

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=title
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=title
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=title
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=title
code (tikz can eb set this way too but you must use append)
102         \semesterplannerLua@formattedEvent{##1}{##2}{##3}{##4}{##5}{##6}{##7}{officehour}{white}
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 validiy checks, modifys 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
144     if(not MIN_BYPASS and opts.from < MIN) then MIN = opts.from end
145     if(not MAX_BYPASS and opts.to > MAX) then MAX = opts.to end
146
147     table.insert(EVENTS, opts)
148 end

draw

149 -- parameters are all global variables
150 function draw(length, width)
151     print("length", length)
152     print("width", width)
153     -- copy relevant variables for working on local copies
154     local events = copy_array(EVENTS)
155     local days = prepareDays(DAYS)
156     local daysE = copy_array(DAYSE)
157     local min, minH, max, maxH = prepareMinMax(MIN, MAX)
158
159     assert(length:match("%d*%.?%d*"), "Length must be a valid length measured in cm")
160     length = tonumber(length)
161
162     textwidth = width
163
164     tex.print([[\\begin{tikzpicture}]]])
165     tex.print([[\\tikzset{defStyle/.style={font=\tiny,anchor=north west,fill=blue!50,draw=black}}]])
166
167     -- print the tabular with the weekday headers
168     tex.print(string.format([[\\foreach \\week [count=\x from 0, evaluate=\x as \y using \x+0.5]]

```

```

169 tex.print(string.format([[\\node[anchor=south] at (\\y/%d* %s, 0) {\\week};]], #days, textwidth,
170 tex.print(string.format([[\\draw (\\x/%d * %s, 0cm) -- (\\x/%d * %s, %dcm);]], #days, textwidth,
length))
171 tex.print("}")
172 tex.print(string.format([[\\draw (%s, 0) -- (%s,%dcm);]], textwidth, textwidth, -
length))
173
174 for i=minH,maxH do
175 tex.print(string.format([[\\node[anchor=east] at (0,%fcm ) {%d:00};]], minuteToFrac(i*length, i))
176 tex.print(string.format([[\\draw (0,%fcm ) -- (%s,%fcm );]], minuteToFrac(i*60,min,max)*length, textwidth, minuteToFrac(i*60,min,max)*length))
177 end
178
179 local d
180 local red = 0.3333 -- calculated in em from inner sep
181 local red_y = 0.25 -- calculated in em
182 for _,e in ipairs(events) do
183 d = search_array(daysE, e.day) - 1
184 tex.print(string.format(
185 [[\\node[defStyle,text width=-%fem+%f%s/%d, text depth=%fcm-%fem, text height=%fem
186 2*red, -- text width
187 e.scale_width, -- text width
188 textwidth,
189 #days, -- text width
190 length*(e.to-e.from)/(max-min), -- text depth
191 2*red+red_y, -- text depth
192 red_y, -- text height
193 e.tikz, -- free tikz code
194 (d+e.offset)/#days, -- xcoord
195 textwidth,
196 minuteToFrac(e.from,min,max)*length, -- ycoord
197 e.content -- content
198 )
199 )
200 end
201 tex.print([[\\end{tikzpicture}]]))
202 end

searchArray
203
204 function search_array(t, s)
205 for k,v in ipairs(t) do
206 if(v == s) then return k end
207 end
208 return nil
209 end
210

minuteToFrac
211 function minuteToFrac(minute, min, max)
212 return (minute-min)/(max-min)
213 end

prepareMinMax
214
215 function prepareMinMax(min, max)
216 local minH = math.floor(min/60)
217 local maxH = math.ceil(max/60)
218 local min = minH*60
219 local max = maxH*60
220 return min, minH, max, maxH
221 end

```


checkKeys

```
222
223 function checkKeys(t, k)
224     for _,x in ipairs(k) do
225         if(t[x] == nil) then
226             return false
227         end
228     end
229     return true
230 end
231
```

dur2Int

```
232 function dur2Int(clk)
233     local f1,f2, t1,t2 = clk:match("(%d%d?):(%d%d)-(%d%d?):(%d%d)$")
234     if(f1 ~= nil and f2 ~= nil and t1 ~= nil and t2 ~= nil) then
235         f1 = tonumber(f1) f2 = tonumber(f2)
236         t1 = tonumber(t1) t2 = tonumber(t2)
237         assert(f1 >= 0 and f1 < 24, "Hours have to be >= 0 && < 24")
238         assert(f2 >= 0 and f2 < 60, "Mins have to be >= 0 && < 60")
239         assert(t1 >= 0 and t1 < 24, "Hours have to be >= 0 && < 24")
240         assert(t2 >= 0 and t2 < 60, "Mins have to be >= 0 && < 60")
241         return f1*60 + f2, t1*60 + t2
242     else
243         error("clk string \"" .. clk .. "\" was no valid clock string")
244     end
245 end
```

prepareDays

```
246
247 function prepareDays(days)
248     local ret = {}
249     for m in days:gmatch("[^,]+") do
250         table.insert(ret, m)
251     end
252     return ret
253 end
```

copyArray

```
254
255 function copy_array(obj)
256     if type(obj) ~= 'table' then return obj end
257     local res = {}
258     for k, v in pairs(obj) do
259         local c = copy_array(v)
260         res[copy_array(k)] = c
261     end
262     return res
263 end
```

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

```
264
265 semesterplannerLua = {
266     init = init,
267     addEvent = addEvent,
268     draw = draw
269 }
270 return semesterplannerLua
271 </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>211</u>	T
\checkKeys <u>222</u>			\tba <u>58</u>
\copyArray <u>254</u>	O		\tbd <u>57</u>
D		\officehour <u>101</u>	\teams <u>54</u>
doing nothing <i>1</i>	P		\tikzset <u>165</u>
\draw . . . <u>149</u> , <u>170</u> , <u>172</u> , <u>176</u>	\phigh <u>50</u>		timetable (environment) . <u>9</u>
dummyEnv (environment) . . <i>1</i>	\plow <u>52</u>		\tiny <u>165</u>
\dummyMacro <i>1</i>	\pmandatory <u>49</u>		\tutorial <u>95</u>
\dur2Int <u>232</u>	\pmid <u>51</u>		
E		\pnone <u>53</u>	W
environments:	\prepareDays <u>246</u>		\week <u>168</u> , <u>169</u>
dummyEnv <i>1</i>	\prepareMinMax <u>214</u>		
timetable <u>9</u>	S	X	
F		\searchArray <u>203</u>	\x <u>168</u> , <u>170</u>
\foreach <u>168</u>		\semesterplanner@event <u>64</u>	
I		\semesterplannerLua@encircle	Y
\init <u>111</u>	 <u>10</u> , <u>49</u> , <u>50</u> ,	\y <u>168</u> , <u>169</u>
L		<u>51</u> , <u>52</u> , <u>53</u> , <u>54</u> , <u>55</u> , <u>56</u>	\youtube <u>56</u>
\lecture <u>89</u>		\semesterplannerLua@event	
	 <u>64</u> , <u>79</u>	
		\semesterplannerLua@formattedEvent	Z
		<u>78</u> , <u>90</u> , <u>93</u> , <u>96</u> , <u>99</u> , <u>102</u>	\zoom <u>55</u>