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

Lukas Heindl
: https://gitlab.com/AtticusSullivan/semesterplanner-lua

Released ?

Abstract

This package provides a mean to easily print a timetable e.g. for a semesterplan. The reason for this package to exist is that I wanted to reimplement https://github.com/nlschn/semesterplanner/ with printing the timetable with `tikz` only (which is more easily to be modified) and with the ability to make entries spanning only a fraction of the column (for showing simultanious events).

Documents using this package need to be compiled with LuaLaTeX. The package requires `xcolor`, `fontawesome`, `tikz` (and `pgfkeys`).

Contents

1 Usage

1.1 timetable

`timetable` `\begin{timetable}[opts]\ldots\end{timetable}`
`opts` are of course optional arguments:

`days` List of the names of the days that should be set as column names. Note that if you specify only 4 names only these 4 columns will be printed (with the first day being identified as Monday) *Default:* *`Mon,Thue,Wend,Thur,Fri`*

`start time` Explicit start-time of the timetable given in minutes (`HH*60 + MM`). Can be set as `start time/.evaluated={HH*60 + MM}`. If this is empty, the start time is derived from the given events. *Default:* *`""`*

*This file describes version ?, last revised ?.

end time Equivalent to **start-time** *Default: ""*

width Give the width of the timetable. (can be given e.g. as **\textwidth** as this is directly given to tikz). *Default:* **\textwidth**

length Give the length of the timetable (measured in **cm**) (has to be a straight number since this is needed in calculation) *Default: 10*

This is the core environment of this package. Within it you can use **\lecture**, **\seminar**, **\tutorial**, **\officehour** and **\meeting**. All these commands are only defined inside the **timetable** environment, and have the same structure.

```
\lecture     \lecture   {Name}{Lecturer}{Place}{Day}{Time}{Priority}{Event-code}
\tutorial    \tutorial  {Name}{Lecturer}{Place}{Day}{Time}{Priority}{Event-code}
\seminar     \seminar   {Name}{Lecturer}{Place}{Day}{Time}{Priority}{Event-code}
\officehour  \officehour{Name}{Lecturer}{Place}{Day}{Time}{Priority}{Event-code}
\meeting     \meeting   {Name}{Lecturer}{Place}{Day}{Time}{Priority}{Event-code}
```

Name Give the name of the lecture

Lecturer Give the name of the lecturer

Place Give the place of the event (most probably the room or an online plattform, see 1.2)

Day The weekday on which the event takes place. Has to be one of **M**, **T**, **W**, **Th**, **F** for **M**onday, **T**huesday, **W**ednesday, **Th**ursday, **F**riday. Might become customizable in a future version.

Time The timespan of the event formatted as **HH:MM-HH:MM** (24H clock)

Priority The priority of the event (see 1.2)

Event-code Free customizable event code. See the documentation at the end for keys that can be used here (all keys in **/event**). To simply pass arguments to the tikz-node that is being created for the event use **tikz/.append={your arguments}** (be careful with **text width**, **text height**, **text depth** as these keys are being used for the dimensions of the node as well as with **anchor**)

The entries **Day** and **Time** are mandatory since they are needed for the positioning of the node. All others are merely necessary for the content of the node and are therefore nor mandatory.

1.1.1 Special Notes

Note that the **length** argument does specify the length of the timetable without taking account of the column headers.

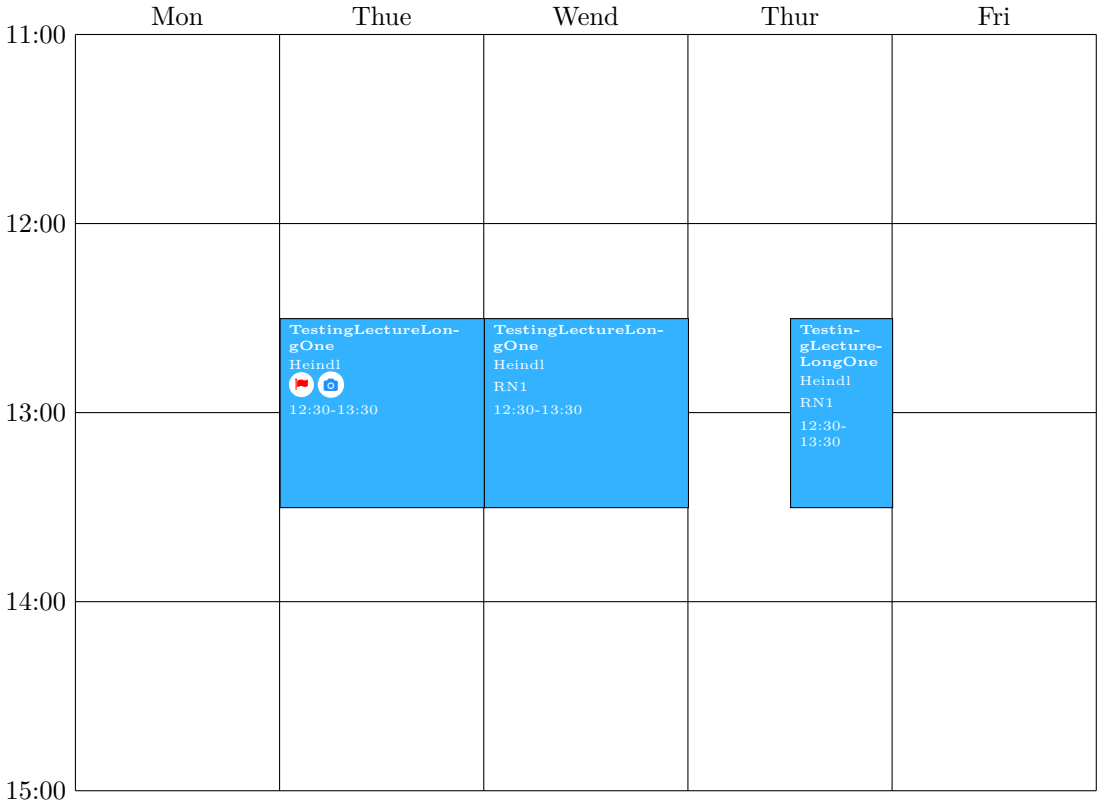
Same goes for the **width** parameter regarding the labels containing the time on the right. Since in this case any tex-lenght is allowed, you can simply try to subtract the length of the clock label using something like **\settowidth{\length}{12:30}** to set a length to the length of a clock label and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

Hint: The content of the environment isn't processed by this package. Only the event commands (so to speak **\lecture**,**\tutorial**,**\seminar**,**\officehour**,**\meeting** are relevant. All other contents are set immediately before the timetable. Therefore, if you wan to add e.g. a **\hspace*{10cm}** to shift the timetable to the left, 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).

1.1.2 Example

```
\begin{timetable}[
        days={Mon,Thue,Wend,Thur,Fri}, start
        time/.evaluated={11*60}, end time/.evaluated={15*60}
    ]
    \lecture[title={TestingLectureLongOne},speaker={Heindl},location={RN1},day={W},time
13:30}]
    \lecture[title={TestingLectureLongOne},speaker={Heindl},location={RN1},day={Th},tim
13:30},offset=0.5,scale width=0.5]
    \lecture[title={TestingLectureLongOne},speaker={Heindl},location={\zoom},day={T},ti
13:30},prio={\phigh}]
\end{timetable}
```

Timetable



1.2 Icons

This package defines some modified fontawesome icons (they are being encircled with a white circle for better readability).

\zoom		\teams	
\BBB		\youtube	
\pmandatory		\phigh	
\pmid		\plow	
\pnone			
\tbd		\tba	

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}
```

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

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

Commands for exams

`\oral`

```
15 \protected\def\oral{\faComment}
```

`\written`

```
16 \protected\def\written{\faPencil}
```

Commands for symbols of priority

`\pmandatory`

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

`\phigh`

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

`\pmid`

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

`\plow`

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

`\pnone`

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

Commands for online platforms.

`\teams`

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

`\zoom`

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

\youtube

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

\BBB

```
25    \protected\def\BBB{\semesterplannerLua@encircle{\textcolor{DodgerBlue}{\faBold}}}
```

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

\tbd

```
26    \protected\def\tbd{\faQuestion}
```

\tba

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

Load the lua modules

```
28 \directlua{sp = require("semesterplanner-lua-timetable.lua")}
29 \directlua{cal = require("semesterplanner-lua-calendar.lua")}
```

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

```
30    \pgfkeys{
```

`/semesterplanner-lua` will be the pgf-path used for this package. Here all used keys are set (and initialized with defaults. `timetable/env/`:

`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.

`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`

`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.

`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.

```
31        /semesterplanner-lua/timetable/env/.cd,
32        days/.initial={Mon,Thue,Wend,Thur,Fri}, days/.default={Mon,Thue,Wend,Thur,Fri},
33        %
34        start time/.initial=, start time/.default=,
35        end time/.initial=, end time/.default=,
36        %
37        width/.initial=\textwidth, width/.default=\textwidth,
38        length/.initial=10, length/.default=10,
39        %
```

`timetable/event/`:

`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. Used in constructing the content as well

`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)

textcolor foreground color of the content text

title title (set in bold by default)

speaker

location

prio

formatter this is special

```
        /semesterplanner-lua/timetable/event/.cd,
        % event arguments
        content/.initial=, content/.default=,
        %
        time/.initial=, time/.default=,
        day/.initial=, day/.default=,
        %
        tikz/.initial=, tikz/.default=,
        scale width/.initial=1, scale width/.default=1,
        offset/.initial=0, offset/.default=0,
        %
        textcolor/.initial=, textcolor/.default=,
        title/.initial=, title/.default=,
        speaker/.initial=, speaker/.default=,
        location/.initial=, location/.default=,
        prio/.initial=, prio/.default=,
        formatter/.initial=timetableformatter, formatter/.default=timetableformatter,
        %
```

calendar/:

draw

room

prio

course

desc

start

end

tikz

period

```
        /semesterplanner-lua/calendar/.cd,
        draw/.initial={true}, draw/.default={true},
        room/.initial={}, room/.default={},
        time/.initial={}, time/.default={},
        prio/.initial={}, prio/.default={},
        course/.initial={}, course/.default={},
        desc/.initial={}, desc/.default={},
        type/.initial={}, type/.default={},
        date/.initial={}, date/.default={},
        end/.initial={}, end/.default={},
        tikz/.initial={}, tikz/.default={},
        period/.initial={nil}, period/.default={nil},
    }
```

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.

```
\newenvironment{timetable}[1][]{
    \section*{\faClockO~Timetable}
```

Read the argumens 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 beeing initialized (erase data from possible previous runs.

```
    \pgfkeys{/semesterplanner-lua/timetable/env/.cd, days,start time,end time, width,length, 
    \directlua{sp.init(
        "\pgfkeysvalueof{/semesterplanner-lua/timetable/env/days}",
        "\pgfkeysvalueof{/semesterplanner-lua/timetable/env/start time}",
        "\pgfkeysvalueof{/semesterplanner-lua/timetable/env/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 herby 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)

```
    \newcommand{\semesterplannerLua@event}[1][]{
        \pgfkeys{/semesterplanner-lua/timetable/event/.cd,content,time,day,tikz,scale
width,offset,textcolor,title,speaker,location,prio,formatter, ##1}
        \directlua{
            sp.addEvent{
                time="\pgfkeysvalueof{/semesterplanner-lua/timetable/event/time}",
                day="\pgfkeysvalueof{/semesterplanner-lua/timetable/event/day}",
                tikz=[[\pgfkeysvalueof{/semesterplanner-lua/timetable/event/tikz}]],
                offset=\pgfkeysvalueof{/semesterplanner-lua/timetable/event/offset},
                scale_width=\pgfkeysvalueof{/semesterplanner-lua/timetable/event/scale width}
                formatter=\pgfkeysvalueof{/semesterplanner-lua/timetable/event/formatter},
                textcolor=[[\pgfkeysvalueof{/semesterplanner-lua/timetable/event/textcolor}]]
                title=[[\pgfkeysvalueof{/semesterplanner-lua/timetable/event/title}]],
                speaker=[[\pgfkeysvalueof{/semesterplanner-lua/timetable/event/speaker}]],
                location=[[\pgfkeysvalueof{/semesterplanner-lua/timetable/event/location}]],
                prio=[[\pgfkeysvalueof{/semesterplanner-lua/timetable/event/prio}]],
            }
        }
    }
```

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

`\lecture`

```
    \newcommand{\lecture}[1][]{
        \semesterplannerLua@event[tikz={fill=lecture,}, textcolor=white, ##1]
    }
```

`\seminar`

```
    \newcommand{\seminar}[1][]{
        \semesterplannerLua@event[tikz={fill=seminar,}, textcolor=white, ##1]
    }
```

`\tutorial`

```
    \newcommand{\tutorial}[1][]{
        \semesterplannerLua@event[tikz={fill=tutorial,}, textcolor=white, ##1]
    }
```

`\meeting`

```
    \newcommand{\meeting}[1][]{
        \semesterplannerLua@event[tikz={fill=meeting,}, textcolor=white, ##1]
    }
```

\officehour

```
    \newcommand{\officehour}[1][]{
        \semesterplannerLua@event[tikz={fill=officehour,}, textcolor=white, ##1]
    }

}{
```

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

```
    \directlua{sp.draw(
        [[\pgfkeysvalueof{/semesterplanner-lua/timetable/env/length}]],
        [[\pgfkeysvalueof{/semesterplanner-lua/timetable/env/width}]])}
}
```

printSpCalendar Print a calendar from startDate to endDate (encoded as YYYY-MM-DD) as one calendar per month in a matrix with the given amount of columns

```
\newcommand{\printSpCalendar}[3][3]{\directlua{cal.drawCalendar("#2", "#3", #1)}}

\newenvironment{appointments}[2][Room]{
    \directlua{cal.init(#2)}
    \newcommand{\appointment}[1][]{
        \pgfkeys{/semesterplanner-lua/calendar/.cd,draw,room,time,prio,course,desc,date,end,t
        \directlua{
            cal.addAppointment
            {
                draw=\pgfkeysvalueof{/semesterplanner-lua/calendar/draw},
                room=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/room}]],
                time=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/time}]],
                prio=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/prio}]],
                course=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/course}]],
                desc=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/desc}]],
                date=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/date}]],
                endDate=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/end}]],
                tikz=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/tikz}]],
                period=\pgfkeysvalueof{/semesterplanner-lua/calendar/period}
            }
        }
    }
    \section*{\faCalendar~Appointments}
    \begin{tabular}{rllll}
        \textbf{Date}&\textbf{Time}&\textbf{Course}&\textbf{Description}&\textbf{#1}&\textbf{
}{
    \end{tabular}
}

\newenvironment{exams}[1]{
    \directlua{cal.init(#1)}
    \newcommand{\exam}[1][]{
        \pgfkeys{/semesterplanner-lua/calendar/.cd,draw,room,time,prio,course,desc,date,end,t
        \directlua{
            cal.addExam
            {
                draw=\pgfkeysvalueof{/semesterplanner-lua/calendar/draw},
                room=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/room}]],
                time=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/time}]],
                course=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/course}]],
                desc=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/desc}]],
                date=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/date}]],
                tikz=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/tikz}]],
                type=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/type}]],
            }
```

```
        }
    }
    \section*{\faStickyNote0~Exams}
    \begin{tabular}{rllll}
        \textbf{Date}&\textbf{Time}&\textbf{Course}&\textbf{Type}&\textbf{Note}\\
}{
    \end{tabular}
}

\newenvironment{deadlines}[1]{
    \directlua{cal.init(#1)}
    \newcommand{\deadline}[1][]{
        \pgfkeys{/semesterplanner-lua/calendar/.cd,draw,room,time,prio,course,desc,date,end,t
        \directlua{
            cal.addDeadline
            {
                draw=\pgfkeysvalueof{/semesterplanner-lua/calendar/draw},
                course=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/course}]],
                desc=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/desc}]],
                date=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/date}]],
                tikz=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/tikz}]],
                prio=[[\pgfkeysvalueof{/semesterplanner-lua/calendar/prio}]],
            }
        }
    }
    \section*{\faStickyNote0~Deadlines}
    \begin{tabular}{rlll}
        \textbf{Date}&\textbf{Course}&\textbf{Description}&\textbf{Prio}\\
}{
    \end{tabular}
}
```

⟨/package⟩

2.2 semesterplanner-lua-timetable.lua

⟨*luaTimetable⟩

`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.

```
function init(days, min, max)
    -- clean up first
    -- global variables
    EVENTS={}
    DAYS = days -- header with names of the days set from tex currently
    DAYSE = {"M","T","W","Th","F"}
    MIN = 25*60 -- bigger than any allowed value could be
    MAX = 0
    MIN_BYPASS = false -- weather min is fixed by the user
    MAX_BYPASS = false -- weather max is fixed by the user

    if(min == "") then
    else
        assert(min:match("^%d+"), "start time has to be an integer representing the HH*60+MM
        MIN = tonumber(min)
        MIN_BYPASS = true
    end

    if(max == "") then
```

```
215     else
216         assert(max:match("^%d+"), "end time has to be an integer representing the HH*60+MM of
217         MAX = tonumber(max)
218         MAX_BYPASS = true
219     end
220 end
221
222 function defaultFormatter(opts)
223     ret = ""
224     for k,v in pairs(opts) do
225         if type(k) == "string" then k = k:gsub("[_^]", "") end
226         if type(v) == "string" then v = v:gsub("[_^]", "") end
227         ret = string.format("%s, %s: %s", ret, tostring(k), tostring(v))
228     end
229     print(ret)
230     return ret
231 end
232
233 function timetableformatter(opts)
234     return string.format(
235         [[\textcolor{%s}{\textbf{%s}\\[.2em]\raggedright{%s}\\[0.5em]\raggedright{%s}\raggedr
236             opts.textcolor, opts.title, opts.speaker, opts.prio, opts.location, opts.time)
237 end
```

`addEvent` Adds the event to the `EVENTS` array after some validiy checks, modifys MIN/MAX if necessary

```
238 -- result are the global variables EVENTS, MIN and MAX
239 function addEvent(opts)
240     print("Reading event on line ", tex.inputlineno)
241     opts.inputlineno = tex.inputlineno
242     if(not checkKeys(opts, {"time", "day", "tikz"})) then
243         error("missing argument")
244     end
245
246     if opts.content == nil then
247         if opts.formatter == nil then
248             opts.content = defaultFormatter(opts)
249         else
250             opts.content = opts.formatter(opts)
251         end
252     end
253
254     opts.from,opts.to = dur2Int(opts.time)
255
256     if(not MIN_BYPASS and opts.from < MIN) then MIN = opts.from end
257     if(not MAX_BYPASS and opts.to   > MAX) then MAX = opts.to   end
258     assert(opts.from < opts.to, "From has to be before to")
259
260     table.insert(EVENTS, opts)
261 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.

```
262 -- parameters are all global variables
263 function draw(length, width)
264     -- copy relevant variables for working on local copies
265     local events = copy_array(EVENTS)
266     local days = prepareDays(DAYS)
267     local daysE = copy_array(DAYSE)
268     local min, minH, max, maxH = prepareMinMax(MIN, MAX)
269
270     assert(length:match("%d*%.?%d*"), "Length must be a valid length measured in cm")
271     length = tonumber(length)
```

```
272
273     textwidth = width
274
275     tex.print([[\begin{tikzpicture}]])
276     tex.print([[\tikzset{defStyle/.style={font=\tiny,anchor=north west,fill=blue!50,draw=blacl
```

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

```
277     -- print the tabular with the weekday headers
278     tex.print(string.format(
279         [[\foreach \week [count=\x from 0, evaluate=\x as \y using \x+0.5] in {%s}{ ]],
280         table.concat(days, ",")
281         )
282     )
283     tex.print(string.format(
284         [[\node[anchor=south] at (\y/%d* %s, 0) {\week};]], #days, textwidth))
285     tex.print(string.format(
286         [[\draw (\x/%d * %s, 0cm) -- (\x/%d * %s, %dcm);]],
287         #days,
288         textwidth,
289         #days,
290         textwidth, -length
291         )
292     )
293     tex.print("}")
294     tex.print(string.format(
295         [[\draw (%s, 0) -- (%s,%dcm);]],
296         textwidth,
297         textwidth,
298         -length
299         )
300     )
301
302     for i=minH,maxH do
303         tex.print(string.format(
304             [[\node[anchor=east] at (0,%fcm ) {%d:00};]],
305             minuteToFrac(i*60,min,max)*-length, i
306             )
307         )
308         tex.print(string.format(
309             [[\draw (0,%fcm ) -- (%s,%fcm );]],
310             minuteToFrac(i*60,min,max)*-length,
311             textwidth,
312             minuteToFrac(i*60,min,max)*-length
313             )
314         )
315     end
316
```

Draw the nodes of the events

```
317     local d
318     local red = 0.3333 -- calculated in em from inner sep
319     local red_y = 0.25 -- calculated in em
320     for _,e in ipairs(events) do
321         if e.from < max and e.to > min then -- only draw if event is in scope (part of the co
322             if e.to   > max then e.to   = max end
323             if e.from < min then e.from = min end
324             print("Drawing event on line ", e.inputlineno)
325             d = search_array(daysE, e.day) - 1
326             tex.print(string.format(
327                 [[\node[defStyle,text width=-%fem+%f%s/%d, text depth=%fcm-%fem, text height=
328                 2*red, -- text width
329                 e.scale_width, -- text width
330                 textwidth,
331                 #days, -- text width
332                 length*(e.to-e.from)/(max-min), -- text depth
```

```
                2*red+red_y, -- text depth
                red_y, -- text height
                e.tikz, -- free tikz code
                (d+e.offset)/#days, -- xcoord
                textwidth,
                minuteToFrac(e.from,min,max)*-length, -- ycoord
                e.content -- content
                )
            )
        end
    end
    tex.print([[\end{tikzpicture}]])
end
```

`searchArray` Searches an array for a given value and returns the index if found. On error `nil` is returned

```
function search_array(t, s)
    for k,v in ipairs(t) do
        if(v == s) then return k end
    end
    return nil
end

```

`minuteToFrac` Calculates at which fraction of the total duration of `max-min` the time `minute` is located

```
function minuteToFrac(minute, min, max)
    return (minute-min)/(max-min)
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)

```
function prepareMinMax(min, max)
    local minH = math.floor(min/60)
    local maxH = math.ceil(max/60)
    local min = minH*60
    local max = maxH*60
    return min, minH, max, maxH
end
```

`checkKeys` Checks if all `k`s are present in table `t`

```
function checkKeys(t, k)
    for _,x in ipairs(k) do
        if(t[x] == nil) then
            return false
        end
    end
    return true
end
```

`dur2Int` Takes a clock duration formatted as `HH:MM-HH:MM`, splits it, checks for validity and returns begin/end time in minutes

```
function dur2Int(clk)
    local f1,f2, t1,t2 = clk:match("^(%d%d?):(%d%d)-(%d%d?):(%d%d)$")
    if(f1 ~= nil and f2 ~= nil and t1 ~= nil and t2 ~= nil) then
        f1 = tonumber(f1) f2 = tonumber(f2)
        t1 = tonumber(t1) t2 = tonumber(t2)
        assert(f1 >= 0 and f1 < 24, "Hours have to be >= 0 && < 24")
        assert(f2 >= 0 and f2 < 60, "Mins have to be >= 0 && < 60")
        assert(t1 >= 0 and t1 < 24, "Hours have to be >= 0 && < 24")
        assert(t2 >= 0 and t2 < 60, "Mins have to be >= 0 && < 60")
        return f1*60 + f2, t1*60 + t2
    else
        error("clk string \"" .. clk .. "\" was no valid clock string")
```

```
383     end
384 end
```

prepareDays Splits the comma-sep string `days` into an array

```
385 function prepareDays(days)
386     local ret = {}
387     for m in days:gmatch("[^,]+") do
388         table.insert(ret, m)
389     end
390     return ret
391 end
```

copyArray Returns a copy of the table `obj`

```
392
393 function copy_array(obj)
394     if type(obj) ~= 'table' then return obj end
395     local res = {}
396     for k, v in pairs(obj) do
397         local c = copy_array(v)
398         res[copy_array(k)] = c
399     end
400     return res
401 end
```

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

```
402
403 semesterplannerLua = {
404     init = init,
405     addEvent = addEvent,
406     draw = draw
407 }
408 return semesterplannerLua
409 ⟨/luaTimetable⟩
```

2.3 semesterplanner-lua-calendar.lua

TODO how to set the paths right in this case Include the date module for time date calculations

```
410 ⟨*luaApp⟩
411 package.path='/usr/share/lua/5.3/?.lua;/usr/share/lua/5.3/?/init.lua;/usr/lib/lua/5.3/?.lua;/
412 package.cpath='/usr/lib/lua/5.3/?.so;/usr/lib/lua/5.3/loadall.so;./?.so;/home/lukas/.luarocks/
413
414 local dateLib = require "date"
```

init Initialize the EVENTS table as some sort of a reset, takes an argument wethet the reset should be executed (to enable concatenation)

```
415 function init(clear)
416     -- clean up first
417     -- global variable
418     if clear then
419         EVENTS = {}
420     end
421 end
422
423 function genDot(opts)
424     dot = ""
425     if opts.draw then
426         dot = string.format([[\tikz[baseline=(X.base)]\node (X) [fill opacity=.5,fill=red,circ
427     end
428     return dot
429 end
430
```

`addEvent` Adds an event to the list, stores the date and how the event should be highlighted (tikz code for a node)

```
function addEvent(opts)
    opts.inputlineno = tex.inputlineno
print(string.format("collecting from line %d", opts.inputlineno))
    if opts.draw then
        assert(opts.date ~= nil and opts.tikz ~= nil, "date and tikz has to be given")
        if opts.endDate == nil or opts.endDate == '' then
            table.insert(EVENTS, {date=dateLib(opts.date), tikz=opts.tikz, period=opts.period
        else
            table.insert(EVENTS, {date=dateLib(opts.date), tikz=opts.tikz, period=opts.period
        end
    end
end

function addAppointment(opts)
    addEvent(opts)
    dot = genDot(opts)
    tex.print(string.format([[ \textit{%s} & %s & %s%s & %s & %s & %s\\]], opts.date, opts.ti
end

function addExam(opts)
    addEvent(opts)
    dot = genDot(opts)
    tex.print(string.format([[ \textit{%s} & %s & %s%s & %s & %s \\]], opts.date, opts.time, 
end

function addDeadline(opts)
    addEvent(opts)
    dot = genDot(opts)
    tex.print(string.format([[ \textit{%s} & %s%s & %s & %s \\]], opts.date, dot, opts.course
end
```

`drawCalendar` Draw the calendar month by month in a matrix with given columns. The calendar starts and ends at the given dates (in YYYY-MM-DD or any other format the datelib understands)

```

function drawCalendar(minDate, maxDate, cols)
    minDate = dateLib(minDate)
    maxDate = dateLib(maxDate)
    tex.print([[\begin{tikzpicture}[every calendar/.style={inner sep=2pt, week list, month la
}}] ]])
    tex.print([[\matrix[column sep=1em, row sep=1em]{]])
        local i = 1
        running = true
        while running do
            -- derive end from start, then check if maxDate is reached
            endDate = minDate:copy():addmonths(1):setday(1):adddays(-1)
            if endDate >= maxDate then
                endDate = maxDate
                running = false
            end
            tex.print(string.format(
            [[\calendar (%04d-%02d) [dates=%04d-%02d-%02d to %04d-%02d-%02d] if (Sunday) [red
\month-\day) [nodes={inner sep=.4em,rectangle,line width=1pt,draw}] if (at least=\year-
\month-\day) {} else [nodes={strike out, draw}]; ]],
                    minDate:getyear(), minDate:getmonth(), minDate:getyear(), minDate:getmonth

            minDate:addmonths(1)
            minDate:setday(1)

            if i % cols == 0 or not running then
                tex.print([[\\]])
```

```
            else
                tex.print([[&]])
            end
            i = i + 1
        end
        tex.print([[ }; ]])
```

Draw highlighting on a background layer so that the calendar is not overdrawn

```
        local usedDates = {}
        tex.print([[\begin{scope}[on background layer] ]])
        for i,ele in ipairs(EVENTS) do
            print(string.format("Drawing item from line %d", ele.inputlineno))
            while ele.date <= maxDate and (ele.endDate == nil or ele.date <= ele.endDate) do
                local xshift = 0
                if usedDates[tostring(ele.date)] ~= nil then
                    xshift = math.ceil(usedDates[tostring(ele.date)] / 2)
                    if usedDates[tostring(ele.date)] % 2 == 0 then
                        xshift = -xshift
                    end
                    usedDates[tostring(ele.date)] = usedDates[tostring(ele.date)] + 1
                else
                    usedDates[tostring(ele.date)] = 1
                end
                tex.print(string.format([[\node[xshift=%d mm, fill opacity=.5,fill=red,circle
  %02d-%04d-%02d-%02d) {};]],
                    xshift, ele.tikz, ele.date:getyear(), ele.date:getmonth(), ele.date:getyea
                if ele.period == nil then break end
                ele.date:adddays(ele.period)
            end
        end
        tex.print([[\end{scope}]])
    tex.print([[\end{tikzpicture}]])
end
```

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

```

semesterplannerLuaCal = {
    init = init,
    addAppointment = addAppointment,
    addDeadline = addDeadline,
    addExam = addExam,
    drawCalendar = drawCalendar,
}
return semesterplannerLuaCal
```
⟨/luaApp⟩

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.