Inhaltsverzeichnis

1. Modul:Aligned table 2
2. Modul:Aligned table/Doku 5

Modul:Aligned table

Implements Vorlage:Tl

```
-- This module implements {{aligned table}}
local p = {}

local function isnotempty(s)
	return s and s:match( '^%s*(.-)%s*$' ) ~= ''
end

function p.table(frame)
	local args = (frame.args[3] ~= nil) and frame.args or frame:getParent().a
	local entries = {}
	local colclass = {}
	local colstyle = {}
	local cols = tonumber(args['cols']) or 2

	-- create the root table
	local root = mw.html.create('table')

	-- add table style for fullwidth
	if isnotempty(args['fullwidth']) then
		root
			:css('width', '100%')
			:css('border-collapse', 'collapse')
			:css('border-spacing', '0px 0px')
			:css('border', 'none')
	end

	-- add table classes
	if isnotempty(args['class']) then
		root:addClass(args['class'])
	end

	-- add table style
	if isnotempty(args['style']) then
		root:cssText(args['style'])
	end

	-- build arrays with the column styles and classes
	if isnotempty(args['leftright']) then
		colstyle[1] = 'text-align:left;'
		colstyle[2] = 'text-align:right;'
	end
	if isnotempty(args['rightleft']) then
		colstyle[1] = 'text-align:right;'
		colstyle[2] = 'text-align:left;'
	end
	for i = 1,cols do
		colclass[ i ] = colclass[ i ] or ''
		colstyle[ i ] = colstyle[ i ] or ''
		if isnotempty(args['colstyle']) then
			colstyle[ i ] = args['colstyle'] .. ';' .. colstyle[ i ]
		end
		if isnotempty(args['colalign' .. tostring(i)]) then
			colstyle[ i ] = 'text-align:' .. args['colalign' .. tostr
		elseif isnotempty(args['col' .. tostring(i) .. 'align']) then
			colstyle[ i ] = 'text-align:' .. args['col' .. tostring(i
```

```
                elseif isnotempty(args['align' .. tostring(i)]) then
                        colstyle[ i ] = 'text-align:' .. args['align' .. tostring
                end
                if isnotempty(args['colnowrap' .. tostring(i)]) then
                        colstyle[ i ] = 'white-space:nowrap;' .. colstyle[ i ]
                elseif isnotempty(args['col' .. tostring(i) .. 'nowrap']) then
                        colstyle[ i ] = 'white-space:nowrap;' .. colstyle[ i ]
                elseif isnotempty(args['nowrap' .. tostring(i)]) then
                        colstyle[ i ] = 'white-space:nowrap;' .. colstyle[ i ]
                end
                if isnotempty(args['colwidth' .. tostring(i)]) then
                        colstyle[ i ] = 'width:' .. args['colwidth' .. tostring(i
                elseif isnotempty(args['col' .. tostring(i) .. 'width']) then
                        colstyle[ i ] = 'width:' .. args['col' .. tostring(i) ..
                elseif isnotempty(args['colwidth']) then
                        colstyle[ i ] = 'width:' .. args['colwidth'] .. ';' .. co
                end
                if isnotempty(args['colstyle' .. tostring(i)]) then
                        colstyle[ i ] = colstyle[ i ] .. args['colstyle' .. tostr
                elseif isnotempty(args['col' .. tostring(i) .. 'style']) then
                        colstyle[ i ] = colstyle[ i ] .. args['col' .. tostring(i
                elseif isnotempty(args['style' .. tostring(i)]) then
                        colstyle[ i ] = colstyle[ i ] .. args['style' .. tostring
                end
                if isnotempty(args['colclass' .. tostring(i)]) then
                        colclass[ i ] =  args['colclass' .. tostring(i)]
                elseif isnotempty(args['col' .. tostring(i) .. 'class']) then
                        colclass[ i ] =  args['col' .. tostring(i) .. 'class']
                elseif isnotempty(args['class' .. tostring(i)]) then
                        colclass[ i ] =  args['class' .. tostring(i)]
                end
        end
        -- compute the maximum cell index
        local cellcount = 0
        for k, v in pairs( args ) do
                if type( k ) == 'number' then
                        cellcount = math.max(cellcount, k)
                end
        end
        -- compute the number of rows
        local rows = math.ceil(cellcount / cols)

        -- build the table content
        if isnotempty(args['title']) then
                local caption = root:tag('caption')
                caption:cssText(args['titlestyle'])
                caption:wikitext(args['title'])
        end
        if isnotempty(args['above']) then
                local row = root:tag('tr')
                local cell = row:tag('th')
                cell:attr('colspan', cols)
                cell:cssText(args['abovestyle'])
                cell:wikitext(args['above'])
        end
        for j=1,rows do
                -- start a new row
                local row = root:tag('tr')
                if isnotempty(args['rowstyle']) then
                        row:cssText(args['rowstyle'])
                else
                        row:css('vertical-align', 'top')
                end
                if isnotempty(args['rowclass']) then
```

```
                        row:addClass(args['rowclass'])
                end
                -- loop over the cells in the row
                for i=1,cols do
                        local cell
                        if isnotempty(args['row' .. tostring(j) .. 'header']) the
                                cell = row:tag('th'):attr('scope','col')
                        elseif isnotempty(args['col' .. tostring(i) .. 'header'])
                                cell = row:tag('th'):attr('scope','row')
                        else
                                cell = row:tag('td')
                        end
                        if args['class' .. tostring(j) .. '.' .. tostring(i)] the
                                cell:addClass(args['class' .. tostring(j) .. '.'
                        else
                                if args['rowclass' .. tostring(j)] then
                                        cell:addClass(args['rowclass' .. tostring
                                elseif args['row' .. tostring(j) .. 'class'] then
                                        cell:addClass(args['row' .. tostring(j) .
                                elseif args['rowevenclass'] and math.fmod(j,2) ==
                                        cell:addClass(args['rowevenclass'])
                                elseif args['rowoddclass'] and math.fmod(j,2) ==
                                        cell:addClass(args['rowoddclass'])
                                end
                                if colclass[i] ~= '' then
                                        cell:addClass(colclass[i])
                                end
                        end
                        if args['style' .. tostring(j) .. '.' .. tostring(i)] the
                                cell:cssText(args['style' .. tostring(j) .. '.' .
                        else
                                if args['rowstyle' .. tostring(j)] then
                                        cell:cssText(args['rowstyle' .. tostring(
                                elseif args['rowevenstyle'] and math.fmod(j,2) ==
                                        cell:cssText(args['rowevenstyle'])
                                elseif args['rowoddstyle'] and math.fmod(j,2) ==
                                        cell:cssText(args['rowoddstyle'])
                                elseif args['row' .. tostring(j) .. 'style'] then
                                        cell:cssText(args['row' .. tostring(j) .
                                end
                                if isnotempty(colstyle[i]) then
                                        cell:cssText(colstyle[i])
                                end
                        end
                        cell:wikitext(mw.ustring.gsub(args[cols*(j - 1) + i] or '
                end
        end
        -- return the root table
        return tostring(root)
end

return p
```

Modul:Aligned table/Doku

Dies ist die Dokumentationsseite für Modul:Aligned table

Implements Vorlage:Tl