
                end
                return nil
            end
        })
        -- Copy all the arguments to a new table, for faster
indexing.
        local args = {}
        for k, v in pairs(origArgs) do

```

```
                args[k] = v
            end
        return p.makeList(listType, args)
    end
end

return p
```

Retrieved from "<https://www.bluegoldwiki.com/index.php?title=Module:List&oldid=370>"

Namespaces

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

Variants

This page was last edited on 19 February 2020, at 06:45.

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](#)