

semesterplanner-lua — Semesterplanner package in lua
with tikz only*

Lukas Heindl

GitLab: <https://gitlab.com/AtticusSullivan/semesterplanner-lua>

Released ?

Abstract

==== Put abstract text here. ====

Contents

1 Usage	1
2 Implementation	2
2.1 semesterplanner-lua.sty	2
2.1.1 Global Stuff	2
2.1.2 Local Stuff (timetable-env local)	3
2.2 semesterplanner-lua.lua	6
3 Change History	10
4 Index	10



1 Usage

==== Put descriptive text here. ====

`\dummyMacro` This macro does nothing. It is merely an example. If this were a real macro, you would put a paragraph here describing what the macro is supposed to do, what its mandatory and optional arguments are, and so forth.

`dummyEnv` This environment does nothing. It is merely an example. If this were a real environment, you would put a paragraph here describing what the environment is supposed to do, what its mandatory and optional arguments are, and so forth.

*This file describes version ?, last revised ?.

	Mon	Thue	Wend	Thur	Fri
11:00					
12:00					
13:00		TestingLectureLongOne Heindl  	TestingLectureLongOne Heindl RN1		
14:00					
15:00					

Special about the `length` argument is that the height of the column headers (namely the weekdaynames) isn't counted to the length you can specify.

Same goes for the width regarding the labels containing the time on the right. Since in this case any tex length is allowed, you can simply try to subtract the length of the clock label using something like `\settowidth{\length}{12:30}` to set a length to the length of a clock label and then subtract this from the length you want to specify.

Hint: The content of the environment isn't processed by thispackage, only the event commands are relevant. All other contents are set immediately before the timetable. Therefore, if you want to add e.g. a `\hspace*{10cm}` before the timetable, the last line of the env would be the place to do so (there musn't be an empty line below since otherwise a new paragraph is started).

2 Implementation

This package uses `semesterplanner-lua` as prefix/directory where possible. Since this is not possible for latex macro names, in this occasions `semesterplannerLua@` is used as prefix.

2.1 semesterplanner-lua.sty

2.1.1 Global Stuff

```
1 \package
```

Define some colors for the course types (can be globally overwritten)

```
2 \definecolor{seminar}{rgb}{1.0, 0.8, 0.0}
3 \definecolor{lecture}{rgb}{0.2, 0.7, 1.0}
4 \definecolor{tutorial}{rgb}{0.0, 0.8, 0.0}
5 \definecolor{meeting}{rgb}{0.8, 0.0, 0.0}
6 \definecolor{officehour}{rgb}{0.0, 0.4, 0.6}
7 \definecolor{DodgerBlue}{HTML}{1E90FF}
```

Load the lua module

```
8 \directlua{sp = require("semesterplanner-lua.lua")}
```

2.1.2 Local Stuff (timetable-env local)

timetable This is the environment doing all the stuff. To gate the positions where the corresponding macros can be used (and in terms of pgfkeys for reasons of default values) all the macros used are put into the environment.

```
9 \newenvironment{timetable}[1][{}]{
```

\semesterplannerLua@encircle This macro puts a circle around its argument for better readability. In this package this is used for the fontawesome symbols.

```
10 \newcommand*{\semesterplannerLua@encircle}[1]{
11 \begin{minipage}[b][1em][c]{1.5em}
12 \begin{tikzpicture}
13 \node[fill,circle,inner sep=1pt,color=white]{##1};
14 \end{tikzpicture}
15 \end{minipage}
16 }
```

Set all the pgfkeys required for the arguments. To achieve that the defaults are restored every time the environment is used, this is inside the environment definition. This of course disables all possibilities of setting a global default but enables setting local defaults for the events

```
17 \pgfkeys{
```

/semesterplanner-lua will be the pgf-path used for this package Set the environment arguments arguments. **days**, **width** and **height** are used later in drawing. **start time** and **end time** are important for collecting the events as well.

days is a list of strings representing the header names for the day columns in the timetable (adding Sat and Sun (additional entries) will result in two more columns.

length is the vertical length of the timetable (not including the clock labels on the side) measured in cm (in future versions this may become measured in pts for better interaction with the LaTeX lengths.

width is the horizontal width of the timetable (not including the column headers on the top) this can be a latex length string or **\textwidth** as well.

start time can be used to set a fixed time where the timetable starts (otherwise this is calculated from the entries) to enable this behaviour this key has to be set to HH*60 + MM (easy way is by using **start time/.evaluated={HH*60+MM}**)

end time equivalent to **start time**

```
18 /semesterplanner-lua/.cd,
19 days/.initial={Mon,Thue,Wend,Thur,Fri},
20 days/.default={Mon,Thue,Wend,Thur,Fri},
21 %
22 start time/.initial=,
23 start time/.default=,
24 end time/.initial=,
25 end time/.default=,
26 %
27 width/.initial=\textwidth,
28 width/.default=\textwidth,
29 length/.initial=10,
30 length/.default=10,
31 %
```

/semesterplanner-lua/event is the path where the keys relevant for the event macro resides

content is the content of the event (is passed on without any formatting). Since this is passed to lua without modification its value must be an unexpanded string (lua will simply print it so the eventually the string will be evaluated)

time is a HH:MM-HH:MM string representing start- and end-time of the event

day is either M,T,W,Th or F specifying the day on which the event takes place

tikz this key allows the user to manually pass options to the node created for this event

scale width allows to scale the width of the event to be able to draw overlapping events besides each other. Will usually be a value between 0 and 1.

offset same goal like **scale width** but shifts the event node by the given value to the right. (Given as value between 0 and 1 indicating how many columns the event should be shifted)

```

32     event/.cd,
33     % event arguments
34     content/.initial=,
35     content/.default=,
36     %
37     time/.initial=,
38     time/.default=,
39     day/.initial=,
40     day/.default=,
41     %
42     tikz/.initial=,
43     tikz/.default=,
44     scale width/.initial=1,
45     scale width/.default=1,
46     offset/.initial=0,
47     offset/.default=0,
48 }

```

Commands for symbols of priority

\pmandatory

```

49     \protected\def\pmandatory{\semesterplannerLua@encircle{\textcolor{red}{\faWarning}}}

```

\phigh

```

50     \protected\def\phigh{\semesterplannerLua@encircle{\textcolor{red}{\faFlag}}}

```

\pmid

```

51     \protected\def\pmid{\semesterplannerLua@encircle{\textcolor{yellow}{\faFlag}}}

```

\plow

```

52     \protected\def\plow{\semesterplannerLua@encircle{\textcolor{green}{\faFlag}}}

```

\pnone

```

53     \protected\def\pnone{\semesterplannerLua@encircle{\textcolor{gray}{\faTimesCircle}}}

```

Commands for online platforms.

\teams

```

54     \protected\def\teams{\semesterplannerLua@encircle{\textcolor{DodgerBlue}{\faWindows}}}

```

\zoom

```

55     \protected\def\zoom{\semesterplannerLua@encircle{\textcolor{DodgerBlue}{\faCamera}}}

```

\youtube

```

56     \protected\def\youtube{\semesterplannerLua@encircle{\textcolor{red}{\faYoutubePlay}}}

```

Command for "To be determined" and "To be Announced"

\tbd

```

57     \protected\def\tbd{\faQuestion}

```

\tba

```
58 \protected\def\tba{\faBullhorn}
```

Read the arguments given by the user after restoring the defaults (Restoring currently makes no sense, since they are created a few lines above anyways, but creation might be moved outside the environment some day.

Afterwards the lua module is being initialized (erase data from possible previous runs.

```
59 \pgfkeys{/semesterplanner-lua/.cd, days,length,width,start time,end time, #1}
60 \directlua{sp.init(
61     "\pgfkeysvalueof{/semesterplanner-lua/days}",
62     "\pgfkeysvalueof{/semesterplanner-lua/start time}",
63     "\pgfkeysvalueof{/semesterplanner-lua/end time}")}
```

\semesterplanner@event Is used to pass the event to the lua engine which in turn will collect the event to draw it in the end. For that the arguments given are parsed after restoring the pgf keys to their default values. The optional argument hereby is a sequence of pgf keys, the second argument is a string representing the content (this MUST be unexpanded since this is passed to lua which in turn will pass it unmodified back)

```
64 \newcommand{\semesterplannerLua@event}[2][]{
65     \pgfkeys{/semesterplanner-lua/event/.cd,content,time,day,tikz,scale width,
66     offset, ##1, content=##2}
67     \directlua{
68         sp.addEvent{
69             time="\pgfkeysvalueof{/semesterplanner-lua/event/time}",
70             day="\pgfkeysvalueof{/semesterplanner-lua/event/day}",
71             tikz=[\pgfkeysvalueof{/semesterplanner-lua/event/tikz}],
72             content=[\pgfkeysvalueof{/semesterplanner-lua/event/content}],
73             offset=\pgfkeysvalueof{/semesterplanner-lua/event/offset},
74             scale_width=\pgfkeysvalueof{/semesterplanner-lua/event/scale width},
75         }
76     }
77 }
```

semesterplannerLua@formattedEvent Simply a layer above **\semesterplannerLua@event** which formats the content before passing it on. This formatting is thought to be a good formatting for lecture-like entries and is heavily stolen from ¹ Takes a number of arguments:

1. title of the event
2. name of the speaker/lecturer
3. location (e.g. roomnumber)
4. day on which the event takes place (for valid values see the **day** pgf key above)
5. time (for valid values / formatting see the **time** pgf key above)
6. priority of the event (no special formatting needed, consider using one of **\phigh**, ...)
7. event code. This is passed to event-pgf unmodified and can overwrite any of the above keys. To add some arguments to tikz simply use **tikz/.append={draw=green}**
8. background color of the event
9. text color of the content

```
78 \def\semesterplannerLua@formattedEvent##1##2##3##4##5##6##7##8##9{
79     \semesterplannerLua@event[time=##5, day=##4, tikz={fill=##8}, ##7]
80     {
81         \unexpanded{
82             \textcolor{##9}{
83                 \textbf{##1}\[.2em]
```

¹<https://github.com/nlschn/semesterplanner/>

```

84         \raggedright{##2}\\[0.5em]\raggedright{##6}\raggedright{##3}
85     }
86 }
87 }
88 }

```

Short-hand macros for different events using the corresponding background color

```

\lecture
89     \def\lecture##1##2##3##4##5##6##7{
90         \semesterplannerLua@formattedEvent{##1}{##2}{##3}{##4}{##5}{##6}{##7}{lecture}{white}
91     }

\seminar
92     \def\seminar##1##2##3##4##5##6##7{ %##1=title, ##2=speaker, ##3=location, ##4=day, ##5=ti
code (tikz can eb set this way too but you must use append)
93         \semesterplannerLua@formattedEvent{##1}{##2}{##3}{##4}{##5}{##6}{##7}{seminar}{white}
94     }

\tutorial
95     \def\tutorial##1##2##3##4##5##6##7{ %##1=title, ##2=speaker, ##3=location, ##4=day, ##5=t
code (tikz can eb set this way too but you must use append)
96         \semesterplannerLua@formattedEvent{##1}{##2}{##3}{##4}{##5}{##6}{##7}{tutorial}{white}
97     }

\meeting
98     \def\meeting##1##2##3##4##5##6##7{ %##1=title, ##2=speaker, ##3=location, ##4=day, ##5=ti
code (tikz can eb set this way too but you must use append)
99         \semesterplannerLua@formattedEvent{##1}{##2}{##3}{##4}{##5}{##6}{##7}{meeting}{white}
100     }

\officehour
101     \def\officehour##1##2##3##4##5##6##7{ %##1=title, ##2=speaker, ##3=location, ##4=day, ##5=
code (tikz can eb set this way too but you must use append)
102         \semesterplannerLua@formattedEvent{##1}{##2}{##3}{##4}{##5}{##6}{##7}{officehour}{whi
103     }

104 }{

At the end of the environment after all events have been collected, generate and output
the tikz code needed to draw the timetable.

105     \directlua{sp.draw(
106         [[\pgfkeysvalueof{/semesterplanner-lua/length}]],
107         [[\pgfkeysvalueof{/semesterplanner-lua/width}]]})}
108 }

109 \end{package}

```

2.2 semesterplanner-lua.lua

```

110 <*luaMain>

init Initialize global variables to remove previous values (e.g. events from the previous
timetable)

days A string with the names of the weekdays for the header

min Time where the timetable should start. If empty this is calculated from the events.

max Time where the timetable should end. If empty this is calculated from the events.

```

```

111 function init(days, min, max)
112     -- clean up first
113     -- global variables
114     EVENTS={}
115     DAYS = days -- header with names of the days set from tex currently
116     DAYSE = {"M","T","W","Th","F"}
117     MIN = 25*60 -- bigger than any allowed value could be
118     MAX = 0
119     MIN_BYPASS = false -- weather min is fixed by the user
120     MAX_BYPASS = false -- weather max is fixed by the user
121
122     if(min == "") then
123     else
124         assert(min:match("^%d+"), "start time has to be an integer representing the HH*60+MM of
125         MIN = tonumber(min)
126         MIN_BYPASS = true
127     end
128
129     if(max == "") then
130     else
131         assert(max:match("^%d+"), "end time has to be an integer representing the HH*60+MM of
132         MAX = tonumber(max)
133         MAX_BYPASS = true
134     end
135 end

```

addEvent Adds the event to the EVENTS array after some validity checks, modifies MIN/MAX if necessary

```

136 -- result are the global variables EVENTS, MIN and MAX
137 function addEvent(opts)
138     if(not checkKeys(opts, {"time", "day", "content", "tikz"})) then
139         error("missing argument")
140     end
141
142     opts.from,opts.to = dur2Int(opts.time)
143     -- TODO convert day to corresponding number
144
145     if(not MIN_BYPASS and opts.from < MIN) then MIN = opts.from end
146     if(not MAX_BYPASS and opts.to > MAX) then MAX = opts.to end
147
148     table.insert(EVENTS, opts)
149 end

```

draw Draws the tikz-timetable with the global variables EVENTS, MIN, MAX, DAYSE and DAYS. In addition length and width are given as direct parameters.

```

150 -- parameters are all global variables
151 function draw(length, width)
152     print("length", length)
153     print("width", width)
154     -- copy relevant variables for working on local copies
155     local events = copy_array(EVENTS)
156     local days = prepareDays(DAYS)
157     local daysE = copy_array(DAYSE)
158     local min, minH, max, maxH = prepareMinMax(MIN, MAX)
159
160     assert(length:match("%d*%.?%d*"), "Length must be a valid length measured in cm")
161     length = tonumber(length)
162
163     textwidth = width
164
165     tex.print([[begin{tikzpicture}]]])
166     tex.print([[tikzset{defStyle/.style={font=\tiny,anchor=north west,fill=blue!50,draw=black

```

Draw the grid of the timetable along with clock and day labels

```

167 -- print the tabular with the weekday headers
168 tex.print(string.format(
169     [[\foreach \week [count=\x from 0, evaluate=\x as \y using \x+0.5] in {%s}{ }],
170     table.concat(days, ",")
171 )
172 )
173 tex.print(string.format(
174     [[\node[anchor=south] at (\y/%d* %s, 0) {\week};]], #days, textwidth))
175 tex.print(string.format(
176     [[\draw (\x/%d * %s, 0cm) -- (\x/%d * %s, %dcm);]],
177     #days,
178     textwidth,
179     #days,
180     textwidth, -length
181 )
182 )
183 tex.print("{}")
184 tex.print(string.format(
185     [[\draw (%s, 0) -- (%s,%dcm);]],
186     textwidth,
187     textwidth,
188     -length
189 )
190 )
191
192 for i=minH,maxH do
193     tex.print(string.format(
194         [[\node[anchor=east] at (0,%fcm ) {%d:00};]],
195         minuteToFrac(i*60,min,max)*-length, i
196     )
197 )
198     tex.print(string.format(
199         [[\draw (0,%fcm ) -- (%s,%fcm );]],
200         minuteToFrac(i*60,min,max)*-length,
201         textwidth,
202         minuteToFrac(i*60,min,max)*-length
203     )
204 )
205 end
206
207 Draw the nodes of the events
208 local d
209 local red = 0.3333 -- calculated in em from inner sep
210 local red_y = 0.25 -- calculated in em
211 for _,e in ipairs(events) do
212     d = search_array(daysE, e.day) - 1
213     tex.print(string.format(
214         [[\node[defStyle,text width=-%fem+%f*s/%d, text depth=%fcm-%fem, text height=%fem
215         2*red, -- text width
216         e.scale_width, -- text width
217         textwidth,
218         #days, -- text width
219         length*(e.to-e.from)/(max-min), -- text depth
220         2*red+red_y, -- text depth
221         red_y, -- text height
222         e.tikz, -- free tikz code
223         (d+e.offset)/#days, -- xcoord
224         textwidth,
225         minuteToFrac(e.from,min,max)*-length, -- ycoord
226         e.content -- content
227     )
228 end

```



```

229     tex.print([[\\end{tikzpicture}]])
230 end

searchArray Searches an array for a given value and returns the index if found. On error nil is
returned
231 function search_array(t, s)
232     for k,v in ipairs(t) do
233         if(v == s) then return k end
234     end
235     return nil
236 end
237

minuteToFrac Calculates at which fraction of the total duration of max-min the time minute is located
238 function minuteToFrac(minute, min, max)
239     return (minute-min)/(max-min)
240 end

prepareMinMax Calculates the next hour of MIN (next before) and MAX (next after) and returns it (the
hour) and the corresponding min/max (same in minutes)
241 function prepareMinMax(min, max)
242     local minH = math.floor(min/60)
243     local maxH = math.ceil(max/60)
244     local min = minH*60
245     local max = maxH*60
246     return min, minH, max, maxH
247 end

checkKeys Checks if all ks are present in table t
248 function checkKeys(t, k)
249     for _,x in ipairs(k) do
250         if(t[x] == nil) then
251             return false
252         end
253     end
254     return true
255 end

dur2Int Takes a clock duration formatted as HH:MM-HH:MM, splits it, checks for validity and returns
begin/end time in minutes
256 function dur2Int(clk)
257     local f1,f2, t1,t2 = clk:match("(%d%d?):(%d%d)-(%d%d?):(%d%d)$")
258     if(f1 ~= nil and f2 ~= nil and t1 ~= nil and t2 ~= nil) then
259         f1 = tonumber(f1) f2 = tonumber(f2)
260         t1 = tonumber(t1) t2 = tonumber(t2)
261         assert(f1 >= 0 and f1 < 24, "Hours have to be >= 0 && < 24")
262         assert(f2 >= 0 and f2 < 60, "Mins have to be >= 0 && < 60")
263         assert(t1 >= 0 and t1 < 24, "Hours have to be >= 0 && < 24")
264         assert(t2 >= 0 and t2 < 60, "Mins have to be >= 0 && < 60")
265         return f1*60 + f2, t1*60 + t2
266     else
267         error("clk string \"" .. clk .. "\" was no valid clock string")
268     end
269 end

prepareDays Splits the comma-sep string days into an array
270 function prepareDays(days)
271     local ret = {}
272     for m in days:gmatch("[^,]+") do
273         table.insert(ret, m)
274     end
275     return ret
276 end

```

```

copyArray Returns a copy of the table obj
277
278 function copy_array(obj)
279     if type(obj) ~= 'table' then return obj end
280     local res = {}
281     for k, v in pairs(obj) do
282         local c = copy_array(v)
283         res[copy_array(k)] = c
284     end
285     return res
286 end

```

Prepare the module semesterplannerLua for exporting (only the functions that should be public)

```

287
288 semesterplannerLua = {
289     init = init,
290     addEvent = addEvent,
291     draw = draw
292 }
293 return semesterplannerLua
294 </luaMain>

```

3 Change History

v1.00
 General: First public release 1

4 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in **roman** refer to the code lines where the entry is used.

A		M	\seminar <u>92</u>
\addEvent <u>136</u>		\meeting <u>98</u>	
C		\minuteToFrac <u>238</u>	T
\checkKeys <u>248</u>		O	\tba <u>58</u>
\copyArray <u>277</u>		\officehour <u>101</u>	\tbd <u>57</u>
D		P	\teams <u>54</u>
doing nothing <i>1</i>		\phigh <u>50</u>	\tikzset <u>166</u>
\draw . . . <u>150</u> , <u>176</u> , <u>185</u> , <u>199</u>		\plow <u>52</u>	timetable (environment) . <u>9</u>
dummyEnv (environment) . . <i>1</i>		\pmandatory <u>49</u>	\tiny <u>166</u>
\dummyMacro <i>1</i>		\pmid <u>51</u>	\tutorial <u>95</u>
\dur2Int <u>256</u>		\pnone <u>53</u>	
E		\prepareDays <u>270</u>	W
environments:		\prepareMinMax <u>241</u>	\week <u>169</u> , <u>174</u>
dummyEnv <i>1</i>		S	X
timetable <u>9</u>		\searchArray <u>231</u>	\x <u>169</u> , <u>176</u>
F		\semesterplanner@event <u>64</u>	
\foreach <u>169</u>		\semesterplannerLua@encircle	Y
I	 <u>10</u> , <u>49</u> , <u>50</u> ,	\y <u>169</u> , <u>174</u>
\init <u>111</u>		<u>51</u> , <u>52</u> , <u>53</u> , <u>54</u> , <u>55</u> , <u>56</u>	\youtube <u>56</u>
L		\semesterplannerLua@event	
\lecture <u>89</u>	 <u>64</u> , <u>79</u>	Z
		\semesterplannerLua@formattedEvent	\zoom <u>55</u>
		<u>78</u> , <u>90</u> , <u>93</u> , <u>96</u> , <u>99</u> , <u>102</u>	