The luatodonotes package*

Fabian Lipp†
fabian.lipp@gmx.de

February 16, 2020

Abstract

The `luatodonotes` package allows you to insert to–do items in your document. At any point in the document a list of all the inserted to–do items can be listed with the `\listoftodos` command.

It is an extended version of the `todonotes` package and uses more advanced algorithms to place the to–do notes on the page. For this algorithms it depends on LuaTeX.

Contents

*This document corresponds to luatodonotes v0.5, dated 2020/02/16.

†This documentation and the whole package is based on version 1.0.2 of the todonotes package by Henrik Skov Midtiby.

1 Introduction

The `luatodonotes` package makes three commands available to the user: `\todo[]{}`, `\missingfigure{}` and `\listoftodos`. `\todo[]{}` and `\missingfigure{}` makes it possible to insert notes in your document about things that has to be done later (todonotes …). This package is based on version 1.0.2 of `todonotes`[1] by Henrik Skov Midtiby.

The positions of the notes on the page is determined using algorithms implemented in Lua, so you have to process your documents using LuaLaTeX. The package can be used as a drop-in replacement for the original `todonotes` package, you only need to modify `\usepackage{todonotes}` to `\usepackage{luatodonotes}`. Note that `todonotes` and `luatodonotes` must not be loaded inside the same document.

Some alternatives for the luatodonotes package are:

- easy-todo
  Depends on `color`, `tocloft` and `ifthen`, small feature set.
- fixmetodonotes
  Depends on `graphicx`, `color`, `transparent`, `watermark`, `fix-cm`, `ulem` and `tocloft`, small feature set.
- todo
  Depends on `amssymb`, medium feature set.
- fixme
  Large package with a lot of features.
- todonotes

Compared to the classical todonotes this package has more advanced algorithms and more configuration options to control the position of the notes on the page. Additionally, we are able to place notes at almost every position on the page, e. g., in floating environments or in footnotes. As a disadvantage luatodonotes requires LuaLaTeX for document processing, so a standard `pdflatex` won't work. Depending on the chosen layout for the to–do notes the runtime can be much higher than with todonotes. Labels placed by luatodonotes can conflict with text placed with `\marginpar`.

The main reason for considering other packages is that the todonotes package is quite large and relies heavily on tikz. This can slow down compilation of large documents significantly. The mentioned alternatives have a different feature set and do not rely on tikz, which makes them require less ressoureces.

1.1 Using LuaLaTeX

It is quite easy to switch from `pdflatex` to `lualatex`. You only have to load a few different packages. A small guide can be found in the LuaLaTeX guide[2].

[1] `http://www.ctan.org/pkg/todonotes`
[2] `http://mirror.ctan.org/info/luatex/lualatex-doc/lualatex-doc.pdf`

The LuaTEX processor (the `lualatex` executable) should be included in all modern TEX distributions, so you do not need to install additional software. You simply have to run `lualatex` instead of `pdflatex` (or instead of `latex`, `xelatex`).

1.2 Usage of luatodonotes

The package is loaded with `\usepackage[`⟨*options*⟩`]{luatodonotes}`. Valid options are described in Section 1.3. Note that `todonotes` must *not* be loaded. You have to use `lualatex` to process your document, `pdflatex` will not work. The package depends on positions written to the aux-file, so you have to run `lualatex` twice or even three times to get the labels and leaders for the notes right.

`\todo` My most common usage of the todonotes package, is to insert an todonotes somewhere in a latex document. An example of this usage is the command

`\todo{Make a cake \ldots}`,

which renders like. The `\todo` command has this structure: `\todo[`⟨*options*⟩`]{`⟨*todo text*⟩`}`. The `todo text` is the text that will be shown in the todonote and in the list of todos. The optional argument `options`, allows the user to customize the appearance of the inserted todonotes. For a description of all the options see section 1.4.

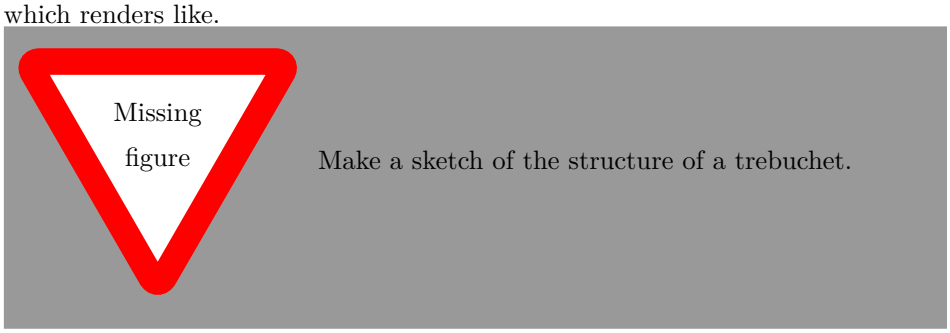
Make a cake …

`\todoarea` The `\todoarea` is similiar to `\todo`, but is able to highlight a specified area in the text, to which the note is connected. The command has this structure: `\todoarea[`⟨*options*⟩`]{`⟨*note text*⟩`}{`⟨*highlighted text*⟩`}`. This command was not tested extensively until now, so it should be used with caution.

`\missingfigure` The `\missingfigure` command inserts an image containing an attention sign and the given text. The command takes only one argument `\missingfigure{`⟨*text*⟩`}`, a text string that could describe what the figure should consist of. An example of its usage could be

`\missingfigure{Make a sketch of the structure of a trebuchet.}`

which renders like.



`\listoftodos` The `\listoftodos` command inserts a list of all the todos in the current document. `\listoftodos` takes no arguments. For this document the list of to–do's looks like.

Todo list

`\todototoc` The `\todototoc` command adds an entry to the table of contents for list of todos. The command should be placed right before the `\listoftodos` command.

1.3 Package options

`disable` If the option `disable` is passed to the package, the macros usually defined by the package (`\todo`, `\todoarea`, `\listoftodos` and `\missingfigure`) are defined as macros with no effect, and thus all inserted notes are removed.

`obeyDraft, obeyFinal` When the option `obeyDraft` is given, the package checks if the one of the options `draft`, `draftcls` or `draftclsnofoot` is given (this option is usually given to the documentclass). If the `draft` option is given, the functionality of the package is enabled and otherwise the effect of the package is disabled. The option `obeyFinal` does something similar, except that the todonotes package is only disabled if the `final` option given.

`danish, german, ngerman, english, french, swedish spanish, catalan, italian portuguese, dutch, croatian` Use translations of the text strings "List of todos" and "Missing figure". The default is to use none of these options, which results in english text strings. Currently the following languages are supported: catalan, croatian, danish, dutch, english, french, german, ngerman, italian, portuguese, spanish and swedish.

`colorinlistoftodos` Adds a small colored square in front of all items in the Todo list. The color of the square is the same as the fill color of the inserted todonote. This can be useful if there are different types of todos (insert reference, explain in detail, ...) where the color of the inserted todonote marks the type of todo.

`color backgroundcolor linecolor bordercolor` These options sets the default colors for the todo command. There is three colors that can be specified. The border color (default `bordercolor=black`) around

the inserted text, the color behind the inserted text (default `backgroundcolor=orange`) and the color of the line connecting the inserted textbox with the current location in the text (default `linecolor=black!30`). Setting the `color` option to `val` passes this value on to the background and line color options. The specified colors must be valid according to the `xcolor` package.

`textsize` `textsize=value` sets the default text size of the inserted todonotes to the given value. Value is the "name" of the used font size, eg. if the desired fontsize is `\tiny` use `textsize=tiny`. The default value is `textsize=normalsize`.

`prependcaption` The `prependcaption` option triggers a special behaviour of the `caption=val` option for the todo command, where the given value `val` is inserted in the inserted todonote.

`shadow` If the `shadow` option is given, the inserted todonotes will be displayed with a gray shadow. I expect that the option will trigger problems with tikz versions prior to 2.0.

`figwidth` `figheight` The `figwidth=length` option and `figheight=length` option set the default width and height of the figure inserted by the `\missingfigure` command. The default value is `\linewidth` for the width and `4cm` for the height.

`leaderwidth` The `leaderwidth=length` option specifies the width of the leader lines. The argument is passed to the `line width` option in TikZ. The default value is `1.6pt`.

`leadertype` The `leadertype=type` option specifies the shape of the leaders, which are drawn between the labels in the margin and the corresponding sites in text. We use the characterization of the leader types known from boundary labeling: p denotes a segment parallel to the left/right side of the text area, while o denotes a orthogonal segment. s is a straight-line segment. The following types are available (`opo` is the default value):

- `s`: Straight-line connection between site and label.
- `sBezier`: Uses the straight-line leaders but transforms them into Bézier curves, which are easier to follow for the reader. The generated curves don't cross each other when the straight-line leaders are crossing-free.
- `opo`: This is the style used in the original todonotes package. The leaders start with a horizontal segment at the site in the text, followed by a vertical segment in the margin beneath the text. The last segment is a vertical segment, which connects to the label.
- `os`: This is the style used in common word processing applications like LibreOffice. The leader also starts with a horizontal segment that leads to the margin and is connected to the label by a straight line.
- `po`: The leader starts with a vertical segment at the site in text and is then connected to the label by a horizontal segment.

`positioning` The `positioning=algorithm` option specifies, which algorithm is used to determine the positions of the notes on the page. You should choose the algorithm depending on the leader type you want to use. You can also use one of the options `s`, `po`, `bezier`, or `opo` to define the positioning algorithm together with the

leadertype. The default value for this option is `inputOrderStacks`. The following algorithms are available:

- `inputOrder`: Place the labels in the order given by the y-coordinates of the corresponding sites in text. Intended for use with *opo*- or *os*-leaders.
- `inputOrderStacks`: Like the algorithm before, but the labels are clustered before they are placed. Thus the labels are placed nearer to their sites. Intended for use with *opo*- or *os*-leaders.
- `sLeaderNorthEast`: Places labels in a way that they can be connected to their sites by straight-line leaders without crossings. The leaders are attached to the upper right or upper left corner of the label (depending on which site of the text the label is placed). Intended for use with *s*-leaders or Bézier leaders.
- `sLeaderNorthEastBelow`: Like the algorithm before, but the leader is attached to a point that is a constant offset below the corner of the label. Intended for use with *s*-leaders or Bézier leaders.
- `sLeaderNorthEastBelowStacks`: Like the algorithm before, but the labels are cluster before they are placed. Thus the labels are placed nearer to their sites. Intended for use with *s*-leaders or Bézier leaders.
- `sLeaderEast`: Like the algorithms before, but the leader is attached to the center of the right or left boundary of the label. Intended for use with *s*-leaders or Bézier leaders.
- `poLeaders`: Calculates label positions that lead to *po*-leaders with minimum total length. This algorithm depends heavily on the number of notes, so the runtime and memory consumption can get very high.
- `poLeadersAvoidLines`: Like the algorithm before, but tries to avoid overlapping of horizontal leader segments with text. This algorithm depends heavily on advanced LuaTEX features to manipulate the data structures of the page, so it possibly could give conflicts with other packages.

`s`
`bezier`
`opo`
`po`

Shorthand options for convenience, which represent common combinations of leadertypes and postioning algorithms. `leadertype` or `positioning` options following one of these options override its settings. They use the following positioning algorithms:

- `s`: `sLeaderNorthEastBelowStacks`
- `bezier`: `sLeaderNorthEastBelowStacks`
- `opo`: `inputOrderStacks`
- `po`: `poLeadersAvoidLines`

`splitting` The `splitting=algorithm` option can be used to place the labels on both sides of the text. The notes are only separated when there is enough space on both sides (see `minNoteWidth`. The default value for this option is `none`. Available algorithms for this option are:

- `none`: Labels are placed in the wider margin only.
- `middle`: The text area is split in the middle in a left and a right half. Labels, whose sites are in the left half of the text, are placed in the left margin, the others in the right margin.
- `median`: The notes are seperated at the median of the sites (sorted by x-coordinate). That is, the number of notes in the left and the right margin is equal (except for one note).
- `weightedMedian`: Considers the height of the labels for the median. So the total height of the labels in the left margin is approximately equal to that in the right margin.

`interNoteSpace` The `interNoteSpace=length` option specifies the minimum vertical distance between two notes. The default value is `5pt`.

`noteInnerSep` The `noteInnerSep=length` option specifies the `inner sep` used for the TikZ nodes, i. e., the distance between the border of the note and the text inside it. The default value is `5pt`.

`routingAreaWidth` The `routingAreaWidth=length` option specifies the width of the so called routing area. This is the area, in which the vertical segment of *opo*-leaders are placed. The area is also used for *os*-leaders. The default value is `0.4cm`.

`minNoteWidth` The `minNoteWidth=length` option specifies the minimum width of the labels. When there is fewer space in one of the margins, this margin is not considered for label placement. If both margins are narrower, no labels are placed and an error message is printed to the console output. The default value of this option is `2.0cm`.

`distanceNotesPageBorder` The `distanceNotesPageBorder=length` option specifies the horizontal distance from the labels to the borders of the paper. You can adjust this setting to your printer margins. The default value of this option is `0.5cm`.

`distanceNotesText` The `distanceNotesPageBorder=length` option specifies the horizontal distance between the labels and the text area. With *opo*- or *os*-leaders the routing area is inserted additionally so the distance between labels and text area increases. The default value of this option is `0.2cm`.

`rasterHeight` The `rasterHeight=length` option is used only for the *po*-leader algorithm. For this algorithm the page is rasterized and the labels are placed only on the positions given by this raster. Decreasing this value can yield better results (i. e., smaller total leader length), but strongly increases the runtime and memory consumption. The default value of this option is `1cm`.

`additionalMargin` The `additionalMargin=length` option extends the page margins horizontally. To achieve this the page width is increased. The page is extended by the given length on both sides. The layout of the page stays the same but the paper format is changed: the height is left unmodified, but the width is increased by the doubled

value of the given length. This option is useful if you have to adhere to a given layout, whose margins are not wide enough to accomodate the notes. You can safely use this option as the final layout of your document does not change when disabling the `luatodonotes` package. The default width of `2cm` for the additional margin is used when the option is given without a length.

`debug` When the `debug` option is activated the package is more verbose on the commandline. Additionally, some markers, which can be used to understand the algorithms, are drawn on the page (depending on the chosen algorithm).

1.4 Options for the `todo` command

There are several options that can be given to the `\todo` command. All the options are described here and often I have included examples of the change in visual appearance. Default values for these options can be set using the presetkeys command.

```
\presetkeys{todonotes}{fancyline, color=blue!30}{}
```

`disable` The `disable` option can be given directly to the todo command. If given the command has no effect.

`color` `backgroundcolor` `linecolor` `bordercolor` These options set the color that is used in the current todo command. The color classes is the same as used in the color package options, see section 1.3. Default values can be set by the color options when the todonotes package is loaded. The todo notes inserted in this paragraph is created with the command `\todo[color=green!40]{And a green note}`. The color of the inserted note could be used to mark different types of tasks (insert references, explain something in detail, ...), this could be streamlined by defining new commands like below.

And a green note

```
\newcommand{\insertref}[1]{\todo[color=green!40]{#1}}
\newcommand{\explainindetail}[1]{\todo[color=red!40]{#1}}
```

An example that uses all of the color options is given below.

Anything but default colors

```
\todo[linecolor=green!70!white, backgroundcolor=blue!20!white,
bordercolor=red]{Anything but default colors}.
```

`line / noline` If you want to get rid of the line connecting the inserted note with the place in the text where the note occurs in the latex code, the option `noline` can be used. `\todo[noline]{A note with no line ...}`

A note with no line connecting it to the placement in the original text.

`inline / noinline` It is possible to place a todonote inside the text instead of placing it in the margin, this could be desirable if the text in the note has a considerable length. `\todo[inline]{A todonote placed in the text}`

A todonote placed in the text

Another usage for the inline option is when you want to add a todonote to a figure caption.

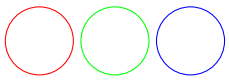


Figure 1: A text explaining the image.

Fill those circles ...

```
\begin{wrapfigure}{r}[20mm]{40mm}
\begin{tikzpicture}
\draw[red] (0, 0) circle(0.45);
\draw[green] (1, 0) circle(0.45);
\draw[blue] (2, 0) circle(0.45);
\end{tikzpicture}
\caption{A text explaining the image.
\todo[inline]{Fill those circles \ldots}}
\end{wrapfigure}
```

`size`

`size=val` changes the size of the text inside the todonote. The commands used to create the notes below are `\todo[size=\Large]{A note with a large font size.}` and `\todo[inline, size=\tiny]{Note with very small font size.}`.

A note with a large font size.

Note with very small font size.

`list / nolist`

When the option `nolist` is given, the todo item will not appear in the list of todos.

`caption`

The `caption` option enables the user to specify a short description of the todonote that are inserted in the list of todos instead of the full todonote text.

A very long and tedious note that cannot be on one line in the list of todos.

```
\todo[caption={Short note}]{A very long and tedious note that
cannot be on one line in the list of todos.}.
```

`prepend / noprepend`

The effect of this option is altered with the package option `prependcaption` or the `prepend / noprepend` option for the todo command.

The options `prepend` and `noprepend` can be used for setting whether a given caption should be prepended to the todonote or not. Globally this can be set using the `prependcaption` option for the package. Below is the effect of the option shown using the code:

Short note with prepend: A very long and tedious note that cannot be on one line in the list of todos.

A very long and tedious note that cannot be on one line in the list of todos.

```
\todo[prepend, caption={Short note with prepend}]{A very long and tedious
note that cannot be on one line in the list of todos.}.
\todo[noprepend, caption={Short note with noprepend}]{A very long and
tedious note that cannot be on one line in the list of todos.}.
```

`author`

The `author` option takes a parameter, the name of the author. The given name is inserted in the todonote.

Xavier: Testing author option.

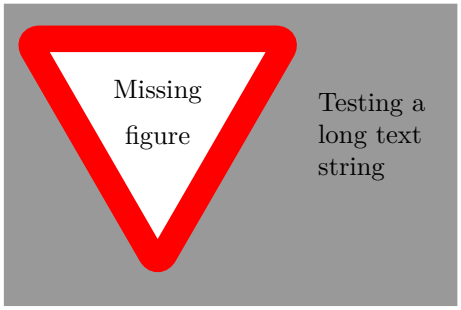
Xavier: Testing author option.

```
\todo[author=Xavier]{Testing author option.}
\todo[author=Xavier, inline]{Testing author option.}
```

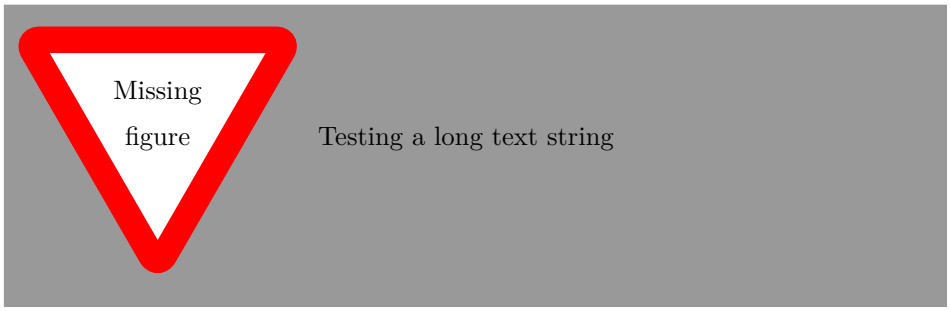
1.5 Options for the `missingfigure` command

`figwidth` The `figwidth=length` option sets the width of the figure inserted by the `\missingfigure` command. Length values below $6cm$ might trigger some problems with the visual appearance. Try to compare the default of the missing figure command, when the option is given or not.

```
\missingfigure[figwidth=6cm]{Testing a long text string}
```

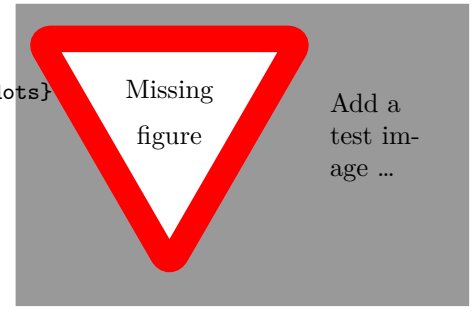



```
\missingfigure{Testing a long text string}
```



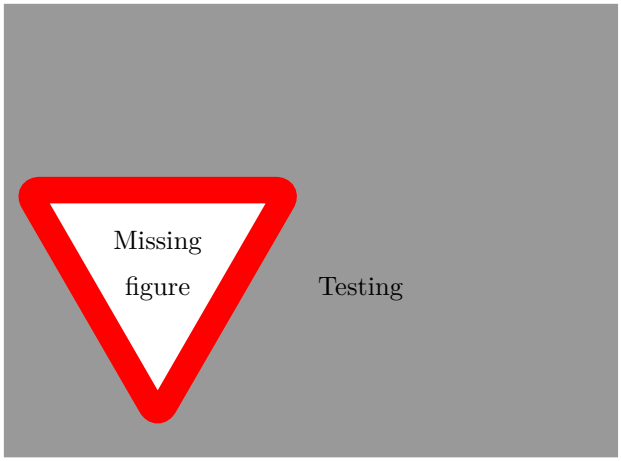
Another usage of the option is when `\missingfigure` is used in the wrapfigure environment.

```
\begin{wrapfigure}{r}[2cm]{6cm}
\missingfigure[figwidth=6cm]{Add a test image \ldots}
\end{wrapfigure}
```



`figheight` The `figheight=length` option changes the height of the inserted missing figure. The default height is 4cm and using values lower than this might cause the warning sign to pop out of the gray area.

```
\missingfigure[figheight=6cm]{Testing a long text string}
```



`figcolor` The `figcolor=color` options sets the background color of inserted missing figures. The default color is `black!40`.

```
\missingfigure[figcolor=white]{Testing figcolor}
```

1.6 Options for the `listoftodos` command

The `\listoftodos` command takes one optional argument, that defines the name of the inserted list of todos.

```
\listoftodos[I can be called anything]
```

1.7 Troubleshooting

1.7.1 Missing Lua files

A potential error message when Lua source files are not found, is the following:

```
! LuaTeX error [\directlua]:1: module 'luatodonotes' not found:
        no field package.preload['luatodonotes']
        [luatexbase.loader] Search failed
```

```
        [kpse lua searcher] file not found: 'luatodonotes'
        [kpse C searcher] file not found: 'luatodonotes'
        [oberdiek.luatex.kpse_module_loader]-eroux Search failed
stack traceback:
        [C]: in function 'require'
        [\directlua]:1: in main chunk.
l.250 \directlua{require("luatodonotes")}
```

This means that the file `luatodonotes.lua` cannot be found by LuaTEX. It depends on the version of your TEX installation in which directories LuaTEX is looking for Lua source files. You can query these paths with the following command:

```
kpsewhich -show-path=lua
```

See the `kpathsea` documentation[3] for the interpretation of this path. The Lua source files of the `luatodonotes` package should be in one of the searched directories. You can modify the path in your TEX configuration or using environment variables. You can query kpathsea for a file using the default TEX search path with:

```
kpsewhich luatodonotes.lua
```

Be sure to run `texhash` (as root if needed) after moving files into the texmf tree.

1.7.2 The `debug` option

You can load the package with the option `debug` (see Section 1.3). It gives some additional information in the console while running LuaTEX and draws additional information into the output document. For example, the size of the computed areas, in which the labels are placed, is shown in the document. Depending on the chosen layout algorithm some intermediate steps of the algorithms are given.

1.8 Known issues

1.8.1 Package loading order

The luatodonotes package requires the following packages:

- ifthen
- xkeyval
- xcolor
- tikz
- graphicx (is loaded via the tikz package)
- luacode
- luatex
- atbegshi
- xstring

[3] http://tug.org/texinfohtml/kpathsea.html

- zref-abspage
- ifoddpage
- soul
- soulpos

When luatodonotes are loaded in the preamble, the package checks if these packages all are loaded. If that is not the case it loads the missing packages with no options given. If you want to give some specific options to some of these packages, you have to load them *before* the luatodonotes package, otherwise you will get an "Option clash" error when latex works on the document.

If both the menukeys and the xcolor (with the option `table`) package should be loaded, the following order must be used.

```
\usepackage[table]{xcolor}
\usepackage{todonotes}
\usepackage{menukeys}
```

1.8.2 Spacing around inserted notes

Inserted todo commands will eat the white space after the command.

```
Testing\todo{Does this eat the space?} testing.
```

Testingtesting.

Does this eat the space?

This can be prevented by adding curly parenthesis after the todo command, like shown below.

```
Testing\todo{Does this eat the space?}{} testing.
```

Testing testing.

Does this eat the space?

1.8.3 Conflicts with the amsart documentclass

The `amsart` document class redefines some internal commands that is used by the todonotes package, this will cause an malfunctioning `\listoftodos` command. The following code to circumvent the problem was given by Dan Luecking on comp.text.tex

```
\makeatletter
  \providecommand\@dotsep{5}
\makeatother
\listoftodos\relax
```

NOT TESTED NOT TESTED NOT TESTED

Dominique suggests the following workaround.

```
\makeatletter
\providecommand\@dotsep{5}
\def\listtodoname{List of Todos}
\def\listoftodos{\@starttoc{tdo}\listtodoname}
\makeatother
```

1.8.4 Unknown option ”remember picture”

If latex throws the error

```
Package tikz Error: I do not know what to do with the option ``remember picture''.
```

It probably means that your latex installation is outdated, as only newer versions of latex driver for tikz supports the `remember picture` option. For additional info consult ”Section 10.2.2 Producing PDF Output” in the tikz manual. http://mirror.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf

1.8.5 List of todo heading is not correctly formatted

If using natbib, the todonotes list title gets screwed up unless you do something like this:

```
\makeatletter\let\chapter\@undefined\makeatother
```

Suggestion by Richard Stanton.

1.8.6 Some commands not working inside notes

Some commands will not work like expected, when used inside of a note. They will cause errors when processing the document or have simply no effect. This is caused by the mechanism used to layout the notes: The content is written into a hbox when a `\todo` is encountered. The contents of this box are then stored until the note is typeset. By that time the contents are taken out of the hbox (by `\unhbox`) and put into a `\parbox` with the width required for the note. I don't have a solution for this problem yet.

2 Implementation

luatodonotes.lua

In this section only the source code of the LaTeX package file (`luatodonotes.sty`) is shown. The Lua code is contained in `luatodonotes.lua` and documented by comments inside this file. These comments are primarily describing technical aspects. Information about the implemented algorithms and some theoretical considerations can be found in the following documents:

- Kindermann, P., Lipp, F., and Wolff, A.: Luatodonotes: Boundary Labeling for Annotations in Texts. In: Duncan, C. and Symvonis, A. (eds.) Proc. 22nd Int. Sympos. Graph Drawing GD'14. LNCS, vol. 8871, pp. 76-88. Springer, Heidelberg (2014) http://dx.doi.org/10.1007/978-3-662-45803-7_7
- Lipp, F.: Boundary Labeling for Annotations in Texts. Master thesis, 2014. http://www1.pub.informatik.uni-wuerzburg.de/pub/theses/2014-lipp-master.pdf

2.1 Dependencies and definitions

Make sure that the classical `todonotes` package is not loaded as we redefine its commands. Additionally we pretend that `todonotes` 1.0.2 is already loaded. So later attempts to load package `todonotes` are simply ignored. Loading both packages in the same document would produce errors (like "Command already defined").

```
\@ifpackageloaded{todonotes}{
  \PackageError{luatodonotes}{%
    Conflicting packages todonotes and luatodonotes\MessageBreak
    loaded. Aborting.}{%
    The package luatodonotes was designed as a replacement for todonotes. So it
    is not possible (and not reasonable) to include both of them in the same
    document.%
    If you want to use luatodonotes you should delete the todonotes
    package from\MessageBreak
    your preamble.\MessageBreak}
}{}
\expandafter\def\csname ver@todonotes.sty\endcsname{2014/07/14}
```

Check if LuaTEX is used.

```
\RequirePackage{ifluatex}
\ifluatex\else
  \PackageError{luatodonotes}{LuaTeX is required for this package. Aborting.}{%
    This package can only be used with the LuaTeX engine\MessageBreak
    (command `lualatex'). Package loading has been stopped\MessageBreak
    to prevent additional errors.}
\fi
```

Loads the packages dependencies.

```
\RequirePackage{ifthen}
```

```
\RequirePackage{xkeyval}
\RequirePackage{xcolor}
\RequirePackage{tikz}
\usetikzlibrary{positioning}
\usetikzlibrary{intersections}
\usetikzlibrary{decorations.pathmorphing}
\RequirePackage{luacode}
\RequirePackage{atbegshi}
\RequirePackage{xstring}
\RequirePackage{zref-abspage}
\RequirePackage{ifoddpage}
\RequirePackage{soul}
\RequirePackage{soulpos}
\RequirePackage{etoolbox}
```

The package `luatex` must not be loaded in new TeX distributions as the definition of `\newattribute` in it conflicts with newer versions of LuaLaTeX. Older versions of `luatexbase` include the package `luatex` by themselves, for newer versions the LuaLaTeXkernel should include the commands that we need (e. g., `\newattribute`).

```
\@ifpackagelater{luatexbase}{2013/05/04}{}{
    \RequirePackage{luatex}
}
```

Some default values are set

```
\newcommand{\@todonotes@text}{}%
\newcommand{\@todonotes@backgroundcolor}{orange}
\newcommand{\@todonotes@linecolor}{black!30}
\newcommand{\@todonotes@bordercolor}{black}
\newcommand{\@todonotes@leaderwidth}{1.6pt}
\newcommand{\@todonotes@textsize}{\normalsize}
\newcommand{\@todonotes@figwidth}{\linewidth}
\newcommand{\@todonotes@figheight}{4cm}
\newcommand{\@todonotes@figcolor}{black!40}
```

Default values for variables added by luatodonotes

```
\newcommand{\@todonotes@positioning}{inputOrderStacks}
\newcommand{\@todonotes@splitting}{none}
\newcommand{\@todonotes@leadertype}{opo}
\newcommand{\@todonotes@interNoteSpace}{5pt}
\newcommand{\@todonotes@noteInnerSep}{5pt}
\newcommand{\@todonotes@routingAreaWidth}{0.4cm}
\newcommand{\@todonotes@minNoteWidth}{2.0cm}
\newcommand{\@todonotes@distanceNotesPageBorder}{0.5cm}
\newcommand{\@todonotes@distanceNotesText}{0.2cm}
\newcommand{\@todonotes@rasterHeight}{1cm}
\newcommand{\@todonotes@additionalMargin}{2cm}
```

```
\AtBeginDocument{
\ifx\undefined\phantomsection
\newcommand{\phantomsection}{}
```

```
\fi
}
```

2.2 Declaration of options for the package

In this part the various options for the package are defined.

Define the default text strings and set localization options for the danish and german languages.

```
\newcommand{\@todonotes@todolistname}{Todo list}
\newcommand{\@todonotes@MissingFigureText}{Figure}
\newcommand{\@todonotes@MissingFigureUp}{Missing}
\newcommand{\@todonotes@MissingFigureDown}{figure}
\newcommand{\@todonotes@SetTodoListName}[1]
    {\renewcommand{\@todonotes@todolistname}{#1}}
\newcommand{\@todonotes@SetMissingFigureText}[1]
    {\renewcommand{\@todonotes@MissingFigureText}{#1}}
\newcommand{\@todonotes@SetMissingFigureUp}[1]
    {\renewcommand{\@todonotes@MissingFigureUp}{#1}}
\newcommand{\@todonotes@SetMissingFigureDown}[1]
    {\renewcommand{\@todonotes@MissingFigureDown}{#1}}
\newif{\if@todonotes@reverseMissingFigureTriangle}
\DeclareOptionX{catalan}{
    \@todonotes@SetTodoListName{Llista de feines pendents}%
    \@todonotes@SetMissingFigureText{Figura}%
    \@todonotes@SetMissingFigureUp{Figura}%
    \@todonotes@SetMissingFigureDown{pendent}%
}
\DeclareOptionX{croatian}{%
   \@todonotes@SetTodoListName{Popis obveza}%
   \@todonotes@SetMissingFigureText{Slika}%
   \@todonotes@SetMissingFigureUp{Nedostaje}%
   \@todonotes@SetMissingFigureDown{slika}%
}
\DeclareOptionX{danish}{%
    \@todonotes@SetTodoListName{G\o{}rem\aa{}lsliste}%
    \@todonotes@SetMissingFigureText{Figur}%
    \@todonotes@SetMissingFigureUp{Manglende}%
    \@todonotes@SetMissingFigureDown{figur}%
}
\DeclareOptionX{dutch}{%
   \@todonotes@SetTodoListName{Lijst van onafgewerkte taken}%
   \@todonotes@SetMissingFigureText{Figuur}%
   \@todonotes@SetMissingFigureUp{Ontbrekende}%
   \@todonotes@SetMissingFigureDown{figuur}%
}
\DeclareOptionX{english}{%
   \@todonotes@SetTodoListName{Todo list}%
   \@todonotes@SetMissingFigureText{Figure}%
   \@todonotes@SetMissingFigureUp{Missing}%
```

```
    \@todonotes@SetMissingFigureDown{figure}%
}
\DeclareOptionX{french}{%
    \@todonotes@SetTodoListName{Liste des points \`a traiter}%
    \@todonotes@SetMissingFigureText{Figure}%
    \@todonotes@SetMissingFigureUp{Figure}%
    \@todonotes@SetMissingFigureDown{manquante}%
    \@todonotes@reverseMissingFigureTrianglefalse
}
\DeclareOptionX{german}{%
    \@todonotes@SetTodoListName{Liste der noch zu erledigenden Punkte}%
    \@todonotes@SetMissingFigureText{Abbildung}%
    \@todonotes@SetMissingFigureUp{Fehlende}%
    \@todonotes@SetMissingFigureDown{Abbildung}%
}
\DeclareOptionX{italian}{
    \@todonotes@SetTodoListName{Elenco delle cose da fare}%
    \@todonotes@SetMissingFigureText{Figura}%
    \@todonotes@SetMissingFigureUp{Figura}%
    \@todonotes@SetMissingFigureDown{mancante}%
}
\DeclareOptionX{ngerman}{%
    \@todonotes@SetTodoListName{Liste der noch zu erledigenden Punkte}%
    \@todonotes@SetMissingFigureText{Abbildung}%
    \@todonotes@SetMissingFigureUp{Fehlende}%
    \@todonotes@SetMissingFigureDown{Abbildung}%
}
\DeclareOptionX{portuguese}{
    \@todonotes@SetTodoListName{Lista de tarefas pendentes}%
    \@todonotes@SetMissingFigureText{Figura}%
    \@todonotes@SetMissingFigureUp{Figura}%
    \@todonotes@SetMissingFigureDown{pendente}%
}
\DeclareOptionX{spanish}{
    \@todonotes@SetTodoListName{Lista de tareas pendientes}%
    \@todonotes@SetMissingFigureText{Figura}%
    \@todonotes@SetMissingFigureUp{Figura}%
    \@todonotes@SetMissingFigureDown{pendiente}%
}
\DeclareOptionX{swedish}{%
    \@todonotes@SetTodoListName{Att g\"{o}ra-lista}%
    \@todonotes@SetMissingFigureText{Figur}%
    \@todonotes@SetMissingFigureUp{Figur}%
    \@todonotes@SetMissingFigureDown{saknas}%
}
```

Create a counter, for storing the number of inserted todos.

```
\newcounter{@todonotes@numberoftodonotes}
```

Create a counter, for storing the number of lines in the current todoarea.

```
\newcounter{@todonotes@numberofLinesInArea}
```

Toggle whether the package should obey the global draft option.

```
\newif{\if@todonotes@obeyDraft}
\DeclareOptionX{obeyDraft}{\@todonotes@obeyDrafttrue}
\newif{\if@todonotes@isDraft}
\DeclareOptionX{draft}{\@todonotes@isDrafttrue}
\DeclareOptionX{draftcls}{\@todonotes@isDrafttrue}
\DeclareOptionX{draftclsnofoot}{\@todonotes@isDrafttrue}
\newif{\if@todonotes@obeyFinal}
\DeclareOptionX{obeyFinal}{\@todonotes@obeyFinaltrue}
\newif{\if@todonotes@isFinal}
\DeclareOptionX{final}{\@todonotes@isFinaltrue}
```

Make it possible to disable the functionality of the package. If this option is given, the commands `\todo{}` and `\listoftodos` are defined as commands with no effect. (But you can still compile you document with these commands).

```
\newif{\if@todonotes@disabled}
\DeclareOptionX{disable}{\@todonotes@disabledtrue}
```

Show small boxes in the list of todos with the color of the inserted todonotes.

```
\newif{\if@todonotes@colorinlistoftodos}
\DeclareOptionX{colorinlistoftodos}{\@todonotes@colorinlistoftodostrue}
```

We only define dvistyle for compatibility with todonotes. The option was intented for use with `tex`, there should be no problems using `luatex`. So we ignore this option and issue a warning.

```
\DeclareOptionX{dvistyle}{\PackageWarningNoLine{luatodonotes}
    {Parameter dvistyle is not supported by luatodonotes.
    Ignoring this option}}
```

Create a color option.

```
\define@key{luatodonotes.sty}%
    {color}{
        \renewcommand{\@todonotes@backgroundcolor}{#1}
        \renewcommand{\@todonotes@linecolor}{#1}}
```

Make the background color of the notes as an option.

```
\define@key{luatodonotes.sty}%
    {backgroundcolor}{\renewcommand{\@todonotes@backgroundcolor}{#1}}
```

Make the line color of the notes as an option.

```
\define@key{luatodonotes.sty}%
    {linecolor}{\renewcommand{\@todonotes@linecolor}{#1}}
```

Make the color of the notes box color as an option.

```
\define@key{luatodonotes.sty}%
    {bordercolor}{\renewcommand{\@todonotes@bordercolor}{#1}}
```

Make the width of the leader line as an option. It is later set as `line width` in TikZ.

```
\define@key{luatodonotes.sty}%
    {leaderwidth}{\renewcommand{\@todonotes@leaderwidth}{#1}}
```

Set whether short captions given as arguments to the todo command should be included in the inserted todonote.

```
\newif{\if@todonotes@prependcaptionglobal}
\@todonotes@prependcaptionglobalfalse
\DeclareOptionX{prependcaption}{\@todonotes@prependcaptionglobaltrue}
```

This option is only there for compatibility with todonotes. We ignore it and issue a warning because the width of our labels is determined dynamically based on the page layout.

```
\define@key{luatodonotes.sty}%
    {textwidth}{\PackageWarningNoLine{luatodonotes}
    {Parameter textwidth is not supported by luatodonotes}}
```

Make the text size as an option. It requires some magic with the `\csname` and `\endcsname` macros, as commands cannot be taken as options for a package.

```
\define@key{luatodonotes.sty}%
    {textsize}{\renewcommand{\@todonotes@textsize}{\csname #1\endcsname}}
```

Add option for shadows behind the inserted notes

```
\newif{\if@todonotes@shadowenabled}
\@todonotes@shadowenabledfalse
\DeclareOptionX{shadow}{\@todonotes@shadowenabledtrue
\usetikzlibrary{shadows}}
```

Add option for the default width of the figure inserted with `\missingfigure`.

```
\define@key{luatodonotes.sty}%
    {figwidth}{\renewcommand{\@todonotes@figwidth}{#1}}
\define@key{luatodonotes.sty}%
    {figheight}{\renewcommand{\@todonotes@figheight}{#1}}
\define@key{luatodonotes.sty}%
    {figcolor}{\renewcommand{\@todonotes@figcolor}{#1}}
```

`s,bezier,opo,po` Provide shorthand options for the most common leader styles.

```
\DeclareOptionX{po}%
    {\setkeys{luatodonotes.sty}{leadertype=po,positioning=poLeadersAvoidLines}}
\DeclareOptionX{s}%
    {\setkeys{luatodonotes.sty}{leadertype=s,positioning=sLeaderNorthEastBelowStacks}}
\DeclareOptionX{bezier}%
    {\setkeys{luatodonotes.sty}{leadertype=sBezier,positioning=sLeaderNorthEastBelowStacks}}
\DeclareOptionX{opo}%
    {\setkeys{luatodonotes.sty}{leadertype=opo,positioning=inputOrderStacks}}
```

Specify the name of the algorithm used to specify the position of the labels.

```
\define@key{luatodonotes.sty}%
    {positioning}{\renewcommand{\@todonotes@positioning}{#1}}
```

Specify the name of the algorithm used to split the notes for left and right side.

```
\define@key{luatodonotes.sty}%
    {splitting}{\renewcommand{\@todonotes@splitting}{#1}}
```

Specify the type of leaders that are drawn.

```
\define@key{luatodonotes.sty}%
    {leadertype}{\renewcommand{\@todonotes@leadertype}{#1}}
```

Specify the vertical distance between the notes.

```
\define@key{luatodonotes.sty}%
    {interNoteSpace}{\renewcommand{\@todonotes@interNoteSpace}{#1}}
```

Specify the distance from the text inside the notes to the border.

```
\define@key{luatodonotes.sty}%
    {noteInnerSep}{\renewcommand{\@todonotes@noteInnerSep}{#1}}
```

Specify the width of the routing area used for *opo*- and *os*-leaders.

```
\define@key{luatodonotes.sty}%
    {routingAreaWidth}{\renewcommand{\@todonotes@routingAreaWidth}{#1}}
```

Minimum width of notes in one margin beside the text to be considered for label placement.

```
\define@key{luatodonotes.sty}%
    {minNoteWidth}{\renewcommand{\@todonotes@minNoteWidth}{#1}}
```

Specify horizontal distance from the notes to the borders of the page.

```
\define@key{luatodonotes.sty}%
    {distanceNotesPageBorder}%
    {\renewcommand{\@todonotes@distanceNotesPageBorder}{#1}}
```

Specify the horizontal distance between the notes and the text area.

```
\define@key{luatodonotes.sty}%
    {distanceNotesText}{\renewcommand{\@todonotes@distanceNotesText}{#1}}
```

Specify the height of the raster used for the *po*-leader algorithm.

```
\define@key{luatodonotes.sty}%
    {rasterHeight}{\renewcommand{\@todonotes@rasterHeight}{#1}}
```

`additionalMargin` Control whether the margin should be enlarged for the notes and its width.

```
\newif{\if@todonotes@additionalMarginEnabled}
\@todonotes@additionalMarginEnabledfalse
\define@key{luatodonotes.sty}%
    {additionalMargin}[\@todonotes@additionalMargin]{%
        \@todonotes@additionalMarginEnabledtrue
        \renewcommand{\@todonotes@additionalMargin}{#1}}
```

This option is used to activate debug mode. Luatex prints more verbose output to the commandline in this mode. Furthermore, some of the algorithms also print debugging hints onto the output page.

```
\newif{\if@todonotes@debugenabled}
\@todonotes@debugenabledfalse
\DeclareOptionX{debug}{\@todonotes@debugenabledtrue}
```

Finally process the given options.

```
\ProcessOptionsX*
```

If the `obeyDraft` is given, check whether one of the `draft`, `draftcls` or `draftclsnofoot` options are given and enable or disable the functionality of this package. If the `obeyFinal` option is given together with the `final` option the todonotes are disabled. The `disable` option will overrule the effect of `obeyDraft`.

```
\if@todonotes@disabled
\else
    \if@todonotes@obeyDraft
        \@todonotes@disabledtrue
        \if@todonotes@isDraft
            \@todonotes@disabledfalse
        \fi
    \fi
    \if@todonotes@obeyFinal
        \@todonotes@disabledfalse
        \if@todonotes@isFinal
            \@todonotes@disabledtrue
        \fi
    \fi
\fi
```

If the option `additionalMargin` is given, we enlarge the margins for the notes.

```
\if@todonotes@additionalMarginEnabled
    \newlength{\@todonotes@modpaperwidth}
    \AfterEndPreamble{%
        \@todonotes@setAdditionalMargin%
```

Additionally, if the `geometry` package is loaded we hook into `\Gm@changelayout` to repeat this computation whenever `\newgeometry` is called.

```
        \ifdefined\Gm@changelayout
            \g@addto@macro{\Gm@changelayout}{\@todonotes@setAdditionalMargin}
        \fi
    }%
\fi%
```

We simply increase the page size by the doubled value of `additionalMargin` and move the contents to the right using `\hoffset`.

```
\newcommand{\@todonotes@setAdditionalMargin}{
    \setlength{\@todonotes@modpaperwidth}{\paperwidth}%
    \addtolength{\@todonotes@modpaperwidth}{\@todonotes@additionalMargin}%
    \addtolength{\@todonotes@modpaperwidth}{\@todonotes@additionalMargin}%
    \ifdefined\pdfpagewidth\else\let\pdfpagewidth\pagewidth\fi
    \pdfpagewidth=\@todonotes@modpaperwidth%
    \addtolength{\hoffset}{\@todonotes@additionalMargin}%
}
```

2.3 Initialisation of our Lua code

In this part we define some of the variables used by Lua depending on the package options and do some other initialisation tasks.

We first need some temporary dimensions, which are written by TEX and read from Lua. We use dimensions here because it is easier to access TEX dimensions from Lua than LATEX lengths. We use `tex.dimen` in Lua to access dimensions. The first dimensions are used when extracting the absolute coordinates of a position on the page.

```
\newdimen\@todonotes@extractx
\newdimen\@todonotes@extracty
```

The following savebox and dimensions are used to calculate the height of a certain label. The box and dimensions are filled by TEX and then read from Lua.

```
\newsavebox\@todonotes@heightcalcbox
\newdimen\@todonotes@heightcalcboxdepth
\newdimen\@todonotes@heightcalcboxheight
```

The following savebox is used to store the contents of a note and is then read from Lua.

```
\newsavebox\@todonotes@notetextbox
```

The following dimensions are used to read `\baselineskip`, `\normalbaselineskip` and `\f@size` from Lua. We need `\normalbaselineskip` as `\baselineskip` is set to 0 inside tabular cells. Dimension `\@todonotes@currentsidemargin` is set to the left margin, i. e., to the value of length `\oddsidemargin` or `\evensidemargin` depending on the type page.

```
\newdimen\@todonotes@baselineskip
\newdimen\@todonotes@normalbaselineskip
\newdimen\@todonotes@fontsize
\newdimen\@todonotes@currentsidemargin
```

Loading our main Lua file.

```
\directlua{require("luatodonotes")}
```

Setting variables to values given by package options.

```
\directlua{luatodonotes.noteInnerSep =
    string.todimen("\luatexluaescapestring{\@todonotes@noteInnerSep}")}
\directlua{luatodonotes.noteInterSpace =
    string.todimen("\luatexluaescapestring{\@todonotes@interNoteSpace}")}
\directlua{luatodonotes.routingAreaWidth =
    string.todimen("\luatexluaescapestring{\@todonotes@routingAreaWidth}")}
\directlua{luatodonotes.minNoteWidth =
    string.todimen("\luatexluaescapestring{\@todonotes@minNoteWidth}")}
\directlua{luatodonotes.distanceNotesPageBorder =
    string.todimen("\luatexluaescapestring{\@todonotes@distanceNotesPageBorder}")}
\directlua{luatodonotes.distanceNotesText =
    string.todimen("\luatexluaescapestring{\@todonotes@distanceNotesText}")}
\directlua{luatodonotes.rasterHeight =
    string.todimen("\luatexluaescapestring{\@todonotes@rasterHeight}")}
```

Set the variables for the used algorithms and leader types depending on the corresponding package options.

```
\directlua{luatodonotes.setPositioningAlgo("\luatexluaescapestring{\@todonotes@positioning}")}
\directlua{luatodonotes.setSplittingAlgo("\luatexluaescapestring{\@todonotes@splitting}")}
```

```
\directlua{luatodonotes.setLeaderType("\luatexluaescapestring{\@todonotes@leadertype}")}
```

The following commands are used to detect the absolute positions of lines on the page.

We first need to define a command to be able to insert the position from `\pdflastypos` into a write-whatsit in Lua. We need this workaround because we cannot insert `\pdflastypos` directly into the tokenlist in the Lua callback `callbackOutputLinePositions()`.

```
\ifdefined\pdflastypos\else\let\pdflastypos\lastypos\fi
\def\@todonotes@pdflastypos{\the\pdflastypos}
```

The following commands are written to the temporary `lpo`-file. When reading this file we call a Lua function for each line in the file and thus can collect the line positions in a Lua table.

```
\newcommand{\@todonotes@lineposition}[3]{%
    \directlua{luatodonotes.linePositionsAddLine(#1,#2,#3)}%
}
\newcommand{\@todonotes@nextpage}{%
    \directlua{luatodonotes.linePositionsNextPage()}%
}%
```

The following macro is used in `AtBeginShipout` to signal in the `lpo`-file that a new page is started.

```
\newcommand{\@todonotes@writeNextpageToLpo}{%
    \ifdefined\tf@lpo%
        \immediate\write\tf@lpo{\@backslashchar @todonotes@nextpage}%
    \fi
}
```

Depending on the debug-option of the package we set the corresponding Lua variable here. Additionally, we prepare to print our notes and leaders in foreground when in debug mode.

```
\if@todonotes@debugenabled
    \directlua{luatodonotes.todonotesDebug = true}
    \newcommand{\@todonotes@AtBeginShipoutUpperLeft}
        {\AtBeginShipoutUpperLeftForeground}
\else
    \directlua{luatodonotes.todonotesDebug = false}
    \newcommand{\@todonotes@AtBeginShipoutUpperLeft}
        {\AtBeginShipoutUpperLeft}
\fi
```

Define commands that are used for every tikzpicture to disable externalization. We only call the `\tikzexternaldisable` command if it is defined (i.e., the externalization library for TikZ is loaded). Externalization is reenabled after the end of the group (if it was enabled before).

```
\newcommand{\@todonotes@before@tikzpict}{\begingroup%
    \ifdefined\tikzexternaldisable\tikzexternaldisable\fi}
\newcommand{\@todonotes@after@tikzpict}{\endgroup}
```

Initialise the script when all Lua variables are set according to the package options.

```
\directlua{luatodonotes.initTodonotes()}
```

Some definitions to highlight areas in text. The first command is needed to accept control spaces (\ ) in arguments for soul commands. After that we define the highlighting command used for todoareas.

```
\soulregister{\ }{0}
\newlength{\todonotes@textmark@width}
\newlength{\todonotes@textmark@fontsize}
\newlength{\todonotes@textmark@linebelow}
\newlength{\todonotes@textmark@lineabove}
\ulposdef{\todonotes@textmark@highlight}{%
    \setlength\todonotes@textmark@width\ulwidth%
    \setlength\todonotes@textmark@fontsize{\f@size pt}%
    \stepcounter{@todonotes@numberofLinesInArea}%
    \ifulstarttype{0}%
        {% begin of area
        \def\todonotes@textmark@decoLeft{}%
        \def\todonotes@textmark@shift{-2pt}%
        \addtolength\todonotes@textmark@width{2pt}%
        \setcounter{@todonotes@numberofLinesInArea}{1}}%
        {\def\todonotes@textmark@decoLeft{@todonotes@todoarea}%
        \def\todonotes@textmark@shift{-4pt}%
        \addtolength\todonotes@textmark@width{4pt}}%
    \ifulendtype{0}%
        {% last line of area
        \def\todonotes@textmark@decoRight{}%
        \addtolength\todonotes@textmark@width{2pt}%
        \directlua{luatodonotes.processLastLineInTodoArea()}}%
        {\def\todonotes@textmark@decoRight{@todonotes@todoarea}%
        \addtolength\todonotes@textmark@width{4pt}}%
    \newcommand{\@todonotes@nodeNamePrefix}%
        {@todonotes@\arabic{@todonotes@numberoftodonotes}%
         @\arabic{@todonotes@numberofLinesInArea} }%
    \hspace*{\todonotes@textmark@shift}{\smash{%
        \@todonotes@before@tikzpict%
        \begin{tikzpicture}[overlay,remember picture,
            deco/.style={}]%
            \setlength\todonotes@textmark@linebelow%
                {-0.95\dimexpr\baselineskip-\f@size pt\relax}%
            \setlength\todonotes@textmark@lineabove%
                {\dimexpr\f@size pt+\todonotes@textmark@linebelow\relax}%
            \coordinate
                (\@todonotes@nodeNamePrefix areaSW)
                at (0,\todonotes@textmark@linebelow);
            \coordinate
                (\@todonotes@nodeNamePrefix areaSE)
                at (\todonotes@textmark@width, \todonotes@textmark@linebelow);
            \coordinate
                (\@todonotes@nodeNamePrefix areaNE)
                at (\todonotes@textmark@width,\todonotes@textmark@lineabove);
```

```
            \coordinate
                (\@todonotes@nodeNamePrefix areaNW)
                at (0,\todonotes@textmark@lineabove);
            \draw[draw=green!70,fill=green,fill opacity=.2]
                (\@todonotes@nodeNamePrefix areaSW)
                decorate[\todonotes@textmark@decoLeft] {
                    -- (\@todonotes@nodeNamePrefix areaNW)
                }
                -- (\@todonotes@nodeNamePrefix areaNE)
                decorate[\todonotes@textmark@decoRight] {
                    -- (\@todonotes@nodeNamePrefix areaSE)
                }
                -- cycle;
        \end{tikzpicture}%
        \@todonotes@after@tikzpict%
    }}%
}%
```

2.4 Options for the todo command

In this part the various options for commands in the package are defined. Set an arbitrarily fill color

```
\newcommand{\@todonotes@currentlinecolor}{}%
\newcommand{\@todonotes@currentbackgroundcolor}{}%
\newcommand{\@todonotes@currentbordercolor}{}%
\define@key{todonotes}{color}{%
    \renewcommand{\@todonotes@currentlinecolor}{#1}%
    \renewcommand{\@todonotes@currentbackgroundcolor}{#1}}%
\define@key{todonotes}{linecolor}{%
    \renewcommand{\@todonotes@currentlinecolor}{#1}}%
\define@key{todonotes}{backgroundcolor}{%
    \renewcommand{\@todonotes@currentbackgroundcolor}{#1}}%
\define@key{todonotes}{bordercolor}{%
    \renewcommand{\@todonotes@currentbordercolor}{#1}}%
\newcommand{\@todonotes@currentleaderwidth}{}%
\define@key{todonotes}{leaderwidth}{%
    \renewcommand{\@todonotes@currentleaderwidth}{#1}}%
```

Set a relative font size

```
\newcommand{\@todonotes@sizecommand}{}%
\define@key{todonotes}{size}{\renewcommand{\@todonotes@sizecommand}{#1}}%
```

Should the todo item be disabled?

```
\newif\if@todonotes@localdisable%
\define@key{todonotes}{disable}[]{\@todonotes@localdisabletrue}%
\define@key{todonotes}{nodisable}[]{\@todonotes@localdisablefalse}%
```

Should the todo item be included in the list of todos?

```
\newif\if@todonotes@appendtolistoftodos%
\define@key{todonotes}{list}[]{\@todonotes@appendtolistoftodostrue}%
\define@key{todonotes}{nolist}[]{\@todonotes@appendtolistoftodosfalse}%
```

Should the todo item be displayed inline?

```
\newif\if@todonotes@inlinenote%
\define@key{todonotes}{inline}[]{\@todonotes@inlinenotetrue}%
\define@key{todonotes}{noinline}[]{\@todonotes@inlinenotefalse}%
\newif\if@todonotes@prependcaption%
\define@key{todonotes}{prepend}[]{\@todonotes@prependcaptiontrue}%
\define@key{todonotes}{noprepend}[]{\@todonotes@prependcaptionfalse}%
```

Should the note in the margin be connected to the insertion point in the text?

```
\newif\if@todonotes@line%
\define@key{todonotes}{line}[]{\@todonotes@linetrue}%
\define@key{todonotes}{noline}[]{\@todonotes@linefalse}%
```

Only here for compatibility with todonotes. We don't need the fancy lines because we have more advanced drawing styles. So we ignore this option and issue a warning.

```
\define@key{todonotes}{fancyline}[]{\PackageWarningNoLine{luatodonotes}
    {Parameter fancyline is not supported by luatodonotes}}%
\define@key{todonotes}{nofancyline}[]{}%
```

Author option.

```
\newcommand{\@todonotes@author}{}%
\newif\if@todonotes@authorgiven%
\define@key{todonotes}{author}{%
    \renewcommand{\@todonotes@author}{#1}%
    \@todonotes@authorgiventrue}%
\define@key{todonotes}{noauthor}[]{\@todonotes@authorgivenfalse}%
```

Should the text in the list of todos be different from the text in the todonote?

```
\newcommand{\@todonotes@caption}{}%
\newif\if@todonotes@captiongiven%
\define@key{todonotes}{caption}%
    {\renewcommand{\@todonotes@caption}{#1}%
    \@todonotes@captiongiventrue}%
\define@key{todonotes}{nocaption}[]{\@todonotes@captiongivenfalse}%
```

Change the current figure width and height.

```
\newcommand{\@todonotes@currentfigwidth}{\@todonotes@figwidth}
\define@key{todonotes}%
    {figwidth}{\renewcommand{\@todonotes@currentfigwidth}{#1-2pt}}
\newcommand{\@todonotes@currentfigheight}{\@todonotes@figheight}
\define@key{todonotes}%
    {figheight}{\renewcommand{\@todonotes@currentfigheight}{#1-2pt}}
\newcommand{\@todonotes@currentfigcolor}{\@todonotes@figcolor}
\define@key{todonotes}%
    {figcolor}{\renewcommand{\@todonotes@currentfigcolor}{#1}}
```

Preset values of the options

```
\presetkeys%
    {todonotes}%
    {linecolor=\@todonotes@linecolor,%
```

```
    backgroundcolor=\@todonotes@backgroundcolor,%
    bordercolor=\@todonotes@bordercolor,%
    leaderwidth=\@todonotes@leaderwidth,%
    nodisable,%
    noinline,%
    nocaption,%
    noauthor,%
    figwidth=\@todonotes@figwidth,%
    figheight=\@todonotes@figheight,%
    figcolor=\@todonotes@figcolor,%
    line, list, size=\@todonotes@textsize}{}%
```

2.5 The main code part

Here are the actual macros defined. The following boolean is used to remember if `\todo` or `\todoarea` was called.

```
\newif\if@todonotes@areaselected%
```

The following token registers are used to access the data for a note (which is stored in macros) from Lua.

```
\newtoks\@todonotes@toks@currentlinecolor%
\newtoks\@todonotes@toks@currentbackgroundcolor%
\newtoks\@todonotes@toks@currentbordercolor%
\newtoks\@todonotes@toks@currentleaderwidth%
\newtoks\@todonotes@toks@sizecommand%
```

If the option "disable" was passed to the package define empty commands.

```
\if@todonotes@disabled%
    \newcommand{\listoftodos}[1][]{}
    \newcommand{\@todo}[2][]{}
    \newcommand{\@todoarea}[3][]{}
    \newcommand{\missingfigure}[2][]{}
\else % \if@todonotes@disabled
```

`\listoftodos` Define the `\listoftodos` command and define the appearance of the list of todos.

```
\newcounter{todonotes@oldtocdepth}
\newcommand{\listoftodos}[1][\@todonotes@todolistname]{%
    \setcounter{todonotes@oldtocdepth}{\value{tocdepth}}%
    \setcounter{tocdepth}{1}%
    \@ifundefined{chapter}{\section*{#1}}{\chapter*{#1}} \@starttoc{tdo}%
    \setcounter{tocdepth}{\value{todonotes@oldtocdepth}}%
}
\newcommand{\l@todo}
    {\@dottedtocline{1}{0em}{2.3em}}
```

Define styles used by the todo command. Colors are set directly when placing the notes.

```
\tikzset{@todonotes@todoarea/.style={
```

```
    decoration={snake,amplitude=3.5pt,segment length=5pt}}}
\tikzset{@todonotes@notestyleraw/.style={
    line width=0.5pt,
    inner sep = \@todonotes@noteInnerSep,
    rounded corners=4pt}}
```

Add shadows and rounded corners to the inserted todonotes.

```
\if@todonotes@shadowenabled
    \tikzset{@todonotes@notestyle/.style={@todonotes@notestyleraw,
        general shadow={shadow xshift=.5ex, shadow yshift=-.5ex,
        opacity=1,fill=black!50}}}
\else
    \tikzset{@todonotes@notestyle/.style={@todonotes@notestyleraw}}
\fi
\tikzset{@todonotes@leader/.style={}}
\tikzset{@todonotes@textmark/.style={rounded corners}}
\tikzset{@todonotes@inlinenote/.style={
    @todonotes@notestyle,
    draw=\@todonotes@currentbordercolor,
    fill=\@todonotes@currentbackgroundcolor,
    text width=\linewidth - 1.6 ex - 1 pt}}
```

`\@todocommon` Common macro used from `\@todo` and `\@todoarea`. Used to actually draw/save the note.

```
\newcommand{\@todocommon}[2]{%
```

Use the global value for determining the default prepend behavior.

```
\if@todonotes@prependcaptionglobal%
\@todonotes@prependcaptiontrue%
\else%
\@todonotes@prependcaptionfalse%
\fi%
```

Store the original text for later usage and parse the given options.

```
\renewcommand{\@todonotes@text}{#2}%
\renewcommand{\@todonotes@caption}{#2}%
\setkeys{todonotes}{#1}%
```

If the option `disable` is given to the command, no output is generated.

```
\if@todonotes@localdisable%
\else%
```

Add the item to the list of todos. When the option `colorinlistoftodos` is given to the package a small colored square is added in front of the text.

```
\addtocounter{@todonotes@numberoftodonotes}{1}%
\if@todonotes@appendtolistoftodos%
    \phantomsection%
    \if@todonotes@captiongiven%
    \else%
        \renewcommand{\@todonotes@caption}{#2}%
    \fi%
```

```
    \@todonotes@addElementToListOfTodos%
\fi%
```

Prepend the short caption given if it is requested

```
\if@todonotes@captiongiven%
    \if@todonotes@prependcaption%
        \renewcommand{\@todonotes@text}{\@todonotes@caption: #2}%
    \fi%
\fi%
```

Place the todonote as indicated by the options (inline or in a marginpar), below is the code for the inline placement.

```
\if@todonotes@inlinenote%
    \@todonotes@drawInlineNote%
\else%
    \@todonotes@drawMarginNoteWithLine%
\fi%\if@todonotes@inlinenote
\fi%\if@todonotes@localdisable
}%
```

`\@todo` Command that draws normal notes.

```
\newcommand{\@todo}[2][]{%
    \@todonotes@areaselectedfalse%
    \@todocommon{#1}{#2}%
}%
```

`\@todoarea` Command that draws notes that highlight a certain area in text.

```
\newcommand{\@todoarea}[3][]{%
    \@todonotes@areaselectedtrue%
    \@todocommon{#1}{#2}%
    \todonotes@textmark@highlight{#3}%
```

Mark the end of the highlighted area with a Tikz coordinate. The begin is marked by `\@todocommon`.

```
    \@todonotes@before@tikzpict%
    \begin{tikzpicture}[remember picture, overlay]%
        \node [coordinate] (@todonotes@\arabic{@todonotes@numberoftodonotes} %
            inTextEnd) {};%
    \end{tikzpicture}%
    \@todonotes@after@tikzpict%
    \zref@label{@todonotes@\arabic{@todonotes@numberoftodonotes}@end}%
}%
```

`drawMarginNoteWithLine` Define helper function `drawMarginNoteWithLine`.

```
\newcommand{\@todonotes@drawMarginNoteWithLine}{%
```

When the todonote should be placed inside a marginpar, the code below is applied. First is the current location in the document stored, this enables us later to connect this point with the inserted todonote.

```
    \@todonotes@before@tikzpict%
    \begin{tikzpicture}[remember picture, overlay]%
```

```
        \node [coordinate] (@todonotes@\arabic{@todonotes@numberoftodonotes} %
            inText) {};%
    \end{tikzpicture}%
    \@todonotes@after@tikzpict%
```

Update the dimensions to be accessed by Lua.

```
    \@todonotes@baselineskip=\baselineskip%
    \@todonotes@normalbaselineskip=\normalbaselineskip%
    \@todonotes@fontsize=\f@size pt%
```

Place a label at the site. We use this to query the page number, on which the note was placed.

```
    \zref@label{@todonotes@\arabic{@todonotes@numberoftodonotes}}%
```

Append author before the note text if one is given.

```
    \if@todonotes@authorgiven%
        \let\@todonotes@text@old=\@todonotes@text
        \renewcommand{\@todonotes@text}{\@todonotes@author: \@todonotes@text@old}%
    \fi%
```

We use edef here to get these macros fully expanded. After that we write them to a toks register and read them from Lua.

```
    \edef\@todonotes@tmp{\@todonotes@currentlinecolor}%
    \@todonotes@toks@currentlinecolor=\expandafter{\@todonotes@tmp}%
    \edef\@todonotes@tmp{\@todonotes@currentbackgroundcolor}%
    \@todonotes@toks@currentbackgroundcolor=\expandafter{\@todonotes@tmp}%
    \edef\@todonotes@tmp{\@todonotes@currentbordercolor}%
    \@todonotes@toks@currentbordercolor=\expandafter{\@todonotes@tmp}%
    \edef\@todonotes@tmp{\@todonotes@currentleaderwidth}%
    \@todonotes@toks@currentleaderwidth=\expandafter{\@todonotes@tmp}%
```

We cannot fully expand the size command (using `\edef` causes errors when compiling).

```
    \@todonotes@toks@sizecommand=\expandafter{\@todonotes@sizecommand}%
```

We store the text that should be shown in this note into a box and copy this box to a variable in Lua. The commands `\@parboxrestore`, `\@marginparreset`, `\@minipagefalse` and `\outer@nobreak` are copied from the definition of `\marginpar` in LaTeX2e to reset font settings, for example. This is important when a note is placed inside a theorem environment.

```
    \savebox\@todonotes@notetextbox{%
        \@parboxrestore
        \@marginparreset
        \@todonotes@sizecommand\@todonotes@text%
        \@minipagefalse
        \outer@nobreak
    }%
```

Prepare parameters and add the note to the list in Lua.

```
    \if@todonotes@line%
        \def\@todonotes@param@drawLeader{true}%
    \else%
```

```
        \def\@todonotes@param@drawLeader{false}%
    \fi%
    \if@todonotes@areaselected%
        \def\@todonotes@param@noteType{area}%
    \else%
        \def\@todonotes@param@noteType{}%
    \fi%
    \directlua{luatodonotes.addNoteToList(\arabic{@todonotes@numberoftodonotes},%
        \@todonotes@param@drawLeader,\luastring0{\@todonotes@param@noteType})}%
}%
```

addElementToListOfTodos Define helper function addElementToListOfTodos.

```
\newcommand{\@todonotes@addElementToListOfTodos}{%
    \if@todonotes@colorinlistoftodos%
        \addcontentsline{tdo}{todo}{%
            \fcolorbox{\@todonotes@currentbordercolor}%
                {\@todonotes@currentbackgroundcolor}%
                {\textcolor{\@todonotes@currentbackgroundcolor}{o}}%
            \ \@todonotes@caption}%
    \else%
        \addcontentsline{tdo}{todo}{\@todonotes@caption}%
    \fi}%
```

drawInlineNote Define helper function drawInlineNote.

```
\newcommand{\@todonotes@drawInlineNote}{%
    {\par\noindent%
        \@todonotes@before@tikzpict%
        \begin{tikzpicture}[remember picture]%
        \draw node[@todonotes@inlinenote,font=\@todonotes@sizecommand]{%
            \if@todonotes@authorgiven%
                {\noindent \@todonotes@sizecommand %
                    \@todonotes@author:\,\@todonotes@text}%
            \else%
                {\noindent \@todonotes@sizecommand \@todonotes@text}%
            \fi};%
        \end{tikzpicture}%
        \@todonotes@after@tikzpict%
    \par}%
    }%
```

\missingfigure Defines the \missingfigure macro.

```
\newcommand{\missingfigure}[2][]{%
\setkeys{todonotes}{#1}%
\addcontentsline{tdo}{todo}{\@todonotes@MissingFigureText: #2}%
\par
\noindent
\@todonotes@before@tikzpict%
\begin{tikzpicture}
\draw[fill=\@todonotes@currentfigcolor, draw = black!40, line width=2pt]
```

```
    (-2, -2.5) rectangle +(\@todonotes@currentfigwidth, \@todonotes@currentfigheight);
\draw (2, -0.3) node[right, text
    width=\@todonotes@currentfigwidth-4.5cm] {#2};
\draw[red, fill=white, rounded corners = 5pt, line width=10pt]
    (30:2cm) -- (150:2cm) -- (270:2cm) -- cycle;
\draw (0, 0.3) node {\@todonotes@MissingFigureUp};
\draw (0, -0.3) node {\@todonotes@MissingFigureDown};
\end{tikzpicture}\hfill
\@todonotes@after@tikzpict%
}% Ending \missingfigure command
\fi% Ending \@todonotes@ifdisabled
```

`\todototoc` Inserts a reference to the list of todos in the table of contents. If `chapter` is defined, `chapter` is used as level otherwise will `section` be used. The `\todototoc` command respects the disable option.

```
\newcommand{\todototoc}
{
    \if@todonotes@disabled
    \else
    \addcontentsline{toc}{\@ifundefined{chapter}{section}{chapter}}{\@todonotes@todolistname}
    \fi
}
```

`\todo` Define the `\todo` command as a redirection to `\@todo`.

```
\newcommand{\todo}[2][]{\@bsphack\@todo[#1]{#2}\@esphack\ignorespaces}%
```

`\todoarea` Define the `\todoarea` command as a redirection to `\@todoarea`. We don't want to ignore spaces after this command.

```
\newcommand{\todoarea}[3][]{\@bsphack\@todoarea[#1]{#2}{#3}\@esphack}%
```

The following commands are executed when a page is complete and is written to the output PDF (shipout in TEX terms). The `\AtBeginShipout` command is provided by package atbegshi.

```
\if@todonotes@disabled
\else
\AtBeginShipout{%
```

We draw to the foreground or background of the page (depending if debug option is set for the package).

```
    \@todonotes@AtBeginShipoutUpperLeft{
        \@todonotes@writeNextpageToLpo
```

Determine if we are on a left or on a right side (important for margins) and set variables accordingly. `\relax` seems to be needed at end to really write new value for currentsidemargin.

```
        \checkoddpage%
        \ifoddpageoroneside%
            \@todonotes@currentsidemargin=\the\oddsidemargin%
        \else%
```

```
            \@todonotes@currentsidemargin=\the\evensidemargin%
        \fi\relax%
```

We switch to the default catcodes of LaTeX here. This is important if catcodes are changed in the main text, e. g., by a verbatim environment at the end of the page.

```
        \BeginCatcodeRegime\CatcodeTableLaTeX
```

Calculates the areas, in which the labels can be placed. This calculation depends on currentsidemargin. So this has to be done inside `\AtBeginShipoutUpperLeft` (otherwise odd/even page detection won't work).

```
        \directlua{luatodonotes.calcLabelAreaDimensions()}%
```

Calculates the needed height for every note. This has to be outside of the tikzpicture because it uses a savebox to compute the height. This box does not work in the tikzpicture.

```
        \directlua{luatodonotes.calcHeightsForNotes()}% has to be outside of tikzpicture
```

Some classes modify the page margins using `\voffset` and `\hoffset`. Our `tikzpicture` would be aligned using this modified page origin. So we overrule the offsets using a `raisebox` and a negative `hspace`.

```
        \raisebox{\voffset}{%
            \hspace{-\hoffset}%
            \@todonotes@before@tikzpict%
            \begin{tikzpicture}[remember picture,overlay]
```

Reads the absolute coordinates of every note on the page and writes them to the Lua objects.

```
                \directlua{luatodonotes.getInputCoordinatesForNotes()}
```

Runs the positioning algorithm and actually draws the notes and leaders.

```
                \directlua{luatodonotes.printNotes()}
            \end{tikzpicture}%
            \@todonotes@after@tikzpict%
        }%
```

Delete the drawn notes from the Lua lists and prepare for the next page.

```
        \directlua{luatodonotes.clearNotes()}%
        \EndCatcodeRegime
    }%
}
\fi % Ending \@todonotes@ifdisabled
```

Change History