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.

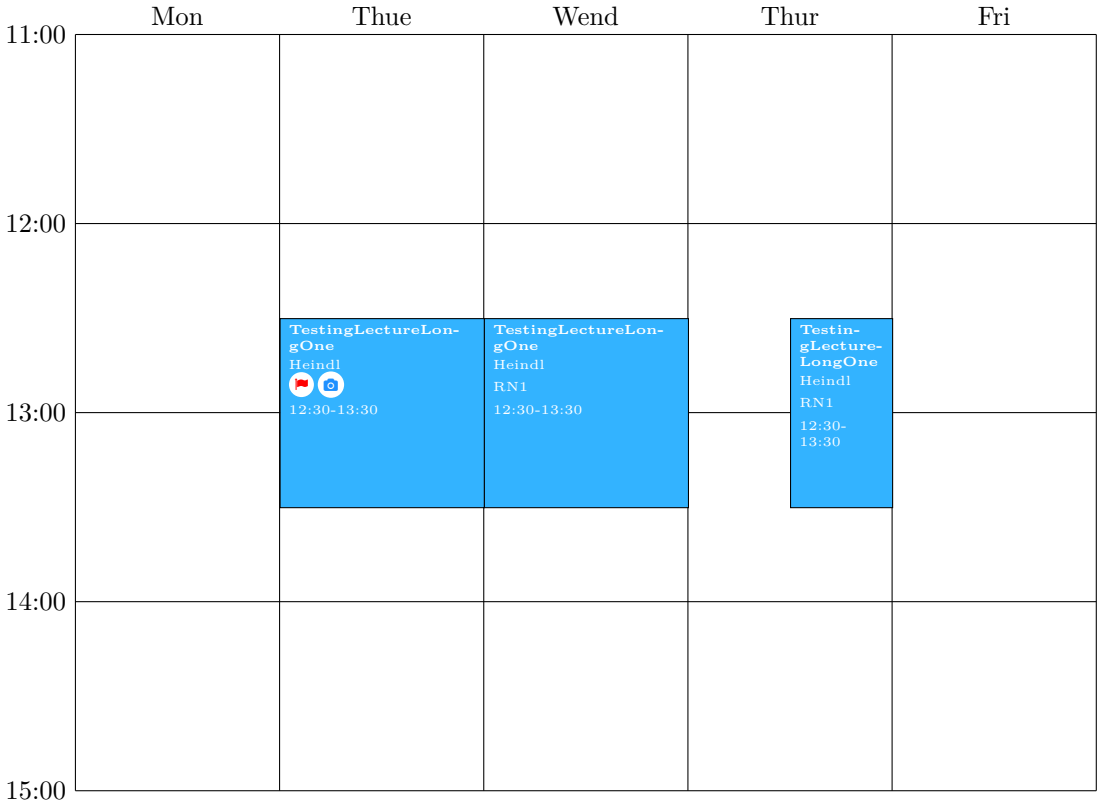
Same goes for the **width** parameter regarding the labels containing the time on the right. Since in this case any tex-lenght is allowed, you can simply try to subtract the length of the clock label using something like **\settowidth{\length}{12:30}** to set a length to the length of a clock label and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

```
40        /semesterplanner-lua/timetable/event/.cd,
41        % event arguments
42        content/.initial=, content/.default=,
43        %
44        time/.initial=, time/.default=,
45        day/.initial=, day/.default=,
46        %
47        tikz/.initial=, tikz/.default=,
48        scale width/.initial=1, scale width/.default=1,
49        offset/.initial=0, offset/.default=0,
50        %
51        textcolor/.initial=, textcolor/.default=,
52        title/.initial=, title/.default=,
53        speaker/.initial=, speaker/.default=,
54        location/.initial=, location/.default=,
55        prio/.initial=, prio/.default=,
56        formatter/.initial=timetableformatter, formatter/.default=timetableformatter,
57        %
```

calendar/:

draw

room

prio

course

desc

start

end

tikz

period

shift

print Only makes sence if the command is suffixed by a % otherwise somehow a space gets inserted (eventhough the % is inserted from lua as well

```
58        /semesterplanner-lua/calendar/.cd,
59        draw/.initial={true}, draw/.default={true},
60        room/.initial={}, room/.default={},
61        time/.initial={}, time/.default={},
62        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},
        shift/.initial={true}, shift/.default={true},
        print/.initial={true}, print/.default={true},
    }
```

2.2 Tikz Calendar add weekday labels

```
\tikzoption{day headings}{\tikzstyle{day heading}=[#1]}
\tikzstyle{day heading}=[]
\tikzstyle{day letter headings}=[
    execute before day scope={ \ifdate{day of month=1}{%
      \pgfmathsetlength{\pgf@ya}{\tikz@lib@cal@yshift}%
      \pgfmathsetlength\pgf@xa{\tikz@lib@cal@xshift}%
      \pgftransformyshift{-\pgf@ya}
      \foreach \d/\l in {0/M,1/T,2/W,3/T,4/F,5/S,6/S} {
        \pgf@xa=\d\pgf@xa%
        \pgftransformxshift{\pgf@xa}%
        \pgftransformyshift{\pgf@ya}%
        \node[every day,day heading]{\l};%
      }
    }{}%
  }%
]
```

2.2.1 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*{\faClock0~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]
        \ignorespaces
    }
```

\seminar

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

\tutorial

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

\meeting

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

\officehour

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

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},
            shift=\pgfkeysvalueof{/semesterplanner-lua/calendar/shift},
            print=\pgfkeysvalueof{/semesterplanner-lua/calendar/print}}}%
        \ignorespaces
        }
    \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}]],
                shift=\pgfkeysvalueof{/semesterplanner-lua/calendar/shift},
                print=\pgfkeysvalueof{/semesterplanner-lua/calendar/print}}}%
        \ignorespaces
        }
    \section*{\faStickyNoteO~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}]],
                shift=\pgfkeysvalueof{/semesterplanner-lua/calendar/shift},
                print=\pgfkeysvalueof{/semesterplanner-lua/calendar/print}}}%
        \ignorespaces
        }
    \section*{\faStickyNoteO~Deadlines}
    \begin{tabular}{rlll}
        \textbf{Date}&\textbf{Course}&\textbf{Description}&\textbf{Prio}\\
```

```
}{
    \end{tabular}
}
```

⟨/package⟩

2.3 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
    else
        assert(max:match("^%d+"), "end time has to be an integer representing the HH*60+MM of
        MAX = tonumber(max)
        MAX_BYPASS = true
    end
end

function defaultFormatter(opts)
    ret = ""
    for k,v in pairs(opts) do
        if type(k) == "string" then k = k:gsub("[_^]", "") end
        if type(v) == "string" then v = v:gsub("[_^]", "") end
        ret = string.format("%s, %s: %s", ret, tostring(k), tostring(v))
    end
    print(ret)
    return ret
end

function timetableformatter(opts)
    return string.format(
        [[\textcolor{%s}{\textbf{%s}\\[.2em]\raggedright{%s}\\[0.5em]\raggedright{%s}\raggedr
            opts.textcolor, opts.title, opts.speaker, opts.prio, opts.location, opts.time)
end
```

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

```
-- result are the global variables EVENTS, MIN and MAX
```

```
function addEvent(opts)
    print("Reading event on line ", tex.inputlineno)
    opts.inputlineno = tex.inputlineno
    if(not checkKeys(opts, {"time", "day", "tikz"})) then
        error("missing argument")
    end

    if opts.content == nil then
        if opts.formatter == nil then
            opts.content = defaultFormatter(opts)
        else
            opts.content = opts.formatter(opts)
        end
    end

    opts.from,opts.to = dur2Int(opts.time)

    if(not MIN_BYPASS and opts.from < MIN) then MIN = opts.from end
    if(not MAX_BYPASS and opts.to   > MAX) then MAX = opts.to   end
    assert(opts.from < opts.to, "From has to be before to")

    table.insert(EVENTS, opts)
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.

```
-- parameters are all global variables
function draw(length, width)
    -- copy relevant variables for working on local copies
    local events = copy_array(EVENTS)
    local days = prepareDays(DAYS)
    local daysE = copy_array(DAYSE)
    local min, minH, max, maxH = prepareMinMax(MIN, MAX)

    assert(length:match("%d*%.?%d*"), "Length must be a valid length measured in cm")
    length = tonumber(length)

    textwidth = width

    tex.print([[\begin{tikzpicture}]])
    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

```
    -- print the tabular with the weekday headers
    tex.print(string.format(
        [[\foreach \week [count=\x from 0, evaluate=\x as \y using \x+0.5] in {%s}{ ]],
        table.concat(days, ",")
        )
    )
    tex.print(string.format(
        [[\node[anchor=south] at (\y/%d* %s, 0) {\week};]], #days, textwidth))
    tex.print(string.format(
        [[\draw (\x/%d * %s, 0cm) -- (\x/%d * %s, %dcm);]],
        #days,
        textwidth,
        #days,
        textwidth, -length
        )
    )
    tex.print("}")
    tex.print(string.format(
        [[\draw (%s, 0) -- (%s,%dcm);]],
        textwidth,
        textwidth,
```

```
        -length
        )
    )

    for i=minH,maxH do
        tex.print(string.format(
            [[\node[anchor=east] at (0,%fcm ) {%d:00};]],
            minuteToFrac(i*60,min,max)*-length, i
            )
        )
        tex.print(string.format(
            [[\draw (0,%fcm ) -- (%s,%fcm );]],
            minuteToFrac(i*60,min,max)*-length,
            textwidth,
            minuteToFrac(i*60,min,max)*-length
            )
        )
    end
```

Draw the nodes of the events

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

```
377 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)

```
378 function prepareMinMax(min, max)
379     local minH = math.floor(min/60)
380     local maxH = math.ceil(max/60)
381     local min = minH*60
382     local max = maxH*60
383     return min, minH, max, maxH
384 end
```

`checkKeys` Checks if all `ks` are present in table `t`

```
385 function checkKeys(t, k)
386     for _,x in ipairs(k) do
387         if(t[x] == nil) then
388             return false
389         end
390     end
391     return true
392 end
```

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

```
393 function dur2Int(clk)
394     local f1,f2, t1,t2 = clk:match("^(%d%d?):(%d%d)-(%d%d?):(%d%d)$")
395     if(f1 ~= nil and f2 ~= nil and t1 ~= nil and t2 ~= nil) then
396         f1 = tonumber(f1) f2 = tonumber(f2)
397         t1 = tonumber(t1) t2 = tonumber(t2)
398         assert(f1 >= 0 and f1 < 24, "Hours have to be >= 0 && < 24")
399         assert(f2 >= 0 and f2 < 60, "Mins have to be >= 0 && < 60")
400         assert(t1 >= 0 and t1 < 24, "Hours have to be >= 0 && < 24")
401         assert(t2 >= 0 and t2 < 60, "Mins have to be >= 0 && < 60")
402         return f1*60 + f2, t1*60 + t2
403     else
404         error("clk string \"" .. clk .. "\" was no valid clock string")
405     end
406 end
```

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

```
407 function prepareDays(days)
408     local ret = {}
409     for m in days:gmatch("[^,]+") do
410         table.insert(ret, m)
411     end
412     return ret
413 end
```

`copyArray` Returns a copy of the table `obj`

```
414
415 function copy_array(obj)
416     if type(obj) ~= 'table' then return obj end
417     local res = {}
418     for k, v in pairs(obj) do
419         local c = copy_array(v)
420         res[copy_array(k)] = c
421     end
422     return res
423 end
```

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

```
424
425 semesterplannerLua = {
426     init = init,
427     addEvent = addEvent,
428     draw = draw
429 }
430 return semesterplannerLua
431 ⟨/luaTimetable⟩
```

2.4 semesterplanner-lua-calendar.lua

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

```
432 ⟨*luaApp⟩
433 package.path='/usr/share/lua/5.3/?.lua;/usr/share/lua/5.3/?/init.lua;/usr/lib/lua/5.3/?.lua;/
434 package.cpath='/usr/lib/lua/5.3/?.so;/usr/lib/lua/5.3/loadall.so;./?.so;/home/lukas/.luarocks/
435
436 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)

```
437 function init(clear)
438     -- clean up first
439     -- global variable
440     if clear then
441         EVENTS = {}
442     end
443 end
444
445 text = {
446     print = function(s)
447         -- print("\"" .. s .. "\"")
448         tex.print(s)
449     end
450 }
451
452 function genDot(opts)
453     dot = ""
454     if opts.draw then
455         dot = string.format([[\tikz[baseline=(X.base)]\node (X) [fill opacity=.5,fill=red,circ
456     end
457     return dot
458 end
459
```

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

```
460 function addEvent(opts)
461     opts.inputlineno = tex.inputlineno
462     print(string.format("collecting from line %d", opts.inputlineno))
463     if opts.draw then
464         assert(opts.date ~= nil and opts.tikz ~= nil, "date and tikz has to be given")
465         if opts.endDate == nil or opts.endDate == '' then
466             table.insert(EVENTS, {shift=opts.shift,date=dateLib(opts.date), tikz=opts.tikz, p
467         else
468             table.insert(EVENTS, {shift=opts.shift,date=dateLib(opts.date), tikz=opts.tikz, p
469         end
470     end
471 end
472
473 function addAppointment(opts)
474     addEvent(opts)
475     dot = genDot(opts)
```

```
476     if opts.print then
477         tex.sprint(string.format([[\textit{%s} & %s & %s%s & %s & %s & %s\\]], opts.date, opts
478     else
479         tex.sprint("%")
480     end
481 end
482
483 function addExam(opts)
484     addEvent(opts)
485     dot = genDot(opts)
486     if opts.print then
487         tex.sprint(string.format([[\textit{%s} & %s & %s%s & %s & %s \\]], opts.date, opts.ti
488     else
489         tex.sprint("%")
490     end
491 end
492
493 function addDeadline(opts)
494     addEvent(opts)
495     dot = genDot(opts)
496     if opts.print then
497         tex.sprint(string.format([[\textit{%s} & %s%s & %s & %s \\]], opts.date, dot, opts.co
498     else
499         tex.sprint("%")
500     end
501 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)

```
502
503 function drawCalendar(minDate, maxDate, cols)
504     minDate = dateLib(minDate)
505     maxDate = dateLib(maxDate)
506     text.print([[\begin{tikzpicture}[every calendar/.style={day headings=red!50,day letter hea
    }, every month/.style={yshift=3ex}}] ]])
507     text.print([[\matrix[column sep=1em, row sep=1em]{]])
508         local i = 1
509         running = true
510         while running do
511             -- derive end from start, then check if maxDate is reached
512             endDate = minDate:copy():addmonths(1):setday(1):adddays(-1)
513             if endDate >= maxDate then
514                 endDate = maxDate
515                 running = false
516             end
517             text.print(string.format(
518             [[\calendar (%04d-%02d) [dates=%04d-%02d-%02d to %04d-%02d-%02d] if (Sunday) [red]
    \month-\day) [nodes={inner sep=.25em,rectangle,line width=1pt,draw}] if (at least=\year-
    \month-\day) {} else [nodes={strike out, draw}]; ]],
519 %04d-%02d-%02d] if (Sunday) [red] if (Saturday) [red!50!white]
520 if (equals=\year-\month-\day) [nodes={inner sep=.25em,rectangle,line width=1pt,draw}] if (at l
    \month-\day) {} else [nodes={strike out, draw}]; ]],
521                     minDate:getyear(), minDate:getmonth(), minDate:getyear(), minDate:getmontl
522
523             minDate:addmonths(1)
524             minDate:setday(1)
525
526             if i % cols == 0 or not running then
527                 text.print([[\\]])
528             else
529                 text.print([[&]])
530             end
```

```
            i = i + 1
        end
        text.print([[ }; ]])

```

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

```
        local usedDates = {}
        text.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 ele.shift then
                    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
                end
                text.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
        text.print([[\end{scope}]])
    text.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.