The todonotes package*

Henrik Skov Midtiby
henrikmidtiby@gmail.com

March 7, 2011

Abstract

The `todonotes` 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.

Contents

*This document corresponds to todonotes .dtx, dated 2011/03/07.

1 Introduction

The `todonotes` 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 . . .). I developed the basic functionality of the package while I worked on my bachelor project.

1.1 Usage

`\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

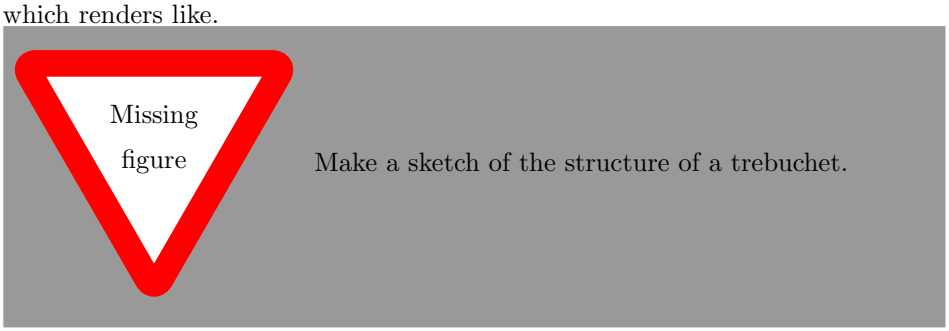
`\todo{Make a cake \ldots}`,

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 1.3.

`\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.

1.2 Package options

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

`obeyDraft` When the option `obeyDraft` is given, the package checks if the option `draft` 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.

`danish, german, ngerman, french spanish, catalan, italian portuguese, dutch` 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, danish, dutch, french, german, ngerman, italian, portuguese and spanish.

`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=orange`). 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.

`textwidth`

`textwidth=length` sets the width of a todo item in the margin to `length`. The width of inline todonotes will allways be the same as the current line width.

`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.

`dvistyle`

When a document with todonotes is compiled with plain latex (to a dvi-file), there is an issue with the visual appearance[1]. The option `dvistyle` changes the appearance of the inserted todonotes to avoid this problem.

`figwidth`

The `figwidth=length` option sets the default width of the figure inserted by the `\missingfigure` command. The default value is `\columnwidth`.

1.3 Options for the todo command

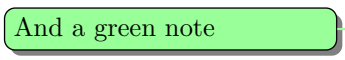
There are several options that can be given to the `\todo` command. All the options are described here and often I have included examples of the change in visual appearance.

`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 1.2. 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.




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

[1] The problem is placement of text inside the colored boxes.

Anything but default colors

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

```
\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 ...}`

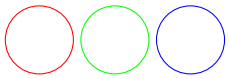
A note with no line connecting it to the placement in the original text.

`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.



```
\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}
```

Figure 1: A text explaining the image.

Fill those circles . . .

`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.}.
```

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

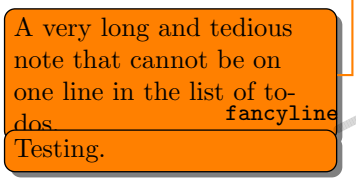
`prepend / noprepend`

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

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

using the code:

```
\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.}.
```



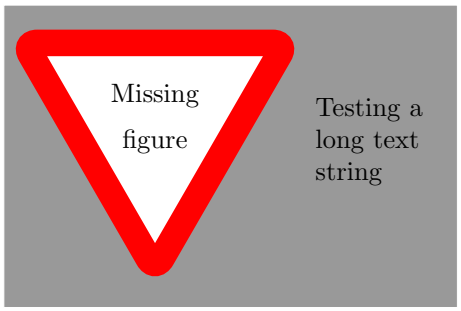
The `fancyline` option inserts a curved arrow, pointing from the inserted note to the insertion point. The option is used like this:

```
\todo[fancyline]{Testing.}
```

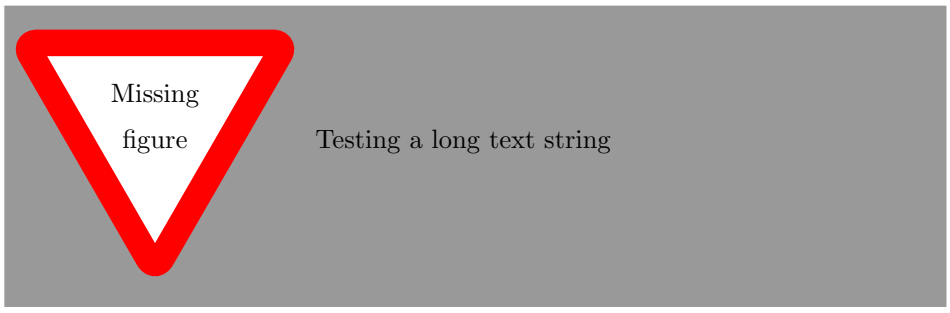
1.4 Options for the missingfigure command

`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}
```

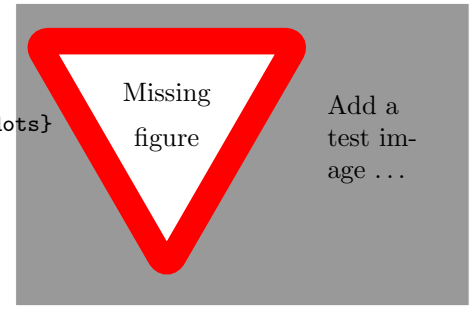



```
\missingfigure{Testing a long text string}
```



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}
```



1.5 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]
```

1.6 Known issues

1.6.1 Package loading order

The todonotes package requires the following packages.

- ifthen
- xkeyval
- xcolor
- tikz
- calc
- graphicx (is loaded via the tikz package)

When todonotes 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 todonotes package, otherwise you will get an "Option clash" error when latex works on the document.

1.6.2 Wrapping of long lines in list of todos

When a document is compiled with latex (and not pdflatex) long items in the list of todos are not wrapped into several lines, and do instead continue to the right out of the page.

1.6.3 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
```

1.6.4 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 9.2.2 Producing PDF Output" in the tikz manual. http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf

1.6.5 Todonotes wrongly placed in the margin

When using some document classes or packages, the todonotes inserted in the page margin can be placed quite oddly. This is often caused by a wrong value of the `\marginparwidth` lenght. Try using the code below in your preamble to see if this cures the problem.

```
\setlength{\marginparwidth}{2cm}
```

If the todonotes are inserted in the wrong margin, the solution is the `\reversemarginpar` command. When this command is issued the following marginpars (which todonotes relies on) is inserted in the other margin.

1.7 Things to improve

This is a list of things I consider to improve sometime in the future. It have not been done yet as I lack the time or skills to implement them. Patches with implementations of these tasks will be appreciated and might be included in the package if it will improve the package quality.

1.7.1 Owner information

Option for the todo command.

```
\todo[owner={Fabrice}]{Stuff}
```

Add info on who "owns" the current todo. Idea: Fabrice Niessen

1.7.2 Due date

Option for the todo command.

```
\todo[due=2008-12-07]{Stuff}
```

Add info on when the current todo is due. Might be enhanced by a time line of the todos that have a due date assigned. Idea: Fabrice Niessen

1.7.3 Mark accomplished todos

```
\todo[done]{Stuff}
```

Idea: Fabrice Niessen

1.8 Usage methods

In this section I have collected some different methods to use the `todonotes` package.

1.8.1 Define new commands with fixed options

Often there is a need for marking different classes of things to do (add reference, rewrite, ...). One way to do this, is to define some new commands as shown below (idea from Florent B.).

```
\newcommand{\addref}{\todo[color=red!40]{Add reference.}}
\newcommand{\rewrite}[1]{\todo[color=green!40]{#1}}
```

To distinguish between the different types of todos, the `todonotes` package can be loaded with the `colorinlistoftodos` option, which adds small colored squares to the list of todos.

```
\usepackage[colorinlistoftodos]{todonotes}
```

1.8.2 Define new commands with arbitrary default options

If you do not like the default values of the standard todo command, it is possible to define a new command with the similar functionality of `\todo` with custom default values.

```
\newcommand{\todoredefined}[2][]
{\todo[color=red, #1]{#2}}
```

Test of newly defined command.

The new command can now be used like shown below

```
\todoredefined{Test of newly defined command.}
\todoredefined[color=green]{Test of newly defined command, requesting a green color.}
```

Test of newly defined command, requesting a green color.

This can be done with all the accepted options for the `\todo` command.

1.8.3 Enumerate todonotes

If the inserted todonotes should be enumerated, it is possible to define a new command with the desired behaviour.

```
\newcounter{todocounter}
\newcommand{\todonum}[2][]
{\stepcounter{todocounter}\todo[#1]{\thetodocounter: #2}}
```

1: A numbered todonote.

2: Another numbered todonote.

The idea is to define a new counter `todocounter`, and insert the value of the counter in each todonote. The new command can be used like

```
\todonum{A numbered todonote.}
\todonum{Another numbered todonote.}
```

1.8.4 Comments "a la Word"

Fabrice Niessen sent me the following use case. The idea is to define a new command `\mycomment` which adds a counter and optionally the initials of the author to the inserted todonote.

```
\newcounter{mycomment}
\newcommand{\mycomment}[2][]{%
   % initials of the author (optional) + note in the margin
   \refstepcounter{mycomment}%
   {%
       \setstretch{0.7}% spacing
       \todo[color={red!100!green!33},size=\small]{%
           \textbf{Comment [\uppercase{#1}\themycomment]:}~#2}%
   }}
```

Comment [HSM1]: Testing first time.

Comment [HSM2]: Testing second time.

The command `\mycomment[HSM]{Testing first time.}` is displayed like shown in the left margin, and another call of the command is added below `\mycomment[HSM]{Testing second time.}`.

1.8.5 Combination with the fixme package

Thomas Arildsen has mailed me this use case. Check the documentation for the `fixme` package, as the code below relies directly on it (the `\FDUser` command is augmented when `\begin{document}` is reached).

```
\usepackage[user,nomargin]{fixme}
\usepackage{todonotes}
\newcommand{\FXUser}[2]{\todo[inline,size=\small]{{\bfseries #1:} #2}}
```

1.8.6 Altering the line spacing of todonotes

The `setspace` package lets you alter the line spacing of smaller sections of your document. The primary construct is the `spacing` environment, which is demonstrated below.

```
\begin{spacing}{0.5}
Some lines with a decreased line spacing. This is accomplished
using the setspace package that is included in standard latex
distributions.
\end{spacing}
```

Some lines with a decreased line spacing. This is accomplished using the setspace package that is included in standard latex distributions.

Using the `spacing` environment we can define a new todonote command using the code below:

```
\newcommand{\smalltodo}[2][]
    {\todo[caption={#2}, #1]
    {\begin{spacing}{0.5}#2\end{spacing}}}
```

Some lines with a decreased line spacing. This is accomplished using the setspace package that is included in standard latex distributions.

Todonotes with decreased line spacing can now be inserted as follows

```
\smalltodo[size=\footnotesize]{
Some lines with a decreased line spacing. This is accomplished
using the setspace package that is included in standard latex
distributions.}
```

A different approach is given by Vitaly.

```
\newcommand{\tinytodo}[2][]
  {\todo[caption={#2}, size=\small, #1]{\renewcommand{\baselinestretch}{0.5}\selectfont#2\par}}
```

Some lines with a decreased line spacing. This is accomplished without using any special packages.

It looks like seen here.

```
\tinytodo{
Some lines with a decreased line spacing. This is accomplished
without using any special packages.}
```

1.8.7 Marking new / old sections

Sometimes a whole section has to be marked by some means. You might want to try the following.

```
\todo[inline, caption={Some text}]{
\begin{minipage}{\linewidth}
Some text that might differ from the text given to the caption
option.
\end{minipage}
}
```

It is important to add the `caption={text}` option, otherwise latex will try to embed a minipage in the table of contents which triggers an error. Inside the minipage environment almost anything could be placed, except for other todo commands.

```
\todo[inline, caption={Examine this new section}]{
\begin{minipage}{\linewidth}
Some text.
\begin{align}
\sin(\theta)^2 + \cos(\theta)^2 = 1
\end{align}
A formula and a list
\begin{itemize}
\item An item
\end{itemize}
\end{minipage}
}
```

The example above renders like

Some text.

$$\sin(\theta)^2 + \cos(\theta)^2 = 1 \tag{1}$$

A formula and a list

- An item

1.8.8 Link to list of todos

Using the `hyperref` package it is possible to add a link from the inserted todonotes to the list of todos. The example were supplied by Andreas Plank.

```
% Define a counter for the inserted todonotes.
\newcounter{todoListItems}
\newcommand{\todoTrans}[2][ ]{
  % Increment counter
  \addtocounter{todoListItems}{1}
```

```
  \todo[%
    caption={\protect\hypertarget{todo\thetodoListItems}{}Translation},
    #1]
  {
    #2 \hfill
    \hyperlink{todo\thetodoListItems}{$\uparrow$}
  }
}
```

The idea behind the code is to embed a `hypertarget` in each entry in the list of todos. In the todonotes a link to the entry in the list of todos is inserted as an arrow that points upwards. Using the `\todoTrans` command like below, the following two notes have been inserted.

```
\todoTrans{papirsflyver}
\todoTrans[inline]{damplokomotiv}
```

1.8.9 Numbered todonotes

The inserted todonotes can be argumented with the current subsubsection number. The code is shown below.

```
\newcommand{\ntodo}[2][]{\todo[#1]{\thesubsubsection{}. #2}}
```

1.8.9. A numbered todo.

By changing `\thesubsubsection` to `\thesection`, the current section number can be inserted instead of the subsubsection number. The result looks like. Which were generated by the code

```
\ntodo{A subsection numbered todo.}.
```

1.8.10 Combining several modifications

Small notes with links back to the list of todos.

Smart notes with links back to the list of todos.

Manduca have combined several of the modifications above into a highly specialized todo command. He uses the code:

```
\newcounter{todoListItems}
\newcommand{\sstodo}[2][]
{\addtocounter{todoListItems}{1}
\todo[caption={\protect\hypertarget{todo\thetodoListItems}{}\thesection. #2}, #1]
{\begin{spacing}{1} \hfill \hyperlink{todo\thetodoListItems}{#2} \end{spacing} }}
```

Using this approach it is possible to customize the behavior of the inserted notes to a very high degree.

2 Implementation

Identifies the package and loads the packages dependences.

```
\ProvidesPackage{todonotes}[2011/03/07]
\RequirePackage{ifthen}
\RequirePackage{xkeyval}
\RequirePackage{xcolor}
\RequirePackage{tikz}
\RequirePackage{calc}
```

Some default values are set

```
\newcommand{\@todonotes@text}{}%
\newcommand{\@todonotes@backgroundcolor}{orange}
\newcommand{\@todonotes@linecolor}{orange}
\newcommand{\@todonotes@bordercolor}{black}
\newcommand{\@todonotes@textwidth}{\marginparwidth}
\newcommand{\@todonotes@textsize}{\normalsize}
\newcommand{\@todonotes@figwidth}{\columnwidth}
```

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

2.1 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{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{pendient}%
}
```

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

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

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}
```

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}
```

The default style behaves bad when compiled with latex (some text placement problems). The dvistyle option, changes the visual behavior to avoid this text placement problem.

```
\newif{\if@todonotes@dviStyle}
\DeclareOptionX{dvistyle}{\@todonotes@dviStyletrue}
```

Create a color option.

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

Make the background color of the notes as an option.

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

Make the line color of the notes as an option.

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

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

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

Make the text width as an option.

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

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{todonotes.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{todonotes.sty}%
    {figwidth}{\renewcommand{\@todonotes@figwidth}{#1}}
```

Make the text width as an option.

```
% Finally process the given options.
%    \begin{macrocode}
\ProcessOptionsX*
```

If the `obeyDraft` is given, check whether the `draft` option is given and enable or disable the functionality of this package. 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
\fi
```

2.2 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}}%
```

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}%
```

Should the connection between note and insertion point be drawn in a fancy way? (does only work if line=true)

```
\newif\if@todonotes@fancyline\@todonotes@fancylinefalse%
\define@key{todonotes}{fancyline}[]{\@todonotes@fancylinetrue}%
\define@key{todonotes}{nofancyline}[]{\@todonotes@fancylinefalse}%
```

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.

```
\newcommand{\@todonotes@currentfigwidth}{\@todonotes@figwidth}
\define@key{todonotes}%
    {figwidth}{\renewcommand{\@todonotes@currentfigwidth}{#1}}
```

Preset values of the options

```
\presetkeys%
    {todonotes}%
```

```
    {linecolor=\@todonotes@linecolor,%
    backgroundcolor=\@todonotes@backgroundcolor,%
    bordercolor=\@todonotes@bordercolor,%
    nofancyline,%
    nodisable,%
    noinline,%
    nocaption,%
    figwidth=\@todonotes@figwidth,%
    line, list, size=\@todonotes@textsize}{}%
```

2.3 The main code part

Here is the actual macros defined. If the option "disable" was passed to the package define empty commands.

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

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

```
\newcommand{\listoftodos}[1][\@todonotes@todolistname]
    {\ifdefined\chapter\chapter*{#1}\else\section*{#1}\fi \@starttoc{tdo}}
\newcommand{\l@todo}
    {\@dottedtocline{1}{0em}{2.3em}}
```

Define styles used by the todo command

```
\tikzstyle{notestyleraw} = [
    draw=\@todonotes@currentbordercolor,
    fill=\@todonotes@currentbackgroundcolor,
    line width=0.5pt,
    text width = \@todonotes@textwidth - 1.6 ex - 1pt,
    inner sep = 0.8 ex,
    rounded corners=4pt]
```

Add shadows and rounded corners to the inserted todonotes.

```
\if@todonotes@shadowenabled
\tikzstyle{notestyle} = [notestyleraw,
    general shadow={shadow xshift=.5ex, shadow yshift=-.5ex,
        opacity=1,fill=black!50}]
\else
\tikzstyle{notestyle} = [notestyleraw]
\fi
\tikzstyle{notestyleleft} = [
    notestyle,
    left]
\tikzstyle{connectstyle} = [
    thick,
    draw=\@todonotes@currentlinecolor]
\tikzstyle{inlinenotestyle} = [
```

```
    notestyle,
    text width=\linewidth - 1.6 ex - 1 pt]
```

`\@todo` Define the `\@todo` command.

```
\newcommand{\@todo}[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
\ignorespaces%
}%
```

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.

```
\begin{tikzpicture}[remember picture, baseline=-0.75ex]%
    \node [coordinate] (inText) {};%
\end{tikzpicture}%
\marginpar[{% Draw note in left margin
    \@todonotes@drawMarginNote%
    \@todonotes@drawLineToLeftMargin%
```

In the book documentclass (which is a twoside layout), the \marginpar marginpar macro takes two arguments \marginpar[left]{right}. If both arguments are given, latex will decide in which side the margin note has to be inserted, and then use the corresponding commands.

```
}]{% Draw note in right margin
    \@todonotes@drawMarginNote%
    \@todonotes@drawLineToRightMargin%
}%
}%
```

addElementToListOfTodos Define helper function addElementToListOfTodos.

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

drawInlineNote Define helper function drawInlineNote.

```
\newcommand{\@todonotes@drawInlineNote}{%
    \if@todonotes@dviStyle%
        {\par\noindent\begin{tikzpicture}[remember picture]%
            \draw node[inlinenotestyle] {};\end{tikzpicture}\par}%
        {\noindent \@todonotes@sizecommand \@todonotes@text}%
        {\par\noindent\begin{tikzpicture}[remember picture]%
            \draw node[inlinenotestyle] {};\end{tikzpicture}\par}%
    \else%
        {\par\noindent\begin{tikzpicture}[remember picture]%
            \draw node[inlinenotestyle,font=\@todonotes@sizecommand] {\@todonotes@text};%
            \end{tikzpicture}\par}%
    \fi}%
```

drawMarginNote Define helper function drawMarginNote.

```
\newcommand{\@todonotes@drawMarginNote}{%
\if@todonotes@dviStyle%
    \begin{tikzpicture}[remember picture]%
        \draw node[notestyle] {};%
    \end{tikzpicture}\\ %
    \begin{minipage}{\@todonotes@textwidth}%
    \@todonotes@sizecommand \@todonotes@text%
    \end{minipage}\\%
    \begin{tikzpicture}[remember picture]%
        \draw node[notestyle] (inNote) {};%
    \end{tikzpicture}%
\else%
    \begin{tikzpicture}[remember picture,baseline=(X.base)]%
     \node(X){\vphantom{X}};%
        \draw node[notestyle,font=\@todonotes@sizecommand,anchor=north] (inNote) at (X.north)%
            {\@todonotes@text};%
    \end{tikzpicture}%
\fi}%
```

drawLineToRightMargin Define helper function drawLineToRightMargin.

```
\newcommand{\@todonotes@drawLineToRightMargin}{%
\if@todonotes@line%
\if@todonotes@fancyline%
\tikz[remember picture,overlay]{%
\tikzstyle{both}=[line width=3pt, draw, opacity=0.15]%
\tikzstyle{line}=[shorten >=5pt, line cap=round]%
\tikzstyle{head}=[shorten >=-1pt, dash pattern=on 0pt off 1pt, ->]%
\foreach \s in {line,head}{%
\draw[both,\s]%
(inNote.north west).. controls +(0:0) and +(90:1.5)..([yshift=1ex] inText);%
};%
}%
\else%
\begin{tikzpicture}[remember picture, overlay]%
\draw[connectstyle]%
([yshift=-0.2cm] inText)%
-| ([xshift=-0.2cm] inNote.west)%
-| (inNote.west);%
\end{tikzpicture}%
\fi
\fi}%
```

drawLineToLeftMargin Define helper function drawLineToLeftMargin.

```
\newcommand{\@todonotes@drawLineToLeftMargin}{
\if@todonotes@line%
\if@todonotes@fancyline%
\tikz[remember picture,overlay]{%
\tikzstyle{both}=[line width=3pt, draw, opacity=0.15]%
\tikzstyle{line}=[shorten >=5pt, line cap=round]%
\tikzstyle{head}=[shorten >=-1pt, dash pattern=on 0pt off 1pt,->]%
```

```
\foreach \s in {line,head}{%
\draw[both,\s]%
(inNote.north east).. controls +(0:0) and +(90:1.5)..([yshift=1ex] inText);%
};%
}%
\else%
\begin{tikzpicture}[remember picture, overlay]%
\draw[connectstyle]%
([yshift=-0.2cm] inText)%
-| ([xshift=0.2cm] inNote.east)%
-| (inNote.east);%
\end{tikzpicture}%
\fi%
\fi}
```

`\missingfigure` Defines the `\missingfigure` macro.

```
\newcommand{\missingfigure}[2][]{
\setkeys{todonotes}{#1}%
\addcontentsline{tdo}{todo}{\@todonotes@MissingFigureText: \protect{#2}}%
\par
\noindent
\begin{tikzpicture}
\draw[fill=black!40, draw = white, line width=0pt]
    (-2, -2.5) rectangle +(\@todonotes@currentfigwidth, 4cm);
\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}
}% 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][]{\@todo[#1]{#2}}
```

Change History