List:پودمان

[illegible] [illegible] [illegible]/List:[illegible] [illegible] [illegible] [

```
	-- 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
			-- برای فارسی‌سازی بهتر، نام متغیر زیر از marginLeft به marginRight تغییر داده شده‌است
			data.marginRight = 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']
		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
	-- Start number for ordered lists
	data.start = num_con("en", args.start)

	-- Two new persian translation for the start parameter
	if args['آغاز'] then
		data.start = num_con("en", args['آغاز'])
	end
	if args['شروع'] then
		data.start = num_con("en", args['شروع'])
	end
```

```
        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

        -- 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 _, 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)]
                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 _, class in ipairs(data.classes or {}) do
                root:addClass(class)
        end
        -- برای فارسی‌سازی بهتر، نام متغیر زیر از marginLeft به marginRight
تغییر داده شده‌است
        root:css{['margin-right'] = data.marginRight}
        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 _, 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 data.templatestyles .. 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 .. '[[رده:الگو:همراه پارامترهای بددانسته]]'
        end
        return ret
end

function p.makeList(listType, args)
        if not listType or not listTypes[listType] then
                error(string.format(
                        "نوع فهرست معتبر «%s» (آرگومان بد #1 برای «makeList»
نمی‌باشد)",
```

```
                        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
return nil end
                        if mw.ustring.find(value, '^%s*[%*#;:]') then
                                return value
                        else
                                return value:match('^%s*(.-)%s*$')
                        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
```