

The mensa-tex class

<https://github.com/SFr682k/mensa-tex>

Sebastian Friedl
sfr682k@t-online.de

2017/09/22

*“I can’t go to a restaurant and order food
because I keep looking at the fonts on the menu”*

— DONALD E. KNUTH —

Abstract

A \LaTeX class for typesetting school cafeteria menus consisting of two lunches (with dessert) and dinner

Contents

Dependencies and other requirements	2
License	2
Call for cooperation	2
1 Using the <code>\documentclass</code> command	3
2 Creating a menu	3
2.1 Setting up the basic information	3
2.2 Adding food	4
2.3 Adding additional information	4
2.3.1 Remarks at the bottom of the page	4
2.3.2 Remarks for additive ingredients	4
2.3.3 Symbols for vegetarian and vegan food	5
2.4 Using fancy colors	5
2.4.1 Ways of changing colors	5
2.4.2 A short note about color selection	5
2.4.3 Using class-defined colors in normal text	5
3 Adding support for other languages	5
3.1 Weekday names	6
3.2 Keywords	6
3.3 Date formats	7
4 A working example	8

Dependencies and other requirements

The `mensa-tex` class requires \LaTeX 2 ϵ and the following packages:

<code>array</code>	<code>datetime2-calc</code>	<code>lmodern</code>
<code>colortbl</code>	<code>geometry</code>	<code>textcomp</code>
<code>datetime2</code>	<code>graphicx</code>	<code>xcolor</code>

License

© 2017 Sebastian Friedl

This work may be distributed and/or modified under the conditions of the \LaTeX Project Public License, either version 1.3c of this license or (at your option) any later version.

The latest version of this license is available at <http://www.latex-project.org/lppl.txt> and version 1.3c or later is part of all distributions of \LaTeX version 2008-05-04 or later.

This work has the LPPL maintenance status ‘maintained’. The current maintainer of this work is Sebastian Friedl.

This work consists of the following files:

- `mensa-tex.cls` and
- `mensa-tex-doc.tex`

Call for cooperation

Please report bugs and other problems as well as suggestions for improvements using the issue tracker on GitHub or sending an email (sfr682k@t-online.de).

1 Using the `\documentclass` command

Using this class is as easy as using the `\documentclass{mensa-tex}` command.

Following class options are available:

`app` Use a layout optimized for small screens using DIN/ISO A6 paper

`en-GB` Use an English localization, British variant (*default*)

`en-US` Use an English localization, American variant

`german` Use a German localization

2 Creating a menu

The following section deals with creating a menu using `mensa-tex`.

All the commands described in this sections are to be used inside the preamble since the menu gets created instantly when using `\begin{document}`.

2.1 Setting up the basic information

The basic information consists of the name of the cafeteria, the institute (or school) it is located at and the image used on the single pages.

It can be set by using the following commands:

`\mensaname` This command is used to declare the name of the cafeteria – maybe something like `\mensaname{Food Corner}` (*default is the plain old boring “Mensa”*).

You may want to change the font size by using arbitrary \TeX font size commands.

`\institute` Sets the name of the institute the cafeteria is located at – for example, you can insert `\institute{University of LOL}` in your preamble if your cafeteria is located at some institute called the “University of LOL”.

Note that information about the institute is only printed when using the normal layout.

`\setimage` Add an image to your diet plans using this command.

Note that you *have* to declare the image by using the `\includegraphics` inside `\setimage` (e.g. `\setimage{\includegraphics[width=\linewidth]{path/to/picture}}`).

The space available for the image depends on the used layout (see table 1).

	normal layout	app layout
width	$.50\text{\textwidth}$ $\approx 9.0\text{cm}$	$.58\text{\textwidth}$ $\approx 4.9\text{cm}$
height	$.15\text{\textheight}$ $\approx 4.0\text{cm}$	$.15\text{\textheight}$ $\approx 1.9\text{cm}$

Table 1: Available space for the header image

2.2 Adding food

For adding information about the food, the following six commands are provided.

`\startdate` This command defines the “start date”, the date of the first entry (*the start date’s week-day happens to be a Monday*). The start date has to be present in YYYY-MM-DD format, e. g. `\startdate{2007-01-01}` (Default is 2001-01-01).

`\monday` Use these five commands to insert food into the empty diet.

`\tuesday` Every single command requires the same four arguments:

`\wednesday` 1. Menu I 2. Menu II 3. Dessert 4. Lunch

`\thursday`

`\friday` For example, to obtain Monday’s menu consisting of

Menu I:	Fish and chips	Dessert:	Chocolate fudge
Menu II:	Crispy fried chicken	Lunch:	DIY hamburgers

you have to write

```
\monday{Fish and chips}%      <-- % is required when
      {Crispy fried chicken}%  commands are continued
      {Chocolate fudge}%       in the following line
      {DIY hamburgers}
```

It is possible to insert the command listed above without line breaks, however, doing so will result in the source being less human-readable.

Due to the menu being implemented in a tabular environment, you have to use `\newline` instead of `\\` to produce additional lines.

Possible, additional hyphenations not found by \TeX can be marked by manually inserting discretionary hyphens (`\-`) (e. g. `hyphen\-`ation).

2.3 Adding additional information

2.3.1 Remarks at the bottom of the page

Some people may want to insert some remarks or annotation at the bottom of the page. These remarks are defined using two commands, depending on the used layout.

`\longremarks` This command defines the remarks used for the normal layout.

They may be quite short (like `\longremarks{Try it!}`) or very, very long.

`\shortremarks` This command defines the remarks used for the app-Layout
Maybe, you should reduce the font size by inserting arbitrary \TeX font size commands (e. g. `\shortremarks{\footnotesize Now using a smaller font size}`)

2.3.2 Remarks for additive ingredients

When food contains additive ingredients, these additive ingredients are usually indicated by adding superscript figures.



You have to insert the legend manually by using remarks (see section 2.3.1)

`\sup` This command is a shortcut providing access to the `\textsuperscript` command. It can be used to produce a superscript 4 with `\sup{4}`.

2.3.3 Symbols for vegetarian and vegan food

Due to the recent development I decided to declare symbols for labeling vegetarian and vegan food.

However, you have to insert the legend manually by using remarks (see section 2.3.1)

- `\vgt` Produces a symbol consisting of one green leaf () for labeling vegetarian food
- `\vgn` Produces a symbol consisting of two green leaves () for labeling vegan food

2.4 Using fancy colors

To avoid a very boring look of the menu, some parts can be typeset in certain colors

The class defines three colors: one for the background of certain boxes, another one for the text inside the boxes and a third one for structure text.

2.4.1 Ways of changing colors

The colors are changed globally by using three commands. The colors themselves are described using the normal syntax of the `xcolor` package.

- `\setbgcolor` Changes the background color of certain boxes (e. g. `\setbgcolor{blue}`)
- `\setcolorfg` Changes the color of text inside colored boxes (e. g. `\setcolorfg{white}`)
- `\setctextcolor` Changes the color of structure text (e. g. `\setctextcolor{red}`)

2.4.2 A short note about color selection

Please ensure, that the colors set with `\setbgcolor` and `\setcolorfg` are distinctive enough from each other. `\setbgcolor{black}` and `\setcolorfg{black!85}` are usually a very, very bad choice.

Also, a clear difference between `\setctextcolor` and the background of the sheet of paper is recommended.

2.4.3 Using class-defined colors in normal text

The colors defined with `\setbgcolor`, `\setcolorfg` and `\setctextcolor` can be used with the `\color` and other color commands.

The corresponding defined colors are called `\bgcolor`, `\colorfg` and `\ctextcolor`.

Examples: `\color{\bgcolor}`, `\color{\colorfg}` and `\color{\ctextcolor}`

3 Adding support for other languages

Currently, the class natively supports English (GB/US) and German.

However, with the instructions in this section, you are able to define additional localizations yourself.

It is recommended to select the localization closest to your localization by using the class options described in section 1 *before* redefining commands.

Code described in this section is to be placed *before* the `\begin{document}` command.

3.1 Weekday names

Weekday names are stored in the `\wdayname` and `\swdayname` command.

To modify them, copy the code printed below into your preamble and replace the English weekday names (and their abbreviations) with the appropriate form of your localization (but leave the `%s` untouched)

```
% Weekday names
\renewcommand{\wdayname}[1]{%
  \ifcase\DTMfetchdow{#1}
    Monday%
  \or
    Tuesday%
  \or
    Wednesday%
  \or
    Thursday%
  \or
    Friday%
  \fi
}

% Short weekday names
\renewcommand{\swdayname}[1]{%
  \ifcase\DTMfetchdow{#1}
    Mon%
  \or
    Tue%
  \or
    Wed%
  \or
    Thu%
  \or
    Fri%
  \fi
}
```

3.2 Keywords

There are four keywords stored in separate commands.

To modify them, copy the code printed below into your preamble and replace the English words by vocabulary appropriate for your localization.

```
\def\menuname{Menu}
\def\dessertname{Dessert}
\def\dinnername{Dinner}
\def\dietname{Diet for}
```

3.3 Date formats

This part is probably the most complicated one when defining own localizations. Basically, there are four commands that have to be redefined:

- `\shortdate`
The short form of the date (e. g. 10/11)
- `\dowshortdate`
The short form of the date, including the weekday (e. g. Sat., 10/11)
- `\longdate`
The long form of the date (e. g. 10/11/2012)
- `\daterange`
A range between two dates (e. g. 10/11–14/11/2012)

When redefining these commands, you have to assemble the templates available for day, month and year in an order matching the localization.

Date templates

- DD** `\DTMtwodigits{\DTMfetchday{#1}}`
Prints the day using two digits
If the day consists of only one digit, a zero is inserted (e. g. 01 instead of 1)
- D** `\DTMfetchday{#1}`
Prints the day using one or two digits

Month templates

- MM** `\DTMtwodigits{\DTMfetchmonth{#1}}`
Prints the month using two digits
- M** `\DTMfetchmonth{#1}`
Prints the month using one or two digits

Year templates

- YYYY** `\DTMfetchyear{#1}`
Prints the year using as many digits as required
- YY** `\DTMtwodigits{\DTMfetchyear{#1}}`
Prints the year using two digits

Inserting weekday names

- “normal” weekday names:** `\wdayname{#1}`
- short weekday names:** `\swdayname{\DTMfetchdow{#1}}`

Redefining the commands using the templates

Now, you only have to redefine the commands. Use the following basic structure:

```
\renewcommand{\shortdate}[1]{%  
    %% INSERT THE TEMPLATE COMBINATIONS HERE %%  
}
```

The templates listed above can be combined suitable.

For example, if you want the long date to be displayed in the YYYY-MM-DD format, the following code does the trick:

```
\renewcommand{\longdate}[1]{%  
    \DTMfetchyear{#1}%           The YYYY template  
    -%                           Year/month separator  
    \DTMtwodigits{\DTMfetchmonth{#1}}% The MM template  
    -%                           Month/day separator  
    \DTMtwodigits{\DTMfetchday{#1}}% The DD template  
}
```

The same principle applies on redefining `\shortdate`, `\dowshortdate` and `\daterange`. The %s avoid spaces after templates and separators when inserting line breaks.

4 A working example