Inhaltsverzeichnis

- 1. Modul:RoundN/Doku
- 2. Modul:RoundN

Modul:RoundN/Doku

Dies ist die Dokumentationsseite für Modul:RoundN

Inhaltsverzeichnis

- 1 Usage
- 2 Parameter List
  - 2.1 Example 1
  - 2.2 Example 2
- 3 Node Functions
- 4 Undocumented features
  - 4.1 Partial List of undocumented params

Usage

This is a module for meta templates such as Vorlage:Tl, Vorlage:Tl, etc. Though originally designed for sports, it can be used to present any tree.

Basic form:

```
{{#invoke:RoundN|main|columns = [number of columns (3 columns for Round8, 4 columns for Rour
```

Alternatively, the shorthand form `{{#invoke:RoundN|N16}}` (or any power of 2 between N2 and N512) is equivalent to Vorlage:Code

Parameter List

- Vorlage:Red parameter names are required.
- *italicized* parameter names are required in some cases
- Vorlage:Blue parameter names involve new features not available prior to the release of this module

Parameter name	Description
Vorlage:TOC tab	Number of columns/eliminations (3 columns for Round8, 4 columns for Round16, etc.) **Note** • For `columns` *less than 4* (i.e. Round2-Round8), the 3rd Place match box is hidden by default. For `columns` *greater or equal to 4*, the 3rd Place match box is visible by default. This reflects the behavior of the templates prior to this module's release. Vorlage:TOC tab While required for invoking the main function (`{{#invoke:RoundN|main|columns=}}`), this module has meta functions in the form of 'N##' which can call main with the right Vorlage:Code for you. For example, you may

Parameter name	Description
	replace `... |main|columns = 7...` with `...|N128 ....` Valid for N# where # is a power of 2. You must invoke this module directly to use this (vs. a template that already has a columns value). Only works up to N512. If say N1024 ever becomes necessary, call main directly and set Vorlage:Para. PS: Though adding shorthands up to N ~70 trillion would be easy, it would make unnecessary overhead as these meta functions are generated eached time this module is called (granted generating such functions up to ~70 trillion is probably less expensive than making the table for columns = 3).
Vorlage:TOC tab	Unnamed parameters (i.e. a value not prefixed by `[param_name] =`) are read sequentially in groups of 5 such that: **Example 1** Vorlage:Demo Placing each group of 5 on a new line is optional, but does make it easier to read. Also, consider adding the comments such as `<!-- Date-Place/Team 1 /Score 1/Team 2/Score 2 -->` on top and `<!-- Finals -->` above the first group of 5 in the finals round, etc.
Vorlage:TOC tab	Set the `style` parameter to add custom CSS to the table. Modul Diskussion:RoundN/testcases/1
Vorlage:TOC tab	For large tables, set Vorlage:Code to the desired height in pixels. Modul Diskussion:RoundN/testcases/1 CSS units are also allowed (i.e. '20em', '30%', etc.). **Note** This works by duplicating the entire table and then using CSS to lock the clone of the table to the top of the div. Conceivably, for extremely large tables, this can result in a significant amount of extra HTML code to download versus if Vorlage:Code is used.
Vorlage:TOC tab	If it is desirable to have the round heading scroll with the table (such as if a larger viewing area is desired), set Vorlage:Code to 'yes' Modul Diskussion:RoundN/testcases/1
Vorlage:TOC tab	If set, boxes for the #th match group will not be shown. Most often used for playoffs or when the number of teams playing in the first round is not a power of 2.

Parameter name	Description

	Modul Diskussion:RoundN/testcases/1 Vorlage:TOC tab: The shorthand `|skipmatch = 1-2;4;6-7` will do the same thing as:
	
	`|skipmatch1=yes`
	`|skipmatch2=yes`
	`|skipmatch4=yes`
	`|skipmatch6=yes`
	`|skipmatch7=yes`
	
	Note:
	
	• Unlike in the original templates, this module does not require leading zeroes in this parameter, i.e. skipmatch001 is the same as skipmatch1 (though it may make your template code easier to read if lead with an appropriate number of zeroes).
	• The 5 parameters that would have populated the skipped box will be ignored regardless of value unless o `mit_blanks` is set to 'yes' (see below).
	• Previously, skipmatch only worked in the first round. This limitation no longer applies. (See Module talk: RoundN/testcases/3)

Vorlage:TOC tab	If `omit_blanks` is set to yes, then all parameters that would have been skipped will instead be shifted to the next non-skipped box. (This is turned off by default because most templates made before the release of this module were required to use empty parameters as placeholders.) Modul Diskussion:RoundN/testcases/1
Vorlage:TOC tab	The Vorlage:Code parameter accepts either Vorlage:Code or V orlage:Code, which will automatically bold the text of the participant with the *higher* or *lower* score, respectively. In other words, set this to 'low' if the lower score wins and 'high' if the high score wins. **Example 2** Modul Diskussion:RoundN/testcases/2 **Note:** • If entering a score that includes non-numbersVorlage:-- such as 3 (6), the Semi Final score for team C in the above exampleVorlage:--the module will first remove all non-digit characters and concatenate the rest. For example, 3 (5) and 3 (6) would be converted to 35 and 36, respectively, before being compared. This should be valid for most cases, however, you may override using the `manualboldmatch##` parameter.

Parameter name	Description
	• If the scores are tied or contain no numbers, then neither will be bolded, however, you may still manually bold them with wikimarkup. • This does not remove any formatting already present.
Vorlage:TOC tab	Modul Diskussion:RoundN/testcases/2
	Vorlage:TOC tab The form `manualboldmatch = 1-3;6;9-12` is also available.
Vorlage:TOC tab	Set `|previewnumbers = yes` to show numbers next to each match group (useful for Vorlage:Code and Vorlage:Code) when viewing on the template page. Note that these numbers will not appear in article space.
Vorlage:TOC tab	Use RD#, replacing # with the desired column such that 1 is the leftmost round and X is the rightmost when `columns = X`. For example: Vorlage:Demo **Note** RD[N+1] = Third Place, and will perform the job of the `Consol` parameter if the latter is omitted, i.e. RD[N+1] is ignored if Consol is true. Also, this alternate name for Consol was not available prior to the release of this module (and is provided because the module's programmer thought 'Consol' was unintuitive).
Vorlage:TOC tab	By default, `|3rdplace=` is set to 'yes' when `columns` is greater than 3 and 'no' otherwise. Override as desired. (See `Consol` if you wish to rename this heading) Vorlage:Demo
Vorlage:TOC tab	Set `Consol=name` to change the 'Third Place' label to 'name'. You may also use the form RD# where # = columns + 1. Vorlage:Demo
Vorlage:TOC tab	Add `|color=yes` Final 7 Gold medalist 3 Silver medalist 2 Third place 8

Parameter name	Description
	Bronze medalist 1 0
Vorlage:TOC tab	Set `team-width` to the desired width in pixels. (Default is 170)
Vorlage:TOC tab	Set `score-width` to the desired width in pixels. (Default is 30)
Vorlage:TOC tab	Setting `|widescore=yes` is basically equal to `|score-width=40`. Provided for compatibility. Ignored if `score-width` is set.
Vorlage:TOC tab	Set `score-boxes` to the desired number of score boxes per match. (Default is 1). The number can be followed by `+ sum`, which will add one more score box with the sum of all the others. For examples, see test case 5 and test case 6.
Vorlage:TOC tab	Set `|template=yes` if used to create a template for a specific game (add V.T.E. link using Vorlage:Tl).
Vorlage:TOC tab	Set `|flex_tree=yes` to make the brackets more compact. That is, to have less space between matches of the same round.

Node Functions

Vorlage:Ombox

For greater customization, node functions may be added to the parameter usually used for providing match information such as dates and location. Simply insert `node_function{`*`function name`*`}` in the appropriate location. A demonstration of most of the node functions can be found at RoundN/testcases/4.

- **orphan** - Current node is disconnected from all other nodes (no lines will be drawn). One practical use for this may be when the winners of the previous round advance to a different tournament instead of playing each other but the losers still play a consolation match for 3rd place, as per this discussion.

For the next three node functions, the parameters usually used for team names and scores should be omitted if these node functions are used.

- **line** - Current node is omitted and replaced with a horizontal line. Text may be displayed above this line via `node_function{line(`*`text`*`)}` (at some point, the preferred format would be `node_function{line}`*`text`* to match the form of the canvas function though backwards compatibility would probably be maintained). If curly brackets are desired in the text, make sure to escape them with `\`.
- **bridge** - Current node is omitted and replaced with a vertical line.
- **canvas** - Current node is replaced with whatever you want via `node_function{canvas}`*`anything`*
- **heading** - Not yet implemented. Planned node function with the form `node_function{heading(`*`name`*`)}`*`date/location`*. Makes a round name heading like for the Consol parameter, but may be placed anywhere, as per the "Elimination Rounds" of RoundN/testcases/6 (which currently renders the planned output of this function without the functions).

Undocumented features

Alas User:Codehydro seriously overbuilt this module and never found the time to document even half the features available. Check out the following examples which may contain some advanced features: Spezial:PrefixIndex/Module talk:RoundN/testcases

Partial List of undocumented params

- no_column_head
- short_brackets
- branch_upwards

Note that some of these features may not have been documented due to incomplete implementation.

Modul:RoundN

Inhaltsverzeichnis

- 1 Usage
- 2 Parameter List
  - 2.1 Example 1
  - 2.2 Example 2
- 3 Node Functions
- 4 Undocumented features
  - 4.1 Partial List of undocumented params

Usage

This is a module for meta templates such as Vorlage:Tl, Vorlage:Tl, etc. Though originally designed for sports, it can be used to present any tree.

Basic form:

```
{{#invoke:RoundN|main|columns = [number of columns (3 columns for Round8, 4 columns for Rour
```

Alternatively, the shorthand form `{{#invoke:RoundN|N16}}` (or any power of 2 between N2 and N512) is equivalent to Vorlage:Code

Parameter List

- Vorlage:Red parameter names are required.
- *italicized* parameter names are required in some cases
- Vorlage:Blue parameter names involve new features not available prior to the release of this module

Parameter name	Description
Vorlage:TOC tab	Number of columns/eliminations (3 columns for Round8, 4 columns for Round16, etc.) **Note** • For `columns` *less than 4* (i.e. Round2-Round8), the 3rd Place match box is hidden by default. For `columns` *greater or equal to 4*, the 3rd Place match box is visible by default. This reflects the behavior of the templates prior to this module's release. Vorlage:TOC tab While required for invoking the main function (`{{#invoke:RoundN|main|columns=}}`), this module has meta functions in the form of 'N##' which can call main

Parameter name	Description
	with the right Vorlage:Code for you. For example, you may replace `... |main|columns = 7...` with `...|N128 ...`. Valid for N# where # is a power of 2. You must invoke this module directly to use this (vs. a template that already has a columns value). Only works up to N512. If say N1024 ever becomes necessary, call main directly and set Vorlage:Para. PS: Though adding shorthands up to N ~70 trillion would be easy, it would make unnecessary overhead as these meta functions are generated eached time this module is called (granted generating such functions up to ~70 trillion is probably less expensive than making the table for columns = 3).
Vorlage:TOC tab	Unnamed parameters (i.e. a value not prefixed by `[param_name] =`) are read sequentially in groups of 5 such that: **Example 1** Vorlage:Demo Placing each group of 5 on a new line is optional, but does make it easier to read. Also, consider adding the comments such as `<!-- Date-Place/Team 1 /Score 1/Team 2/Score 2 -->` on top and `<!-- Finals -->` above the first group of 5 in the finals round, etc.
Vorlage:TOC tab	Set the `style` parameter to add custom CSS to the table. Modul Diskussion:RoundN/testcases/1
Vorlage:TOC tab	For large tables, set Vorlage:Code to the desired height in pixels. Modul Diskussion:RoundN/testcases/1 CSS units are also allowed (i.e. '20em', '30%', etc.). **Note** This works by duplicating the entire table and then using CSS to lock the clone of the table to the top of the div. Conceivably, for extremely large tables, this can result in a significant amount of extra HTML code to download versus if Vorlage:Code is used.
Vorlage:TOC tab	If it is desirable to have the round heading scroll with the table (such as if a larger viewing area is desired), set Vorlage:Code to 'yes' Modul Diskussion:RoundN/testcases/1
Vorlage:TOC tab	If set, boxes for the #th match group will not be shown. Most often used for playoffs or when the number of teams playing in the first round is not a power of 2.

Parameter name	Description

	Modul Diskussion:RoundN/testcases/1 Vorlage:TOC tab: The shorthand `|skipmatch = 1-2;4;6-7` will do the same thing as:
	
	`|skipmatch1=yes`
	`|skipmatch2=yes`
	`|skipmatch4=yes`
	`|skipmatch6=yes`
	`|skipmatch7=yes`
	
	Note:
	
	• Unlike in the original templates, this module does not require leading zeroes in this parameter, i.e. skipmatch001 is the same as skipmatch1 (though it may make your template code easier to read if lead with an appropriate number of zeroes).
	• The 5 parameters that would have populated the skipped box will be ignored regardless of value unless o `mit_blanks` is set to 'yes' (see below).
	• Previously, skipmatch only worked in the first round. This limitation no longer applies. (See Module talk: RoundN/testcases/3)

Vorlage:TOC tab	If `omit_blanks` is set to yes, then all parameters that would have been skipped will instead be shifted to the next non-skipped box. (This is turned off by default because most templates made before the release of this module were required to use empty parameters as placeholders.) Modul Diskussion:RoundN/testcases/1
Vorlage:TOC tab	The Vorlage:Code parameter accepts either Vorlage:Code or V orlage:Code, which will automatically bold the text of the participant with the *higher* or *lower* score, respectively. In other words, set this to 'low' if the lower score wins and 'high' if the high score wins. **Example 2** Modul Diskussion:RoundN/testcases/2 **Note:** • If entering a score that includes non-numbersVorlage:-- such as 3 (6), the Semi Final score for team C in the above exampleVorlage:--the module will first remove all non-digit characters and concatenate the rest. For example, 3 (5) and 3 (6) would be converted to 35 and 36, respectively, before being compared. This should be valid for most cases, however, you may override using the `manualboldmatch##` parameter.

Parameter name	Description
	• If the scores are tied or contain no numbers, then neither will be bolded, however, you may still manually bold them with wikimarkup. • This does not remove any formatting already present.
Vorlage:TOC tab	Modul Diskussion:RoundN/testcases/2
	Vorlage:TOC tab The form `manualboldmatch = 1-3;6;9-12` is also available.
Vorlage:TOC tab	Set `|previewnumbers = yes` to show numbers next to each match group (useful for Vorlage:Code and Vorlage:Code) when viewing on the template page. Note that these numbers will not appear in article space.
Vorlage:TOC tab	Use RD#, replacing # with the desired column such that 1 is the leftmost round and X is the rightmost when `columns = X`. For example: Vorlage:Demo **Note** RD[N+1] = Third Place, and will perform the job of the `Consol` parameter if the latter is omitted, i.e. RD[N+1] is ignored if Consol is true. Also, this alternate name for Consol was not available prior to the release of this module (and is provided because the module's programmer thought 'Consol' was unintuitive).
Vorlage:TOC tab	By default, `|3rdplace=` is set to 'yes' when `columns` is greater than 3 and 'no' otherwise. Override as desired. (See `Consol` if you wish to rename this heading) Vorlage:Demo
Vorlage:TOC tab	Set `Consol=name` to change the 'Third Place' label to 'name'. You may also use the form RD# where # = columns + 1. Vorlage:Demo
Vorlage:TOC tab	Add `|color=yes` Final 7 Gold medalist 3 Silver medalist 2 Third place 8

Parameter name	Description
	Bronze medalist 1 0
Vorlage:TOC tab	Set `team-width` to the desired width in pixels. (Default is 170)
Vorlage:TOC tab	Set `score-width` to the desired width in pixels. (Default is 30)
Vorlage:TOC tab	Setting `|widescore=yes` is basically equal to `|score-width=40`. Provided for compatibility. Ignored if `score-width` is set.
Vorlage:TOC tab	Set `score-boxes` to the desired number of score boxes per match. (Default is 1). The number can be followed by `+ sum`, which will add one more score box with the sum of all the others. For examples, see test case 5 and test case 6.
Vorlage:TOC tab	Set `|template=yes` if used to create a template for a specific game (add V.T.E. link using Vorlage:Tl).
Vorlage:TOC tab	Set `|flex_tree=yes` to make the brackets more compact. That is, to have less space between matches of the same round.

Node Functions

Vorlage:Ombox

For greater customization, node functions may be added to the parameter usually used for providing match information such as dates and location. Simply insert `node_function{`*`function name`*`}` in the appropriate location. A demonstration of most of the node functions can be found at RoundN/testcases/4.

- **orphan** - Current node is disconnected from all other nodes (no lines will be drawn). One practical use for this may be when the winners of the previous round advance to a different tournament instead of playing each other but the losers still play a consolation match for 3rd place, as per this discussion.

For the next three node functions, the parameters usually used for team names and scores should be omitted if these node functions are used.

- **line** - Current node is omitted and replaced with a horizontal line. Text may be displayed above this line via `node_function{line(`*`text`*`)}` (at some point, the preferred format would be `node_function{line}`*`text`* to match the form of the canvas function though backwards compatibility would probably be maintained). If curly brackets are desired in the text, make sure to escape them with `\`.
- **bridge** - Current node is omitted and replaced with a vertical line.
- **canvas** - Current node is replaced with whatever you want via `node_function{canvas}`*`anything`*
- **heading** - Not yet implemented. Planned node function with the form `node_function{heading(`*`name`*`)}`*`date/location`*. Makes a round name heading like for the Consol parameter, but may be placed anywhere, as per the "Elimination Rounds" of RoundN/testcases/6 (which currently renders the planned output of this function without the functions).

Undocumented features

Alas User:Codehydro seriously overbuilt this module and never found the time to document even half the features available. Check out the following examples which may contain some advanced features: Spezial:PrefixIndex/Module talk:RoundN/testcases

Partial List of undocumented params

- no_column_head
- short_brackets
- branch_upwards

Note that some of these features may not have been documented due to incomplete implementation.

```
local p = {
        RD = {
                'Quarter-finals',
                'Semi-finals',
                'Final',
                'Third place'
        },
        bgColor = {head = '#f2f2f2', 'gold', 'silver', '#C96', '#f9f9f9'},
        reuseStr = {},
        saveStr = function(self, name, ...)
                if not self.reuseStr[name] then
                        self.reuseStr[name] = table.concat{...}
                end
                return self.reuseStr[name]
        end
}

--Provides a convenient naming shortcut up to {{#invoke:RoundN|N512}} = {{invoke:RoundN|mair
for columns = 1, 9 do
        local N = math.pow(2, columns)
        p['N' .. N] = function(frame)
                return p.main(frame.args, columns)
        end
        p['n' .. N] = p['N' .. N]--to make case insensitive
end

--saves memory and avoids errors when using a nil as a table by providing a temporary table;
p.nilAsTab = {
        __index = function(t, i)
                return setmetatable({}, setmetatable(p.nilAsTab, {__index = {t = t, i = i}})
        end,
        __newindex = function (pt, pi, v) --store new values in actual table rather than tem
                rawset(p.nilAsTab.t, p.nilAsTab.i, {})[p.nilAsTab.i][pi] = v
                setmetatable(p.nilAsTab.t[p.nilAsTab.i], {__call = p.nilAsTab.__call})
        end,
        __call = function(t, i)
                return t and rawget(t, i)
        end
```

```
}
--never assign a value to these or they will stop being empty
local infiniteEmpty = setmetatable({}, {__index = setmetatable({}, p.nilAsTab), p.nilAsTab})
local callableEmpty = setmetatable({}, p.nilAsTab)

local rowNum, head, m, col, tab, esc = {}, {}, {num = 1, phase = 0, bold = infiniteEmpty}, {
        bs = require'Module:Escape',--backslash
        comma = {['(%([^,]*),([^%)]*%))'] = '%1|@!#|%2'},--escape commas in ()
}
local nodeFunc = {
        scanPattern = function(self, args, step)
                self.pattern = nil
                if args[step] then
                        self.pattern, self.nonFunc = string.match(esc.bs:text(args[step]), '
                end
                if self.pattern then
                        for k, v in pairs(esc.comma) do
                                self.pattern = self.pattern:gsub(k, v)
                        end
                        self.nonFunc = self.nonFunc and esc.bs:undo(self.nonFunc)
                        self.pattern = mw.text.split(self.pattern, '%s*,%s*')
                        for k, v in ipairs(self.pattern) do
                                local func, arg = string.match(v, '^(%w+)%(?([^%)]*)')
                                if func and self[func] and self[func].main then
                                        self.pattern[k] = func
                                        if arg then
                                                for x, y in pairs(esc.comma) do
                                                        arg = esc.bs:undo(arg):gsub(y:gsub('
                                                end
                                                self[func].arg = self[func].arg or {}
                                                self[func].arg[m.num] = arg
                                        end
                                end
                        end
                end
                return self.pattern
        end,
        helper = {
                topBranch = function()--node is top of fork if top is 0
                        return (m.num - col.top) % 2
                end,
                addText = function(text)
                        if text and text ~= '' then
                                tab.r:wikitext(text)
                        end
                end
        },
        line = {--this node is omitted and replaced with a line
                main = function(x)
                        local h = p.getNodeFunc()
                        if m.available then
                                local text, topId, isTop, notTop = h.line.arg[m.num] or '',
                                isTop = topId == 0
                                notTop = {[isTop and 1 or 0] = p.reuseStr.solid}
                                for k = 0, 1 do
                                        tab.r = rowNum[m.r + k * 4]:tag'td'
```

```
                                        :css(notTop[k] and
                                                {[isTop and 'border-top' or 'border-
                                                or {}
                                        )
                                        :attr{
                                                rowspan = ({[0] = 4, 2})[k],
                                                colspan = p.colspan
                                        }
                                h.addText(text or h.nonFunc)
                                text = nil
                        end
                        m.available = false
                else
                        return nil
                end
                return x
        end
},
bridge = {--Draw a line to the neighboring node in the same column that is not conne
        main = function(x)
                local h = p.getNodeFunc()
                h.bridge.lay[col.c][m.num - col.top + 1 + (h.topBranch() == 1 and 1
                h.addText(nonFunc)
                return x
        end,
        lay = setmetatable({}, p.nilAsTab)
},
canvas = {--Merges all cells in node. Content will be the next parameter.
        main = function(x)
                local h = p.getNodeFunc()
                if m.available then
                        tab.r = rowNum[m.r]:tag'td'
                                :attr{
                                        rowspan = 6,
                                        colspan = p.colspan
                                }
                        h.addText(h.nonFunc)
                        m.available = false
                        return x
                else
                        return nil
                end
        end
},
orphan = {--sets a flag for skipMatch to be set by p._main
        main = function(x)
                p.getNodeFunc().orphan.num = m.num
                return x
        end
},
skipAllowed = {--table of supported node functions when node is skipped (i.e. by ski
        bridge = true,
        canvas = true
}
}
```

```
setmetatable(nodeFunc.helper, {__index = nodeFunc})
function p.getNodeFunc()
	return nodeFunc.helper
end

local function newRow(bodyRow)
	local first = p.flex_tree.merge and mw.clone(p.flex_tree.cell) or p.flex_tree.cell
	tab.r = tab:tag'tr'
		:node(first)
	if bodyRow then
		table.insert(rowNum, bodyRow, tab.r)
		if p.flex_tree.merge then
			rowNum[bodyRow].first = first
			rowNum[bodyRow].first.unchanged = true
		end
	end
end

local function drawHead(text, row3rd)
	local td = (row3rd and rowNum[row3rd]:tag'td':attr{rowspan = 2}
		or head.row:tag'td')
		:attr{colspan = p.colspan}
	if text ~= 'omit_label' then
		td:wikitext(text):css{
			['text-align'] = 'center',
			border = '1px solid #aaa',
			background = p.bgColor.head
		}
	end
end

local function spacer(width)
	tab.r:tag'td'
		:attr{width = width}
		:wikitext(p.no_column_head and '' or ' ')
end

local function dpBox(v, r)
	p.dpBoxBase = p.dpBoxBase or mw.html.create'td':attr{rowspan = 2, colspan = p.colspa
	if not v then
		p.dpBoxEmpty = p.previewnumbers and mw.clone(p.dpBoxBase) or p.dpBoxEmpty or
		rowNum[r]:node(p.dpBoxEmpty)
	else
		rowNum[r]:node(mw.clone(p.dpBoxBase):wikitext(v))
	end
end

p.scoreWasher = {
	numberFormat = '%-?%d+%.?%d*',
	main = function (self, s)
		if s then
			for _, cycle in ipairs(self.cycles) do
				s = s:gsub(unpack(cycle))
			end
			if p.scoreSumBox and self.plus then
				local t = 0
```

```
                                for _, part in ipairs(mw.text.split(s, self.plus)) do
                                        t = t + (tonumber(part:match('%-?%d+%.?%d*')) or 0)
                                end
                                return t
                        end
                        return tonumber(s:match(self.numberFormat)) or math.huge
                end
                return 0
        end,
        spin = function(self, v)
                table.insert(self, v)
                return self
        end,
        load = function (self, cycle)
                local wash, rinse = 0, {spin = self.spin}
                for v in cycle:gfind('%(([^%(%)]-)%)') do
                        if v == '_plus_' then
                                self.plus = v
                                rinse:spin(v)
                                cycle = cycle:gsub('%(_plus_%)', '', 1)
                        else
                                wash = wash + 1
                                rinse:spin('%'):spin(wash)
                        end
                end
                table.insert(self.cycles, {esc.bs:undo(cycle, '%%'), table.concat(rinse)})
        end,
        init = function(self, setting)
                self.cycles = {original = setting}
                for cycle in (setting and esc.bs:text(setting) or '{<.->} {[^%d]*}'):gfind('
                        self:load(cycle)
                end
        end,
                sum = function (clean)
                local sum = {0, 0}
                for _, box in ipairs(clean) do
                        for team, score in ipairs(box) do
                                sum[team] = sum[team] + score
                        end
                end
                return unpack(math.max(unpack(sum)) == math.huge and {'&mdash;', '&mdash;'}
        end
}

local function boldWin(s1, s2)
        return setmetatable(
                p.bold and s1 ~= s2 and (math[({'min', 'max'})[p.bold]](s1, s2) == s1 and {t
                p.nilAsTab
        )
end

local function maxSpan(span, start, rows)
        return math.min(span, math.max(0, rows - start + 1))
end

--in case of templates like RDseed need padding value
```

```
p.teamBoxPadding = function()
	return '.6ex'
end
p.teamBoxPadTab = {padding = '0 ' .. p.teamBoxPadding()}
p.teamBoxNormal = {border = '1px solid #aaa', background = p.bgColor[4]}
local function teamBox(v, r, f)
	if p.flex_tree.merge and not v and f.phase == 2 then
		for i = -2, 0 do
			if rowNum[r + i].first.unchanged then
				rowNum[r + i].first.unchanged = nil
				rowNum[r + i].first:node(p.unflex_div)
			end
		end
		tab.r:attr{rowspan = 4}:css{['vertical-align'] = 'center'}
	else
		if not p.bold then
		--backwards compatability (wikitemplates bold each arg individually)
			local hasBold, b = tostring(v):gsub("([^']*)'''([^']*)", '%1<b>%2</b
			if b == 1 then
				v = hasBold
			end
		end
		local cell
		if f[1] then
			cell = f.sumBox and f.sumBox[1] and
				{padding = f.sumBox[1]}
				or {['border-left'] = f.borderLeft}
			cell['text-align'] = v and f[1]
		else
			cell = p.teamBoxPadTab
		end
		tab.r = rowNum[r]:tag'td'
			:css(p.teamBoxCSS)
			:css(cell)
			:attr{rowspan = 2}
			:node(mw.html.create(f.bold and 'b'):wikitext(v or f[1] and '' or '&
	end
end

function p._main(args)
	function args:clean(key, params)--prevent html comments from breaking named args and
		params = params or {}
		local clean = args[key] or params.ifNil
		if clean then
			params.append = params.append or ''
			clean = mw.text.decode(clean):gsub('<!%-.-%->', ''):gsub(params.patt
			clean = clean ~= params.append and clean or params.ifNil
		end
		args[key] = params.keepOld and args[key] or clean
		return clean
	end
	p.cols = tonumber(args:clean('columns', {pattern = '%D'}))
	p.tCols = (tonumber(args:clean('final_RDs_excluded', {pattern = '%D'})) or 0) + p.co
	local matchPer = {
		pattern = '%d*per%d+[%-x]%d+',
		vals = '(%d*)per(%d+)([%-x])(%d+)'
```

```
}
local skipMatch, unBold  = {}, {}--(skip|manualbold)match# to boolean
for k, _ in pairs(args) do
	local mType, mNum = string.match(k, '^(%l+)match(%d*)$')
	mType, mNum = ({skip = skipMatch, manualbold = unBold})[mType], tonumber(mNu
	if mType then
		if mNum then
			mType[mNum] = args:clean(k) == 'yes' or args[k] == 'true'
		else
			for pattern in args:clean(k, {ifNil = ''}):gfind(matchPer.pa
				local d1, period, op, d2 = pattern:match(matchPer.va
				d1 = tonumber(d1) or 1
				d2 = op == '-' and d2 or (d1 + period * (d2 - 1))
				for y = d1, d2, period do
					mType[y] = true
				end
			end
			for _, x in ipairs(mw.text.split(args[k]:gsub(matchPer.patte
				x = mw.text.split(x, '-')
				for y = tonumber(x[1]) or 1, tonumber(x[2] or x[1])
					mType[y] = true
				end
			end
		end
	end
end
for _, v in ipairs({--more args to boolean
	'widescore',
	'template',
	'article_include',
	'color',
	'3rdplace',
	'omit_blanks',
	'scroll_head_unlock',
	'previewnumbers',
	'flex_tree',
	'no_column_head',
	'short_brackets',
	'branch_upwards'
}) do
	if args[v] and (p[v] == nil or type(p[v]) == 'boolean') then
		p[v] = args:clean(v) == 'yes' or args[v] == 'true'
	end
end
p.namespace = mw.title.getCurrentTitle().namespace
p.previewnumbers = p.namespace ~= 0 and p.previewnumbers
p.scoreWasher:init(args['score-clean'])
p.scoreWasher.demo = args.demoWash and tonumber(args:clean('demoWash', {pattern = '%
p.scoreSumBox = args['score-boxes'] and args['score-boxes']:match('%d ?%+ ?sum')
p.bold = ({low = 1, high = 2})[args:clean('bold_winner')] or p.scoreSumBox and 2
local sumBox = p.scoreSumBox and 1 or 0
p.scoreBoxes = (tonumber(args:clean('score-boxes', {pattern = '%D'})) or 1) + sumBox
p.scoreSumBox = p.scoreBoxes > 0 and p.scoreSumBox or nil
local boxStyle = p.scoreBoxes > 1 and
	(p.scoreSumBox and
		setmetatable(
```

```
                        {{}, [p.scoreBoxes] = {'0 1ex'}},
                        {__call = function(t, i) if t[i] then return nil end return
                )
                or setmetatable(
                        {},
                        {__call = function() return 0 end}
                )
        )
        or setmetatable({}, {__call = function() return nil end})
p.colspan = p.scoreBoxes > 0 and (p.scoreBoxes + 1) or nil
local nodeArgs = {
        score = p.scoreBoxes - sumBox,
        team = {offset = 1 + p.scoreBoxes - sumBox}
}
nodeArgs.all = 1 + nodeArgs.team.offset * 2
nodeArgs.tableSum = {
        __add = function(v, t)
                if #t == 3 then
                        return v + nodeArgs.all
                end
                local s = v
                for i, n in ipairs(t) do
                        s = s + n
                end
                return s--[[ + (p.scoreSumBox and #t == 3 and -2 or 0) --merging dis
        end
}
nodeArgs.team[1] = 1--constant to be replaced later by new param
nodeArgs.team[2] = nodeArgs.team[1] + nodeArgs.team.offset
nodeArgs.blank = setmetatable({}, nodeArgs.tableSum)
p.unflex_div = mw.html.create'div'
                                :css{overflow = 'hidden', height = '1ex'}
                                :wikitext' '
p.flex_tree = setmetatable({},{__index = {
        merge = p.flex_tree and p.scoreBoxes == 0,
        wt = p.flex_tree and '' or ' ',
        cell = mw.html.create'td'
                :node(not p.flex_tree and p.unflex_div or nil)
}})
if args:clean'scroll_height' then
        local fontSize, fontUnit = args.style and args.style:match('font%-size *: *(
        if fontSize then
                local units = {
                        em = 1,
                        ex = 2,
                        ['%'] = 0.01
                }
                fontSize, fontUnit = {fontSize * fontUnit}
        end
end
tab
        :cssText(table.concat{args.scroll_height and 'padding' or 'margin', ':', for
        :attr{cellpadding = 0, cellspacing = 0}
if not p.no_column_head then--headings row
        newRow()
        head.row = tab.r
```

```
                :css{['white-space'] = args.scroll_height and 'nowrap'}
        newRow()
else
        tab.r = tab:tag'tr'
        tab.r:tag'td'
end
local sp = {--set column widths
        args['team-width'] or 170,
        p.widescore and 40 or 30,
        p.short_brackets and 6 or 15,
        p.short_brackets and 4 or 20
}
local scoreWidth = args:clean('score-width', {pattern = '[^%d;]'}) and mw.text.split
scoreWidth[1] = tonumber(scoreWidth[1], 10)
if p.scoreSumBox and #scoreWidth ~= 1 then
        local _scoreWidth = {}
        for k = 1, p.scoreBoxes - 1 do
                _scoreWidth[k] = tonumber(scoreWidth[k], 10) or math.ceil(sp[2] * 0.
        end
        setmetatable(scoreWidth, _scoreWidth)
end
if p.template or p.article_include then
        p.template = mw.title.new(args.name)
        p.templateFixedName = (p.template.namespace == 0 and not p.article_include a
end
p.template = p.template and mw.title.new(args:clean('name', {pattern = ''}))
local head_br = {
        count = 0,
        compare = function (self, text)
                if text and args.scroll_height then
                        local _, count = text:gsub('<br[ >/]', '%1')
                        self.count = math.max(self.count, count)
                end
                return text
        end
}
p.branch_upwards = p.branch_upwards and 0
for k = 1, p.cols do
        if k > 1 then
                spacer(sp[3])
                spacer(sp[4])
                if not p.no_column_head then
                        head.row:tag'td':attr{colspan = 2}
                end
        end
        spacer(sp[1])
        for s = 1, p.scoreBoxes do
                spacer(#scoreWidth == 1 and scoreWidth[1] or scoreWidth[s] or sp[2])
        end
        if not p.no_column_head then
                head.wt = head_br:compare(args:clean('RD' .. k, {pattern = ''}))
                        or p.RD[#p.RD + k - p.tCols - 1]
                        or ('Round of ' .. math.pow(2, p.tCols - k + 1))
                drawHead(
                        k == 1 and p.template and mw.getCurrentFrame():expandTemplat
                                title = 'navbar-header',
```

```
                                        args = {head.wt, p.templateFixedName}
                                } or head.wt
                        )
                end
        end
        sp.row = tab.r
        col.tot = math.pow(2, p.tCols - 1)
        local step, bump, bumpBase, rows = 1, 0, mw.html.create'td':attr{colspan = p.colspar
        args.line_px = table.concat{args:clean('line_px') or 3, args.line_px ~= '0' and 'px'
        tab.line = {--reduces concats and 'or' statements
                {
                        [true] = args.line_px,
                        [false] = 0
                },
                args.line_px:rep(2):gsub('(%a)(%d)', '%1 %2', 1)
        }
        p['3rdplace'] =  p.tCols == p.cols and (p['3rdplace'] or p.cols > 3 and nil == p['3r
        if p['3rdplace'] then
                p.textThird = args.Consol or args['RD' .. (p.cols + 1)] or p.RD[4]
                local no3rdText = p.no_column_head or p.textThird and p.textThird:match('omi
                rowNum.third = math.max(math.pow(2, p.branch_upwards and -3 or p.cols - 2) *
        end
        for r = 1, rowNum.third or rows do
                newRow(r)
        end
        p:saveStr('solid', tab.line[1][true], ' solid')
        p.cornerDiv = mw.html.create'div':css{height = tab.line[1][true], ['border-right'] =
        for c = 1, p.cols do
                col.c = c
                local bumps = bump
                if c > 1 then
                        col.tot = col.tot + math.pow(2, p.tCols - c)
                        if p.branch_upwards then
                                bumps = 0
                                rowNum[1]:tag'td':attr{rowspan = 4}
                        else
                        rowNum[1]:node(c < p.cols and
                                mw.clone(bumpBase):attr{rowspan = bump}
                                or p.no_column_head and p.template and
                                        mw.html.create'td':wikitext(mw.getCurrentFrame():exp
                                                title = 'navbar-header',
                                                args = {'', p.templateFixedName}
                                        })
                        )
                        end
                end
                col.top = m.num
                p.span = p.tCols > c and bump * 2 or p.branch_upwards or math.max((bump - 1)
                col.show3rd = p['3rdplace'] and c == p.tCols and rowNum.third
                local colorFinal, bumpMid = p.color and c == p.tCols, p.span > 0 and mw.clor
                for r = 1, col.show3rd or rows, 2 do
                        m.r = r + bumps
                        if col.show3rd or rowNum[m.r] and m.num <= col.tot then
                                if m.phase == 0 then
                                        m.showBox = setmetatable({1, nodeArgs.team.offset, r
                                        if nodeFunc:scanPattern(args, step) then
```

```
                        nodeFunc.called = {}
                        m.available = true
                else
                        m.available = nil
                end
        end
        if skipMatch[m.num] then
                if m.phase == 0 then
                        if nodeFunc.pattern then
                                for x, y in ipairs(nodeFunc.pattern)
                                        if nodeFunc.skipAllowed[y] t
                                                nodeFunc.called[y] =
                                        end
                                end
                        end
                        local canvas = nodeFunc.pattern and nodeFunc
                        rowNum[m.r + (canvas or 0)]:tag'td':attr{row
                elseif m.phase == 2 then
                        if nodeFunc.pattern and (nodeFunc.called.bri
                                step = step + 1
                        end
                        m.num = m.num + 1
                        step = step + (p.omit_blanks and 0 or m.show
                        bumps = bumps + (col.show3rd and 0 or maxSpa
                end
        elseif m.phase == 0 then
                if nodeFunc.pattern then
                        for x, y in ipairs(nodeFunc.pattern) do
                                if nodeFunc[y] and nodeFunc[y].main
                                        nodeFunc.called[y] = nodeFur
                                end
                        end
                        if m.available == false then
                                m.showBox = nodeArgs.blank
                                step = step + 1
                        end
                end
                if m.showBox[1] then
                        if col.show3rd then
                                col.show3rd = (m.num - col.top) * 2
                                if col.show3rd == 2 then
                                        if p.textThird:match('omit_l
                                                p.textThird = nil
                                        end
                                        if rowNum[rows + 1] and p.cc
                                                rowNum[rows + 1]:tag
                                                        rowspan = m.
                                                        colspan = (p
                                                }
                                        end
                                        if p.tCols == 1 then
                                                bumps = p.textThird
                                        elseif p.branch_upwards ther
                                                r = 7
                                                bumps = p.textThird
                                        end
```

```
                                        m.r = r + bumps
                                        if p.textThird then
                                                drawHead(p.textThirc
                                                bumps = bumps + 2
                                                m.r = r + bumps
                                        end
                                end
                        end
                        dpBox(nodeFunc.pattern and nodeFunc.nonFunc
                        if p.previewnumbers then
                                rowNum[m.r].nodes[#rowNum[m.r].nodes
                                        :tag'div'
                                                :css{
                                                        float = 'lef
                                                        border = '1p
                                                        padding = '0
                                                        ['color'] =
                                                }
                                                :wikitext(m.num)
                                                :attr{title = 'Numbe
                        end
                end
                if p.colspan then
                        m.nonEmpty = {}
                        for s = step + 2, step + nodeArgs.team.offse
                                local i = {s, s + nodeArgs.team.offs
                                if args[i[1]] or args[i[2]] then
                                        table.insert(m.nonEmpty, i)
                                end
                        end
                        if p.bold and m.showBox[2] and m.showBox[3]
                                m.bold = {
                                        box = {},
                                        clean = {}
                                }
                                local notSummed = not p.scoreSumBox
                                for s, i in ipairs(m.nonEmpty) do
                                        m.bold.clean[s] = {p.scoreWa
                                        m.bold.box[s] = notSummed ar
                                end
                                if p.scoreSumBox and m.nonEmpty[2] t
                                        local i = {-step, -step - 1}
                                        table.insert(m.nonEmpty, i)
                                        args[i[1]], args[i[2]] = p.s
                                        m.bold.box[p.scoreBoxes] = b
                                end
                                getmetatable(boxStyle).__index = p.s
                                m.bold.win = m.bold.box[#m.nonEmpty]
                        else
                                m.bold = infiniteEmpty
                        end
                end
        else
                if m.showBox[m.phase] then
                        p.teamBoxCSS = colorFinal and
                                {border = p.teamBoxNormal.border, ba
```

```
                                        or p.teamBoxNormal
                                local f = {phase = m.phase, bold = m.bold.wi
                                teamBox(args[step + nodeArgs.team[m.phase]],
                                f[1] = 'center'
                                if p.colspan then
                                        if m.nonEmpty[1] then
                                                local loneSum
                                                if #m.nonEmpty < p.scoreBoxe
                                                        loneSum = #m.nonEmpt
                                                        tab.r:attr{colspan =
                                                end
                                                for s, i in ipairs(m.nonEmpt
                                                        f.borderLeft = boxSt
                                                        f.sumBox = loneSum c
                                                        f.bold = m.bold.box[
                                                        teamBox(args[i[m.pha
                                                end
                                        else
                                                for s = 1, p.scoreBoxes do
                                                        f.borderLeft = boxSt
                                                        teamBox(nil, m.r, f)
                                                end
                                        end
                                end
                        end
                        if m.phase == 2 then
                                col.show3rd = col.show3rd ~= 2 and col.show3
                                if p.scoreWasher.demo and p.scoreWasher.demo
                                        table.insert(m.bold.clean, 1, {args[
                                        return table.concat{
                                                'Score data for match specif
                                                mw.dumpObject{scores = m.bol
                                                '<table>',
                                                tostring(sp.row), '<tr>',
                                                tostring(rowNum[m.r - 4]), '
                                                tostring(rowNum[m.r - 2]), '
                                                tostring(rowNum[m.r]), '</ta
                                        }
                                end
                                if nodeFunc.orphan.num == m.num then
                                        skipMatch[m.num] = 'orphan'
                                end
                                step = step + m.showBox
                                m.num = m.num + 1
                                if bump > 0 and rowNum[m.r + 2] and not (noc
                                        bumps = bumps + p.span
                                        rowNum[m.r + 2]:node(bumpMid)
                                end
                                r = r + (col.show3rd or bump)
                        end
                end
                m.phase = (m.phase + 1) % 3
        end
end
if p.cols > c then--draw lines to next round
        p.unit = bump + 3
```

```
bump = 3 * math.pow(2, c) - 3
bumps = p.branch_upwards and 4 or (p.unit + 1)
rowNum[1]
        :tag'td':attr{rowspan = bumps}
if not p.branch_upwards then
        rowNum[1]:tag'td'
                :attr{rowspan = (p.branch_upwards or bump) + 4}
                :css(nodeFunc.bridge.lay[c](0) and
                        {['border-right'] = p.reuseStr.solid}
                        or {}
                )
end
col.n = 0
for r = bumps + 1, rows, p.unit * 2 do
        tab.r = rowNum[r]:tag'td'
        local interval = ((r - bumps - 1) / (p.unit * 2)) % 4
        if interval % 2 == 0 then
                --col.t and col.t2 control whether lines are drawn
                col.t = col.t2 or skipMatch[col.tot + col.n / 2 + 1]
                col.n = col.n + 2
                col.t2 = skipMatch[col.tot + col.n / 2 + 1] and 3 or
                if col.t == 0 then
                        tab.r
                                :attr{rowspan = maxSpan(p.unit * 2,
                                :css(skipMatch[col.tot + col.n / 2]
                                        border = p.reuseStr.solid,
                                        ['border-left'] = 0
                                })
                else
                        tab.r
                                :attr{rowspan = maxSpan(p.unit, r, r
                                :cssText(col.t == 2 and
                                        p:saveStr('topRight', 'borde
                                        or col.t == 1 and (nodeFunc.
                                                p:saveStr('right', '
                                                or 'vertical-align:b
                                        )
                                        or nil
                                )
                                :node(col.t == 1 and interval > 0 ar
                        rowNum[r + (p.branch_upwards and (4 - bump)
                                :attr{rowspan = maxSpan(p.unit, r +
                                :cssText(col.t == 1 and
                                        p:saveStr('bttmRght', 'borde
                                        or col.t == 2 and (nodeFunc.
                                                p:saveStr('right', '
                                                or 'vertical-align:t
                                        )
                                        or nil
                                )
                                :node(col.t == 2 and interval ~= 2 a
                end
                col.t = {
                        col.t < 3,
                        rowNum[r + p.unit * 5] and col.t2 < 3 or fal
                }
```

```
                                        rowNum[r + (p.branch_upwards or p.unit)]:tag'td'
                                                :attr{rowspan = maxSpan(p.unit * 4, r + (p.b
                                                :css(interval == 0 and (col.t[1] or col.t[2]
                                                        ['border-width'] = table.concat{tab.
                                                        ['border-style'] = 'solid'
                                                } or {})
                                        else
                                                tab.r
                                                :attr{rowspan = maxSpan(p.unit * 2, r, rows)
                                                :css(nodeFunc.bridge.lay[c](col.n) and
                                                        {['border-right'] = p.reuseStr.solic
                                                        or {}
                                                )
                                        end
                                end
                        end
                end
        end
        local lock_height = (head_br.count or 0) + 1
        return args.scroll_height and
                mw.html.create'div'
                        :cssText'border-bottom:1px solid #eee;display:inline-block'
                        :node(not (p.scroll_head_unlock or p.no_column_head) and mw.html.cre
                                :css{
                                        overflow = 'hidden',
                                        height = lock_height * 1.4 + 1.6 .. 'em',
                                        ['border-bottom'] = 'inherit',
                                        ['margin-right'] = '17px'
                                }
                                :node(mw.clone(tab))
                        )
                        :tag'div'
                                :css{
                                        ['overflow-y'] = 'scroll',
                                        ['max-height'] = tonumber(args.scroll_height, 10) ar
                                }
                                :node(not (p.scroll_head_unlock or p.no_column_head) and
                                        tab:css{['margin-top'] = math.floor(-10 * (lock_heig
                                        or tab
                                )
                        :done()
                or tab
end

--[[local standard = {
        'beta' = {
                bold_winner = 'high',
                omit_blanks = 'yes',
                auto_3rd = 'yes'
        }
}--]]
function p.main(frame, columns)
        local args = require'Module:Arguments'.getArgs(frame, {trim = false})
        args.columns = args.columns or columns
        return p._main(args)
end
```

```
function p.seed(frame)
	local parent = frame:getParent() or frame
	local function arg(k, alt)
		return parent.args[k] or frame.args[k] or alt
	end
	local padding, width = arg(2, p.teamBoxPadding()), arg(3, arg('widescore') and 40 or
	padding = tonumber(padding) and tonumber(padding) .. 'px' or padding
	width = tonumber(width) and tonumber(width) .. 'px' or width
	return mw.html.create'div'
		:css{
			margin = ('-1px %s -1px -%s'):format(padding, padding),
			float = 'left',
			['background-color'] = p.bgColor.head,
			border = '1px solid #aaa',
			['text-align'] = 'center',
			width = width
		}
		:wikitext(arg(1, ' '))
end

return p
```