



Inhaltsverzeichnis

1. Modul:Chart	2
2. Vorlage:Graph:Chart	22

Modul:Chart

Mobile View Problem: Bar charts behave unpredictably, causing problems with the axes and legend. Use [Template:Graph:Chart](#) instead. Pie charts aren't too bad. Module Chart exports two functions: bar chart and pie chart

Note - [Template:Graph:Chart](#) is an alternative template, that may be more suitable for your use case.

Inhaltsverzeichnis	
1 Drawing Bar charts: "bar chart"	2
1.1 Parameters	2
1.2 Examples	4
1.2.1 Basic	4
1.2.2 Stacked	5
1.2.3 Scale per group	6
2 Drawing Pie charts: "pie chart"	8
2.1 Parameters	8
2.2 Examples	12

Drawing Bar charts: "bar chart"

Parameters

parameter name	what it does
delimiter	string to delimit multiple values when given. default to colon (:). normally you do not want to touch this, it's provided for the off-chance you'll want to use colon as part of one of the parameters.
width	number. if provided, must be at least 200. default: 500
height	number. if provided, must be at least 200. default: 350
group n	(where "n" is a number. use "group 1", "group 2" etc. for as many groups as there are in the graph) the values to be charted. see below.
tooltip	tooltip to be associated with specific bar. If no tooltip for a specific bar is defined, and this bar has a link, then this link will be used as tooltip. Otherwise, the tooltip will be

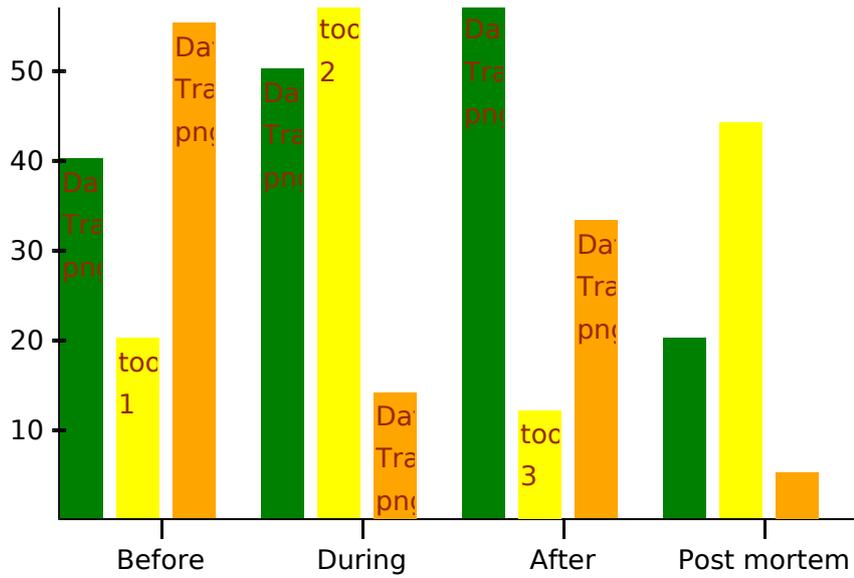
parameter name	what it does
tip n	combined from the group name and the value, optionally with "units prefix" and "units suffix".
link s n	links to articles to be associated with specific bar
stack	whether to stack the different groups on top of each other. do not specify to show bars side by side. Any non-empty value means "yes". To say "no", simply do not supply this parameter at all, or leave the value blank.
Vorlage: No wrap	useful only with stack: when set to true, tooltip will show accumulated value of all blocks up to current one
colors	the colors used to denote the various groups. should have exactly as many values as # of groups. can be given as standard html-recognized color names, or using #xxx or #xxxxxx notation.
x leg end s	The legends for the X values. Wikicode, such as internal links or templates can be used.
Vorlage: No wrap	if set to true, group legends will not be shown below chart. Any non-empty value means "yes". To say "no", simply do not supply this parameter at all, or leave the value blank.
Vorlage: No wrap	set to use separate Y- scale for each group. leave empty to use one scale for all groups. incompatible with "stack". Note that even if some of the scales are exactly the same, they will be drawn separately when this setting is on. Any non-empty value means "yes". To say "no", simply do not supply this parameter at all, or leave the value blank.
Vorlage: No	

parameter name	what it does
wrap	used in tooltip. e.g., \$, so values will show as "\$500" instead of "500" in the tooltip
Vorlage : No wrap	ditto for units suffix. use, e.g. "Kg" so values will show as 88Kg instead of 88 in tooltip. underscore ("_") are replaced by spaces, to allow a space between the value and the suffix.
Vorlage : No wrap	names of different groups
Vorlage : No wrap	number of tick marks on the y axis. if the value is negative or omitted, the module will attempt to automatically calculate a sensible number of tick marks.

Examples

Basic

```
{{ #invoke:Chart | bar chart
| group 1 = 40 : 50 : 60 : 20
| group 2 = 20 : 60 : 12 : 44
| group 3 = 55 : 14 : 33 : 5
| links 1 = Apple : McCintosh : Golden delicious
| links 2 = Banana : Apricot : Peach
| links 3 = Orange : Pear : Bear
| tooltip 2 = tooltip 1 : tooltip 2 : tooltip 3 : tooltip 4
| colors = green : yellow : orange
| group names = Apple : Banana : Orange
| x legends = Before : During : After : Post mortem
}}
```



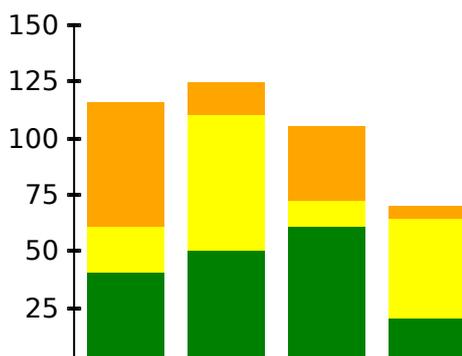
Stacked

Here is the same graph, with more modest height and width, using "stack", and adding "units suffix" for good measure:

```

{{ #invoke:Chart | bar chart
| height = 250
| width = 300
| stack = 1
| group 1 = 40 : 50 : 60 : 20
| group 2 = 20 : 60 : 12 : 44
| group 3 = 55 : 14 : 33 : 5
| colors = green : yellow : orange
| group names = Apple : Banana : Orange
| units suffix = Kg
| x legends = Before : During : After : Post mortem
}}

```



Before During After Post
mortem



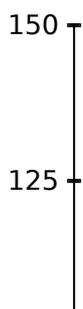
Scale per group

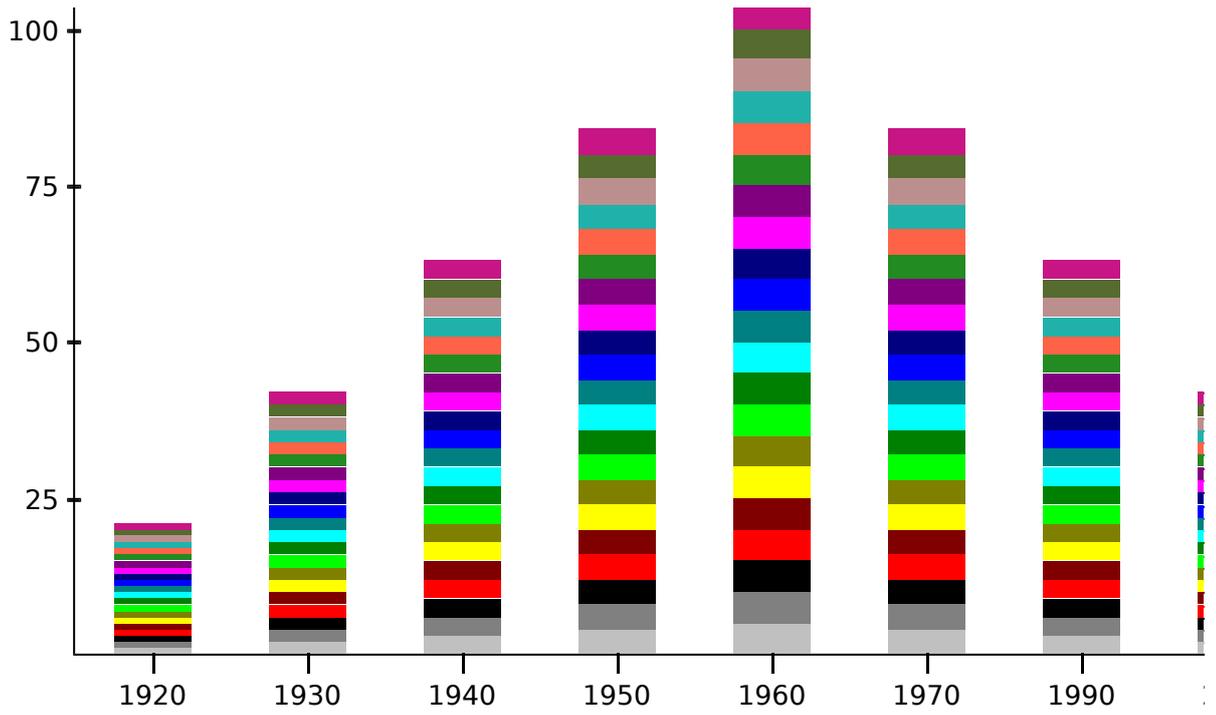
This option has been disabled. It was rarely used and broke in the last code update. Here is an example with large number of groups - mainly to test how it looks with large number of legends:

```

{{ #invoke:Chart | bar chart
  width = 800
  height = 550
  group 1 = 1:2:3:4:5:4:3:2:1
  group 2 = 1:2:3:4:5:4:3:2:1
  group 3 = 1:2:3:4:5:4:3:2:1
  group 4 = 1:2:3:4:5:4:3:2:1
  group 5 = 1:2:3:4:5:4:3:2:1
  group 6 = 1:2:3:4:5:4:3:2:1
  group 7 = 1:2:3:4:5:4:3:2:1
  group 8 = 1:2:3:4:5:4:3:2:1
  group 9 = 1:2:3:4:5:4:3:2:1
  group 10 = 1:2:3:4:5:4:3:2:1
  group 11 = 1:2:3:4:5:4:3:2:1
  group 12 = 1:2:3:4:5:4:3:2:1
  group 13 = 1:2:3:4:5:4:3:2:1
  group 14 = 1:2:3:4:5:4:3:2:1
  group 15 = 1:2:3:4:5:4:3:2:1
  group 16 = 1:2:3:4:5:4:3:2:1
  group 17 = 1:2:3:4:5:4:3:2:1
  group 18 = 1:2:3:4:5:4:3:2:1
  group 19 = 1:2:3:4:5:4:3:2:1
  group 20 = 1:2:3:4:5:4:3:2:1
  group 21 = 1:2:3:4:5:4:3:2:1
  colors = Silver:Gray:Black:Red:Maroon:Yellow:Olive:Lime:Green:Aqua:Teal:Blue:Na
  group names = Alabama:Alaska:Arizona:Arkansas:California:Colorado:Connecticut:
  x legends = 1920 : 1930 : 1940: 1950 : 1960 : 1970 : 1990 : 2000 : 2010
  units prefix = $
  units suffix = _billion
  stack = 1
}}

```





- Alabama
- Alaska
- Arizona
- Arkansas
- California
- Colorado
- Connecticut
- Delaware
- Florida
- Georgia
- Hawaii
- Idaho
- Illinois
- Indiana
- Iowa
- Kansas
- Kentucky
- Louisiana
- Maine
- Maryland
- Massachusetts

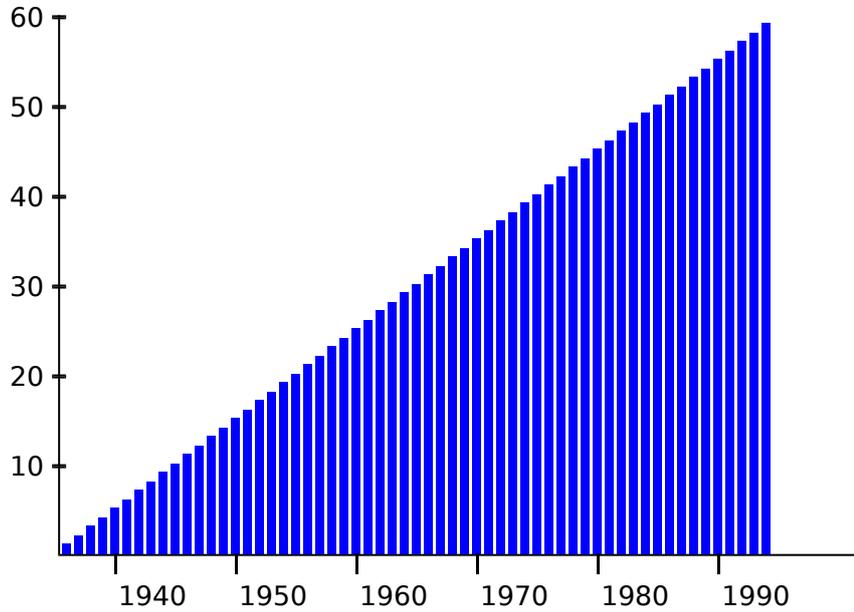
If there are many values, x legends can be diluted by using delimiters with nothing in between:



```

{{ #invoke:Chart | bar chart
| group 1 = 1:2:3:4:5:6:7:8:9:10:11:12:13:14:15:16:17:18:19:20:21:22:23:24:25:26:
:31:32:33:34:35:36:37:38:39:40:41:42:43:44:45:46:47:48:49:50:51:52:53:54:55:56:57
| units suffix = _Things
| group names = Some
| x legends = ::::1940::::::::::1950::::::::::1960::::::::::1970::::::::::1980::
}}

```



Vorlage:-

Drawing Pie charts: "pie chart"

Parameters

<p>what it does</p>
<p>parameter name</p>
<p>detail</p>

parameter name	what it does
delimiter	string to delimit multiple values when given. default to colon (:). normally you do not want to touch this, it's provided for the off-chance you'll want to use colon as part of one of the parameters.
radius	number. The radius of the pie in pixels
slices	<p>Tuples, in parenthesis. Use delimiter inside the tuple:</p> <pre>(Value1 : Name1 : Color1 : Link1) (Value2 : Name2 : Color2 : Link2) ...</pre> <p>The values are numbers. The numbers can be integers or decimal fractions, or using the scientific notation: 7.24e6, 7,240,000, or 7240000.00 are all acceptable for 7 Million and 240 thousands.</p> <p>Names are strings. Colors are optional. you can use any Web colors, such as "red" or "#FF0000". Up to 26 default colors are defined, but if your pie has more than 26 slices, you must define the colors of slice #27 and up. Links can be external or internal links, including linking to internal anchors and paragraphs in the same article, like so: [[Article Tooltip]] for internal link, [[#Paragraph name Tooltip]] for linking to an anchor in same article, or [http://example.org Tooltip] for external link.</p>
or	

p a r a m e t e r n a m e	what it does
l a g e N o w r a p	<p>alternative syntax to "slices". n is the slice number, beginning with 1. make sure not to skip: if you define "slice 1", "slice 2", "slice 4", "slice 5"..., skipping slice 3, only the first two slices will be shown. this syntax is incompatible with "slices", i.e., they should not be used in conjunction in the same invocation. Using both "slices" and "slice n" in the same invocation will cause unpredictable results. The value is like a single "tuple" as explained above, but without the parenthesis:</p> <pre>slice 1 = Value1 : Name1 : Color1 : Link1 slice 2 = Value2 : Name2 : Color2 : Link2 ...</pre> <p>This syntax allows you to use parenthesis in names, links, and colors.</p>
p e r c e n t	<p>if used, the percentage of each slice will be calculated and added to the legend: so if you have two slices, like so: (1 : Younglings) (3 : elders), and use define "percent", the legends will become "Younglings: 1 (25%)" and "Elders: 3 (75%)", instead of simply "Younglings: 1" and "Elders: 3". Any non-empty value means "yes". To say "no", simply do not supply this parameter at all, or leave the value blank.</p>
V o r l a g e :	



<p>p a r a m e t e r n a m e</p>	<p>what it does</p>
<p>N o w r a p</p>	<p>used in the legend. e.g., defining "units prefix=\$", values will show as "\$500" instead of "500" in the legends</p>
<p>V o r l a g e : N o w r a p</p>	<p>ditto for units suffix. use, e.g. "Kg" so values will show as 88Kg instead of 88 in legend. underscore ("_") are replaced by spaces, to allow a space between the value and the suffix.</p>
<p>V o r l a g e :</p>	

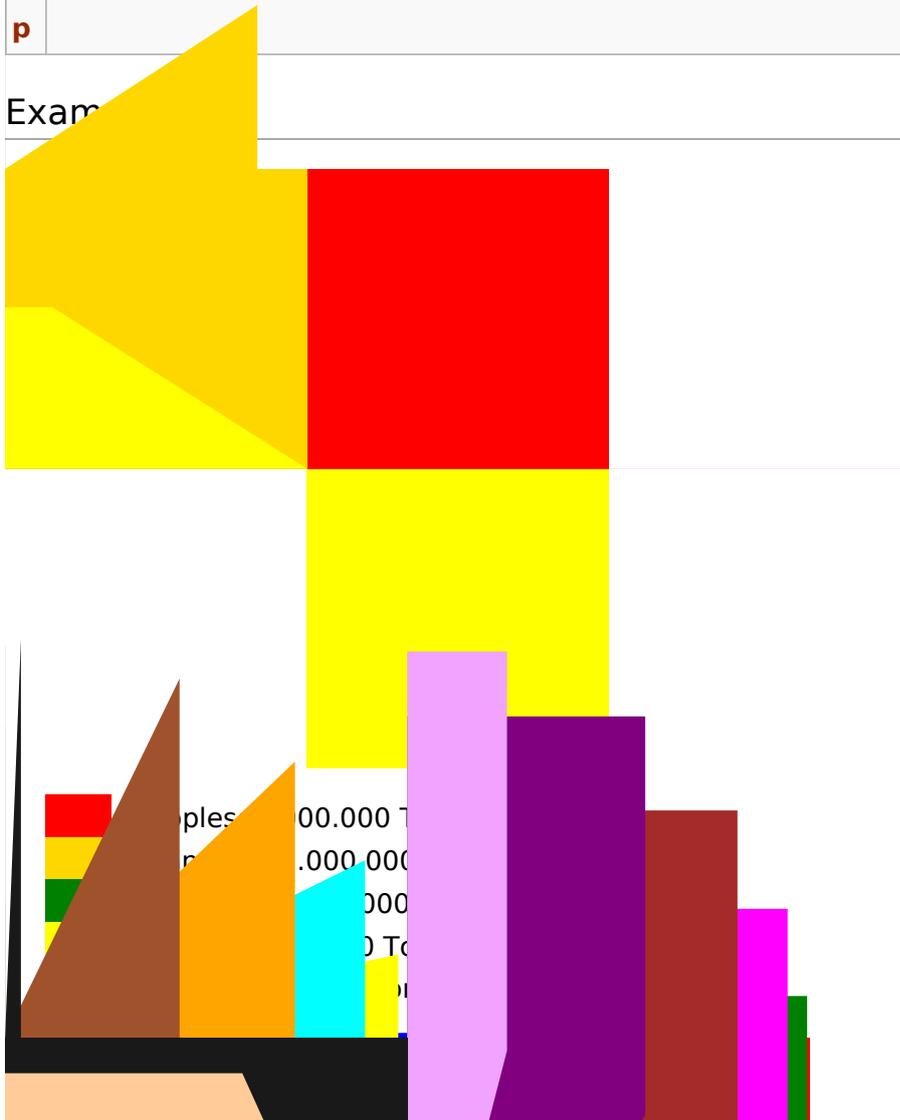


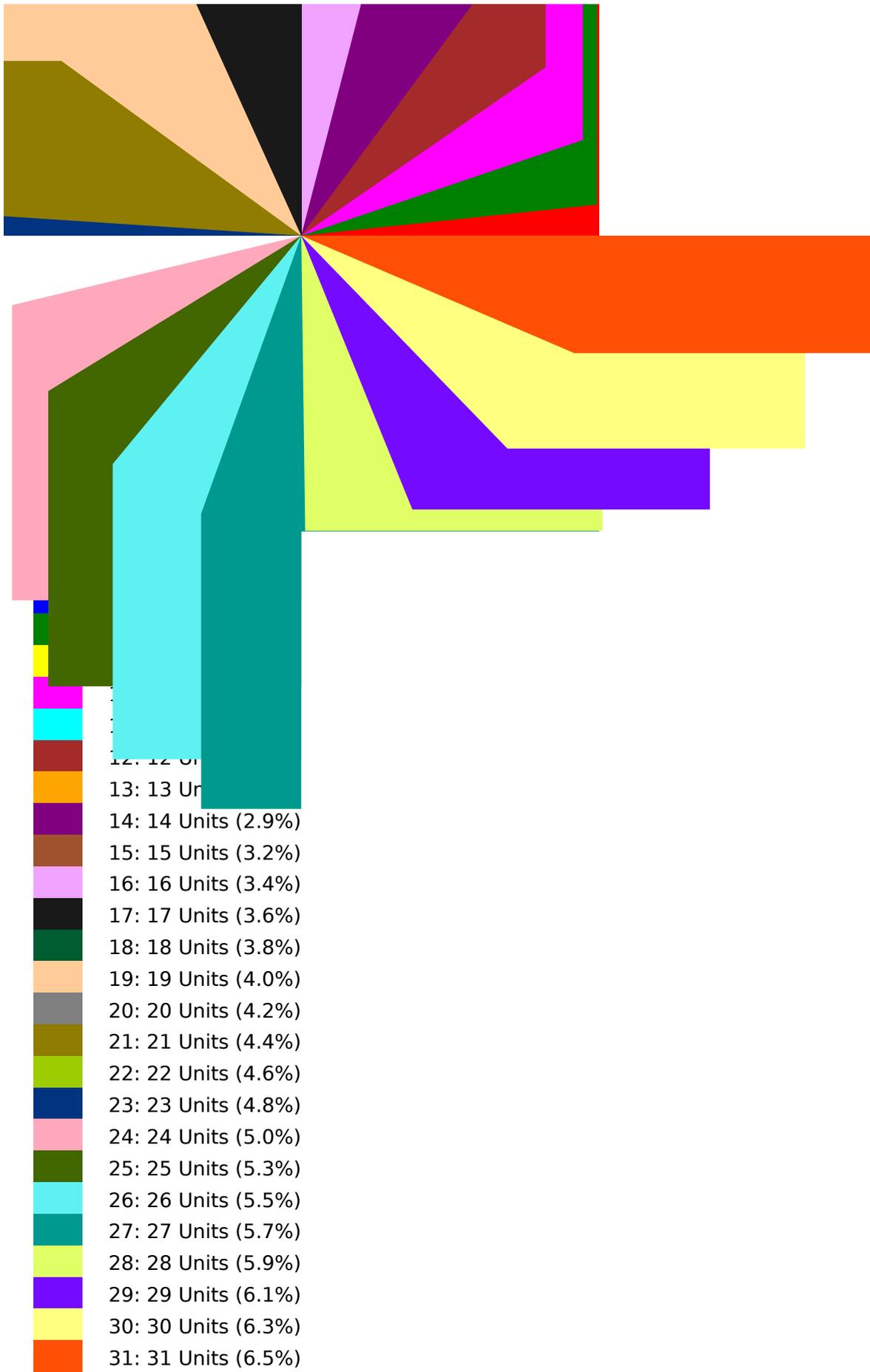
p
a
r
a
m
e
t
e
r
n
a
m
e

what it does

N Setting to true prevents displaying of the group legends under the chart. Any non-empty value means "yes". To say "no", simply do not supply this parameter at all, or leave the value blank.
o
w
r
a
p

Exam





```
--[[
    keywords are used for languages: they are the names of the actual
    parameters of the template
]]

local keywords = {
    barChart = 'bar chart',
    pieChart = 'pie chart',
    width = 'width',
    height = 'height',
    stack = 'stack',
    colors = 'colors',
    group = 'group',
    xlegend = 'x legends',
    yticks = 'y tick marks',
    tooltip = 'tooltip',
    accumulateTooltip = 'tooltip value accumulation',
    links = 'links',
    defcolor = 'default color',
    scalePerGroup = 'scale per group',
    unitsPrefix = 'units prefix',
    unitsSuffix = 'units suffix',
    groupNames = 'group names',
    hideGroupLegends = 'hide group legends',
    slices = 'slices',
    slice = 'slice',
    radius = 'radius',
    percent = 'percent',

} -- here is what you want to translate

local defColors = mw.loadData("Module:Chart/Default colors")
local hideGroupLegends

local function nulOrWhitespace( s )
    return not s or mw.text.trim( s ) == ''
end

local function createGroupList( tab, legends, cols )
    if #legends > 1 and not hideGroupLegends then
        table.insert( tab, mw.text.tag( 'div' ) )
        local list = {}
        local spanStyle = "padding:0 1em;background-color:%s;border:1px solid %s"
        for gi = 1, #legends do
            local span = mw.text.tag( 'span', { style = string.format( spanStyle, defColors[gi] ) } )
            table.insert( list, mw.text.tag( 'li', {}, span ) )
        end
        table.insert( tab,
            mw.text.tag( 'ul',
                {style="width:100%;list-style:none;column-width:100%;border:none"},
                table.concat( list, '\n' )
            )
        )
        table.insert( tab, '</div>' )
    end
end

local function pieChart( frame )
    local res, imslices, args = {}, {}, frame.args
    local radius
    local values, colors, names, legends, links = {}, {}, {}, {}, {}
    local delimiter = args.delimiter or ':'
    local lang = mw.getContentLanguage()
```

```
local function getArg( s, def, subst, with )
    local result = args[keywords[s]] or def or ''
    if subst and with then result = string.gsub( result, subst, with )
    return result
end

local function analyzeParams()
    local function addSlice( i, slice )
        local value, name, color, link = unpack( mw.text.split( slice, ' ' ) )
        values[i] = tonumber( lang:parseFormattedNumber( value ) )
            or error( string.format( 'Slice %d: "%s"', first, slice ) )
        colors[i] = not nulOrWhitespace( color ) and color or def
        names[i] = name or ''
        links[i] = link
    end

    radius = getArg( 'radius', 150 )
    hideGroupLegends = not nulOrWhitespace( args[keywords.hideGroupLegends] )
    local slicesStr = getArg( 'slices' )
    local prefix = getArg( 'unitsPrefix', '', '-', '' )
    local suffix = getArg( 'unitsSuffix', '', '-', '' )
    local percent = args[keywords.percent]
    local sum = 0
    local i = 0
    for slice in string.gmatch( slicesStr or '', "%b()" ) do
        i = i + 1
        addSlice( i, string.match( slice, '^((%S*(.-)%S*)$)' ) )
    end

    for k, v in pairs(args) do
        local ind = string.match( k, '^' .. keywords.slice .. '%S*' )
        if ind then addSlice( tonumber( ind ), v ) end
    end

    for _, val in ipairs( values ) do sum = sum + val end
    for i, value in ipairs( values ) do
        local addprec = percent and string.format( ' (%0.1f%%)', value )
        legends[i] = string.format( '%s: %s%s%s%s', names[i], prefix, value, suffix, addprec )
        links[i] = mw.text.trim( links[i] or string.format( '[[#%s|]]', names[i] ) )
    end
end

local function addRes( ... )
    for _, v in pairs( { ... } ) do
        table.insert( res, v )
    end
end

local function createImageMap()
    addRes( '{{#tag:imagemap|', 'Image:Circle frame.svg{|!}}' .. ( radius )
    addRes( unpack( imsllices ) )
    addRes( 'desc none', '}}' )
end

local function drawSlice( i, q, start )
    local color = colors[i]
    local angle = start * 2 * math.pi
    local sin, cos = math.abs( math.sin( angle ) ), math.abs( math.cos( angle ) )
    local wsin, wcos = sin * radius, cos * radius
    local s1, s2, w1, w2, w3, w4, border
    if q == 1 then
        border = 'left'
        w1, w2, w3, w4 = 0, 0, wsin, wcos
    end
end
```

```
        s1, s2 = 'bottom', 'left'
    elseif q == 2 then
        border = 'bottom'
        w1, w2, w3, w4 = 0, wcos, wsin, 0
        s1, s2 = 'bottom', 'right'
    elseif q == 3 then
        border = 'right'
        w1, w2, w3, w4 = wsin, wcos, 0, 0
        s1, s2 = 'top', 'right'
    else
        border = 'top'
        w1, w2, w3, w4 = wsin, 0, 0, wcos
        s1, s2 = 'top', 'left'
    end

    local style = string.format( 'border:solid transparent;position:
if start <= ( q - 1 ) * 0.25 then
        style = string.format( '%s;border:0;background-color:%s',
    else
        style = string.format( '%s;border-width:%spx %spx %spx %s',
    end
    addRes( mw.text.tag( 'div', { style = style }, ' ' ) )
end

local function createSlices()
    local function coordsOfAngle( angle )
        return ( 100 + math.floor( 100 * math.cos( angle ) ) ) .
    end

    local sum, start = 0, 0
    for _, value in ipairs( values ) do sum = sum + value end
    for i, value in ipairs(values) do
        local poly = { 'poly 100 100' }
        local startC, endC = start / sum, ( start + value ) / sum
        local startQ, endQ = math.floor( startC * 4 + 1 ), math.flo
        for q = startQ, math.min( endQ, 4 ) do drawSlice( i, q, s
        for angle = startC * 2 * math.pi, endC * 2 * math.pi, 0.0
            table.insert( poly, coordsOfAngle( angle ) )
        end
        table.insert( poly, coordsOfAngle( endC * 2 * math.pi )
        table.insert( imslices, table.concat( poly, ' ' ) )
        start = start + values[i]
    end
end

analyzeParams()
if #values == 0 then error( "no slices found - can't draw pie chart" ) end
addRes( mw.text.tag( 'div', { class = 'chart noresize', style = string.f
addRes( mw.text.tag( 'div', { style = string.format( 'position:relative;
createSlices()
addRes( mw.text.tag( 'div', { style = string.format( 'position:absolute;
createImageMap()
addRes( '</div>' ) -- close "position:relative" div that contains slices
addRes( '</div>' ) -- close "position:relative" div that contains slices
createGroupList( res, legends, colors ) -- legends
addRes( '</div>' ) -- close containing div
return frame:preprocess( table.concat( res, '\n' ) )
end

local function barChart( frame )
    local res = {}
    local args = frame.args -- can be changed to frame:getParent().args
    local values, xlegends, colors, tooltips, yscalses = {}, {}, {}, {}, {}
```

```
local groupNames, unitsSuffix, unitsPrefix, links = {}, {}, {}, {}
local width, height, yticks, stack, delimiter = 500, 350, -1, false, args
local chartWidth, chartHeight, defcolor, scalePerGroup, accumulateTooltip

local numGroups, numValues
local scaleWidth

local function validate()
    local function asGroups( name, tab, toDuplicate, emptyOK )
        if #tab == 0 and not emptyOK then
            error( "must supply values for " .. keywords[name] )
        end
        if #tab == 1 and toDuplicate then
            for i = 2, numGroups do tab[i] = tab[1] end
        end
        if #tab > 0 and #tab ~= numGroups then
            error ( keywords[name] .. ' must contain the same' )
        end
    end
end

-- do all sorts of validation here, so we can assume all params are valid
-- among other things, replace numerical values with mw.language

chartHeight = height - 80
numGroups = #values
numValues = #values[1]
defcolor = defcolor or 'blue'
colors[1] = colors[1] or defcolor
scaleWidth = scalePerGroup and 80 * numGroups or 100
chartWidth = width - scaleWidth
asGroups( 'unitsPrefix', unitsPrefix, true, true )
asGroups( 'unitsSuffix', unitsSuffix, true, true )
asGroups( 'colors', colors, true, true )
asGroups( 'groupNames', groupNames, false, false )
if stack and scalePerGroup then
    error( string.format( 'Illegal settings: %s and %s are in' ) )
end
for gi = 2, numGroups do
    if #values[gi] ~= numValues then error( keywords.group .
end
if #xlegends ~= numValues then error( 'Illegal number of ' .. key
end

local function extractParams()
    local function testone( keyword, key, val, tab )
        local i = keyword == key and 0 or key:match( keyword ..
        if not i then return end
        i = tonumber( i ) or error("Expect numerical index for k
        if i > 0 then tab[i] = {} end
        for s in mw.text.gsplit( val, '%s*' .. delimiter .. '%s*'
            table.insert( i == 0 and tab or tab[i], s )
        end
        return true
    end
end

for k, v in pairs( args ) do
    if k == keywords.width then
        width = tonumber( v )
        if not width or width < 200 then
            error( 'Illegal width value (must be a nu
        end
    elseif k == keywords.height then
```



```
        height = tonumber( v )
        if not height or height < 200 then
            error( 'Illegal height value (must be a r
        end
elseif k == keywords.stack then stack = true
elseif k == keywords.yticks then yticks = tonumber(v) or
elseif k == keywords.scalePerGroup then scalePerGroup = t
elseif k == keywords.defcolor then defcolor = v
elseif k == keywords.accumulateTooltip then accumulateTo
elseif k == keywords.hideGroupLegends then hideGroupLeger
else
    for keyword, tab in pairs( {
        group = values,
        xlegend = xlegends,
        colors = colors,
        tooltip = tooltips,
        unitsPrefix = unitsPrefix,
        unitsSuffix = unitsSuffix,
        groupNames = groupNames,
        links = links,
    } ) do
        if testone( keywords[keyword], k,
            then break
        end
    end
end
end
end
end

local function roundup( x ) -- returns the next round number: eg., for 30
    local ordermag = 10 ^ math.floor( math.log10( x ) )
    local normalized = x / ordermag
    local top = normalized >= 1.5 and ( math.floor( normalized + 1 )
    return ordermag * top, top, ordermag
end

local function calcHeightLimits() -- if limits were passed by user, use t
    if stack then
        local sums = {}
        for _, group in pairs( values ) do
            for i, val in ipairs( group ) do sums[i] = ( sums
            end
        local sum = math.max( unpack( sums ) )
        for i = 1, #values do yscales[i] = sum end
    else
        for i, group in ipairs( values ) do yscales[i] = math.max
        end
        for i, scale in ipairs( yscales ) do yscales[i] = roundup( scale
        if not scalePerGroup then for i = 1, #values do yscales[i] = math
    end

local function tooltip( gi, i, val )
    if tooltips and tooltips[gi] and not nulOrWhitespace( tooltips[gi]
    local groupName = mw.text.killMarkers(not nulOrWhitespace( groupN
    local prefix = unitsPrefix[gi] or unitsPrefix[1] or ''
    local suffix = unitsSuffix[gi] or unitsSuffix[1] or ''
    return string.gsub(groupName .. prefix .. mw.getContentLanguage()
end

local function calcHeights( gi, i, val )
    local barHeight = math.floor( val / yscales[gi] * chartHeight + 0
    local top, base = chartHeight - barHeight, 0
    if stack then
        local rawbase = 0
```

```
        for j = 1, gi - 1 do rawbase = rawbase + values[j][i] end
        base = math.floor( chartHeight * rawbase / yscales[gi] )
    end
    if barHeight < 2 then
        barHeight = 2 -- Otherwise the template would try to create
    end
    return barHeight, top - base
end

local function groupBounds( i )
    local setWidth = math.floor( chartWidth / numValues )
    local setOffset = ( i - 1 ) * setWidth
    return setOffset, setWidth
end

local function calcx( gi, i )
    local setOffset, setWidth = groupBounds( i )
    if stack or numGroups == 1 then
        local barWidth = math.min( 38, math.floor( 0.8 * setWidth ) )
        return setOffset + (setWidth - barWidth) / 2, barWidth
    end
    setWidth = 0.85 * setWidth
    local barWidth = math.floor( 0.75 * setWidth / numGroups )
    local left = setOffset + math.floor( ( gi - 1 ) / numGroups * setWidth )
    return left, barWidth
end

local function drawbar( gi, i, val, ttval )
    if val == '0' then return end -- do not show single line (border)

    local color, tooltip, custom = colors[gi] or defcolor or 'blue',
    local left, barWidth = calcx( gi, i )
    local barHeight, top = calcHeights( gi, i, val )

    -- borders so it shows up when printing
    local style = string.format("position:absolute;left:%spx;top:%spx;
                                width:%spx;height:%spx;
                                left, top, barHeight-1, barWidth)
    local link = links[gi] and links[gi][i] or ''
    local img = not nulOrWhitespace( link ) and string.format( '[[File:
table.insert( res, mw.text.tag( 'div', { style = style, title = ttval } )
end

local function drawYScale()
    local function drawSingle( gi, color, width, yticks, single )
        local yscale = yscales[gi]
        local _, top, ordermag = roundup( yscale * 0.999 )
        local numnotches = yticks >= 0 and yticks or
            (top <= 1.5 and top * 4
             or top < 4 and top * 2
             or top)
        local valStyleStr =
            single and 'position:absolute;height=20px;text-align:center'
            or 'position:absolute;height=20px;text-align:right'
        local notchStyleStr = 'position:absolute;height=1px;min-width:100%'
        for i = 1, numnotches do
            local val = i / numnotches * yscale
            local y = chartHeight - calcHeights( gi, 1, val )
            local div = mw.text.tag( 'div', { style = string.format(
table.insert( res, div )
            div = mw.text.tag( 'div', { style = string.format(
table.insert( res, div )
        end
    end
```

```
        if scalePerGroup then
            local colWidth = 80
            local colStyle = "position:absolute;height:%spx;min-width:"
            for gi = 1, numGroups do
                local left = ( gi - 1 ) * colWidth
                local color = colors[gi] or defcolor
                table.insert( res, mw.text.tag( 'div', { style =
                    drawSingle( gi, color, colWidth, yticks )
                    table.insert( res, '</div>' )
                end
            else
                drawSingle( 1, 'black', scaleWidth, yticks, true )
            end
        end
    end

    local function drawXlegends()
        local setOffset, setWidth
        local legendDivStyleFormat = "position:absolute;left:%spx;top:10"
        local tickDivstyleFormat = "position:absolute;left:%spx;height:10"
        for i = 1, numValues do
            if not nulOrWhitespace( xlegends[i] ) then
                setOffset, setWidth = groupBounds( i )
                -- setWidth = 0.85 * setWidth
                table.insert( res, mw.text.tag( 'div', { style =
                    table.insert( res, mw.text.tag( 'div', { style =
                end
            end
        end

    end

    local function drawChart()
        table.insert( res, mw.text.tag( 'div', { class = 'chart noresize'
        table.insert( res, mw.text.tag( 'div', { style = string.format("f

        table.insert( res, mw.text.tag( 'div', { style = string.format("f
        local acum = stack and accumulateTooltip and {}
        for gi, group in pairs( values ) do
            for i, val in ipairs( group ) do
                if acum then acum[i] = ( acum[i] or 0 ) + val end
                drawbar( gi, i, val, acum and acum[i] )
            end
        end
        table.insert( res, '</div>' )
        table.insert( res, mw.text.tag( 'div', { style = string.format("f
        drawYScale()
        table.insert( res, '</div>' )
        table.insert( res, mw.text.tag( 'div', { style = string.format( '
        drawXlegends()
        table.insert( res, '</div>' )
        table.insert( res, '</div>' )
        createGroupList( res, groupNames, colors )
        table.insert( res, '</div>' )

    end

    extractParams()
    validate()
    calcHeightLimits()
    drawChart()
    return table.concat( res, "\n" )

end
```



```
return {  
    ['bar-chart'] = barChart,  
    [keywords.barChart] = barChart,  
    [keywords.pieChart] = pieChart,  
}
```

Vorlage:Graph:Chart

Vorlage:Lua Vorlage:Uses TemplateStyles

Parameters

chart

Creates a JSON object for <graph> to display charts. In the article namespace the template **Vorlage:TI** should be used instead. See its page for use cases.

Parameters:

- **width:** width of the chart
- **height:** height of the chart
- **type:** type of the chart: line for **line charts**, area for **area charts**, and rect for (column) **bar charts**, and pie for **pie charts**. Multiple series can stacked using the stacked prefix, e.g. stackedarea.
- **interpolate:** **interpolation** method for line and area charts. It is recommended to use monotone for a **monotone cubic interpolation** – further supported values are listed at <https://github.com/vega/vega/wiki/Marks#area>.
- **colors:** color palette of the chart as a comma-separated list of colors. The color values must be given either as #rgb/#rrggbb/#aarrggbb or by a **CSS color name**. For #aarrggbb the aa component denotes the **alpha channel**, i.e. FF=100% opacity, 80=50% opacity/transparency, etc. (The default color palette is **category10**).
- **xAxisTitle** and **yAxisTitle:** captions of the x and y axes
- **xAxisMin**, **xAxisMax**, **yAxisMin**, and **yAxisMax:** minimum and maximum values of the x and y axes
- **xAxisFormat** and **yAxisFormat:** changes the formatting of the axis labels. Supported values are listed at <https://github.com/d3/d3-3.x-api-reference/blob/master/Formatting.md#numbers> for numbers and <https://github.com/d3/d3-3.x-api-reference/blob/master/Time-Formatting.md> for date/time. For example, the format % can be used to output percentages.
- **xAxisAngle:** rotates the x axis labels by the specified angle. Recommended values are: -45, +45, -90, +90
- **xType** and **yType:** Data types of the values, e.g. integer for integers, number for real numbers, date for dates (e.g. YYYY/MM/DD), and string for ordinal values.
- **x:** the x-values as a comma-separated list (if a value itself contains a comma it must be escaped with a backslash, i.e. it needs to be written as \,)
- **y** or **y1**, **y2**, ...: the y-values for one or several data series, respectively. For pie charts y2 denotes the radiuses of the corresponding sectors.
- **legend:** show legend (only works in case of multiple data series)
- **y1Title**, **y2Title**, ...: defines the label of the respective data series in the legend
- **linewidth:** line width for line charts or distance between the pie segments for pie charts



- **showValues:** Additionally, output the y values as text. (Currently, only (non-stacked) bar and pie charts are supported.) The output can be configured used the following parameters provided as `name1:value1, name2:value2`:
 - **format:** Format the output according to <https://github.com/d3/d3-3.x-api-reference/blob/master/Formatting.md#numbers> for numbers and <https://github.com/d3/d3-3.x-api-reference/blob/master/Time-Formatting.md> for date/time.
 - **fontcolor:** text color
 - **fontsize:** text size
 - **offset:** move text by the given offset. For bar charts and pie charts with `midangle` this also defines if the text is inside or outside the chart.
 - **angle** (pie charts only): text angle in degrees or `midangle` (default) for dynamic angles based on the mid-angle of the pie sector.
- **innerRadius:** For pie charts: defines the inner radius to create a *doughnut chart*.
- **formatjson:** format JSON object for better legibility

Examples

Line Chart:

```
{{Graph:Chart|width=400|height=100|type=line|x=1,2,3,4,5,6,7,8|y=10, 12, 6, 14, 2, 10, 7, 9}}
```

Note: The y-axis starts from the smallest y value, though this can be overridden with the `yAxisMin` parameter.

Area chart:

```
{{Graph:Chart|width=400|height=100|type=area|x=1,2,3,4,5,6,7,8|y=10, 12, 6, 14, 2, 10, 7, 9}}
```

Note: The y-axis starts from zero

Bar chart:



```
{{Graph:Chart|width=400|height=100|xAxisTitle=X|yAxisTitle=Y|type=rect|x=1,2,3,4,5,6,7,8|y=10, 12, 6, 14, 2, 10, 7, 9}}
```

Line chart with more than one data series, using colors:

```
{{Graph:Chart|width=400|height=100|xAxisTitle=X|yAxisTitle=Y|legend=Legend|type=line|x=1,2,3,4,5,6,7,8|y1=10, 12, 6, 14, 2, 10, 7, 9 |y2=2,4,6,8,13,11,9,2|colors=#0000aa,#ff8000}}
```

Area chart with more than one data series showing blended overlap:

```
{{Graph:Chart|width=400|height=100|xAxisTitle=X|yAxisTitle=Y|legend=Legend|type=area|x=1,2,3,4,5,6,7,8|y1=10, 12, 6, 14, 2, 10, 7, 9 |y2=2,4,6,8,13,11,9,2|colors=#800000aa,#80ff8000}}
```

Bar chart with multiple data series:

```
{{Graph:Chart|width=400|height=100|xAxisTitle=X|yAxisTitle=Y|legend=Legend|type=rect|x=1,2,3,4,5,6,7,8|y1=10, 12, 6, 14, 2, 10, 7, 9 |y2=2,4,6,8,13,11,9,2|colors=#800000aa,#80ff8000}}
```

Area chart with smoothed data values:



```
{{Graph:  
Chart|width=400|height=100|xAxisTitle=X|yAxisTitle=Y|legend=Legend|type=stackedarea:  
2,3,4,5,6,7,8|y1=10, 12, 6, 14, 2, 10, 7, 9 |y2=2,4,6,8,13,11,9,2  
|interpolate=monotone|colors=seagreen, orchid}}
```

Bar chart with stacked data series:

```
{{Graph:  
Chart|width=400|height=100|xAxisTitle=X|yAxisTitle=Y|legend=Legend|type=stackedrect  
2,3,4,5,6,7,8|y1=10, 12, 6, 14, 2, 10, 7, 9 |y2=2,4,6,8,13,11,9,2|y1Title=Data  
A|y2Title=Data B|colors=seagreen, orchid}}
```

Note that the order of the stacking is governed by the alphabetic value of the y-Titles used for the legend. You can always prepend a number 1, 2, 3 to establish whatever stacking order you want.