`todo` package for appending a *to-do* list to a document

Federico Garcia
`federook@gmail.com`

2010/03/31

Abstract

`todo` provides commands for the typesetting of a *to-do* list in any document, with a customizable format. This is version v2.4142, released on 2010/03/31.

A sample file accompanies this release (`todo-spl.tex`). It contains all the different possibilities and should be useful as a quick usage guide.

Contents

1 Preface to version 2.4142

The present version of `todo` corrects some bugs and adds some features. Thanks to all the users who wrote to me in the last few years with these suggestions. I'm finally getting around to putting it together.

1.1 Correction reference

These are bugs of version 1.1 of `todo`:

1. There is a spurious space after the superscript marks. **Solved.**
2. `\todo` gives an error if it's the first thing on a line. **Solved.**
3. `\todos` complains when there have been no `\todo` commands. **Solved.**
4. The todo counter confuses the `backref` package.

These have been taken care of in the present version (2.4241). The `backref` problem is actually there with any referrable counter, not only `todo`'s, and for example a `\label` to a figure or an equation will also confuse `backref`.

1.2 Addition reference

New features include:

1. A quiet `\todo*` command that adds an item to the todo list, but makes no fuss in the text.
2. A verbose `\Todo` command that puts its text within the running text.
3. A `\done` command that indicates the thing `\todo` was already done, allowing to keep it in the document for reference.
4. A `todoenv` environment for additional formatting options.

2 Basic usage

See also the accompanying sample file, `todo-spl.tex` and `todo-spl.dvi`.

2.1 Basic behavior

A ‘todo’ has two main components: its text (whatever it is that is yet to be done), and a mark, that identifies the point in the document where the todo belongs. The most basic macros of the `todo` package gather the text of several todos, and compile a list of them, to be printed at the end of the document, while at the same time putting the marks in the document itself.

By default, the mark is ‘**To do**’. By default, it is typeset as a superscript at the point the todo is invoked, and it is followed by the todo number, which identifies it in the list of todos. (The todo number is a `\ref`, pointing to the item in the todo list at the end of the document; if running `hyperref`, it will be an interactive link.) The actual text of the todo is put in the list of todos, at the end of the document.

`\todo` That is what happens when the user inserts `\todo{`⟨*todo text*⟩`}` in a document. Additional commands and alternatives provide for variations of this basic behavior.

2.2 Display options

`marginpar` The todos’ marks show up as superscripts by default, but they can also be put as marginal notes (`\marginpar`’s). That may be more useful if one doesn’t want the todos to interfere with the layout of the text. In order to do this, the package should be loaded with `marginpar` option: `\usepackage[marginpar]{todo}`.

2.3 Custom marks

`\todo[]` The default text for a mark is ‘**To do**’, but `\todo` has an optional argument for different marks. In `\todo[Fix]{To fix}`, the mark within the document will be ‘Fix’, not ‘To do’. (It will still be followed by the number of the todo.) The todo text (“To fix”) will go to the final list of todos, as usual, and will also include the custom mark.

`\todoformat` All marks, the default or the optional one, are formatted by `\todoformat`. This is by default equal to `\bfseries`, but it can be redefined with any other declaration(s) for font, size, alignment, etc.

`\todomark` The text of the default mark itself can be changed by redefining `\todomark`, whose default value is ‘To do’.

2.4 Quiet todos

`\todo*` It is possible to omit the mark of a todo, so as to not affect text layout at all, with the starred command `\todo*`. It takes the same arguments as the regular `\todo` (including the optional argument), and it puts the todo text in the todo list at the end, but it does nothing to the actual text of the document.

`nothing` `hide` If this is the desired behavior for all todos, option `nothing` can be loaded (`\usepackage[nothing]{todo}`) and all todo commands will be quiet. Another option, `hide`, will cause not only all `\todo`’s, but also the final list of todos, to disappear.

2.5 Verbose todos

`\Todo` On the other hand, it is possible to make todos that include their text (what is yet to be done) within the document's text, as part of the todo mark. This is achieved with uppercase `\Todo`. Once again, it takes an optional argument for its mark (if other than 'To do'). The text of the todo will be appended both in the todo list *and* in the running text of the document, either as a superscript or a marginpar (according to package options), and following the mark and todo number.

`\Todo*` A concoction like `\Todo*` ('quiet verbose todo') is not too sensible, but it is provided for completeness. It will behave as a quiet `\todo*`, and issue a package warning.

3 Done todos

`\done` When a todo (whatever had to be done) has been done, the user can simply delete the todo command to remove it from the document. But sometimes it is a good idea to leave the todo information in the input file, for reference, and also to keep the original todo numbering. The command `\done` is a convenient way of dealing with this situation. Any `\todo` (or `\Todo`) can be preceded by `\done`, and then its behavior changes: it won't put anything in the text, but the todo will be numbered and added to the list of todos at the end (with the difference that its box will be checkmarked.)

A `\done` command that is not followed by a `\todo` or a `\Todo` will have no effect, and will issue a package warning ('`floating \done`...').

`\done[]` The `\done` command itself has an optional argument, a ⟨*done text*⟩ intended for notes with the date it was 'done', or the initials of whoever 'did it'. So:

`\done` Text text `\todo`: This creates a warning, but the `\done` is otherwise ignored.

`\done\todo{Fix}`: This will checkmark the box of the 'Fix' item in the *to-do* list.

`\done[By me]\todo{Fix}`: The item in the list will be checkmarked and have 'By me' added to it.

4 For full flexibility: the `todoenv` environment

`todoenv` There is a different kind of todo that is supported by the package. Instead of a reminder for a quick fix ('get bibliography,' 'complete the proof,' or stuff like that), intended mainly for private use, a todo can be conceived of as part of the document—like in semi-public reports of on-going projects or developing software, describing future features or project steps. On the other hand, a verbose todo (`\Todo` above) may be problematic when its text runs for longer than one line, in which case it would be better just to surround it with todo marks instead of to put it all *in* a mark.

The `todoenv` environment is designed for these cases and to provide further options:

`\begin{todoenv}`⟨*extended todo text*⟩`\end{todoenv}`.

The default behavior of this construction is to typeset the ⟨*extended todo text*⟩ normally, within the document, but inserting a superscript (or a marginpar, according to package options) at the beginning ('To do begin') and one at the end ('To do end (n)').

`\todoopen` `\todoclose` `\astodos` But this can be modified with a number of commands. `\todoopen` holds the contents of the beginning mark, and `\todoclose` the contents of the ending mark. (The definitions of these two commands can use `\astodos{`⟨*mark text*⟩`}`, so that they behave as a superscript or as a marginpar, or even as nothing, according to the options of the package.)

`\todoenvformat` The text of the todo is formatted by `\todoenvformat`, that can contain standard declarations (font, size, alignment, etc.). (`\todoenvformat` is initially empty.)

The ⟨*todo text*⟩ of a `todoenv` environment is not quoted in the final todo list. The respective item for the todo will (like regular todos) have a reference to the number and page of the todo, followed by `see text`.

5 The final list of todos

`\todos` Command `\todos` prints the final list that gathers the todo text of all the todos (except `todoenv` environments, that simply refer to the text). Each item has a box, checkmarked when `\done`, the todo number, and a reference (and interactive link if `hyperref` is on) to the page where it appears.

`\todoname` By default, the list appears in a new page, under the heading '**To do. . .** '. This can be modified by redefining `\todoname`.

After `\todos`, all todo mechanisms are disabled. The list is really intended to be the last thing in the document.

6 Reference

6.1 Package options

`marginpar` `superscript` `nothing` The option `marginpar` makes the mark to appear not as a superscript, but as a margin par, like in the next paragraph. The option `superscript`, selected by default, makes it appear as a superscript, as explained in subsection **??**. A third option is `nothing`, which prevents `\todo` from insert anything in the text, while still appending entries to the list.

`hide` Another options is `hide`, which causes all `\todo`'s and `\todos` to be ignored (they will only produce a warning). Unlike `nothing`, `hide` will also omit the final list of todos.

6.2 Cross referencing

The `\todo` command creates internal labels and references, so that each entry in the list has a correct reference, and potentially a hyperlink, to the page in which the *to-do* was executed.

`\label`

The user can add his own `\label` to any `\todo`. The matching `\ref` will make reference to the number of the *to-do*. However, `\pageref` will lead to the page in which the `\todo` occurred, unless the `\label` has been put *inside* the ⟨*text*⟩ (and then the pageref points to the actual text of the `\todo`, as is probably more desirable). Another reason to put it there is that a `hyperref` link will point to the actual text of the *to-do* (not to the mark).

7 Implementation

7.1 Identification

```
%<*package>
\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\ProvidesPackage{todo}[2010/03/31 v2.4142 To-do list (Federico Garcia)]
\RequirePackage{amssymb}
```

7.2 Variables

`\@todotoks`
`\if@todo`

`\@todotoks` is the token register that will store the entries of the list.The `\if@todo` test is used by the `\done` mechanism.

```
\newtoks\@todotoks\@todotoks{}
\newif\if@todo\@todotrue
\newcounter{todo}\setcounter{todo}{0}
\newcommand*\todomark{To~do}
\newcommand*\todoname{To do\dots}
```

7.3 Basic workings and package options

A todo consists of two processes: the *gathering* stage, that manages the todo's information (counter, cross referencing, and the addition of the ⟨*todo text*⟩ to the final list), and the *display* stage, that inserts the appropriate mark for the todo in the running text.

The first process is in the main shared by all the different versions of the todo commands. The second process, however, depends on the current options, and on the exact variety of todo command that is used. This is handled by the wild-card command `\@tododisplay`, whose meaning can be `\textsuperscript`, `\marginpar`, or a package warning, according to the options.

```
\newcommand\@todomark{}
\let\@todohide\relax
\let\todoformat\bfseries
\DeclareOption{nothing}{\renewcommand*\@tododisplay[1]{}}
\DeclareOption{hide}{\let\@todohide\@gobble
```

```
   \renewcommand*\@tododisplay[1]{%
   \PackageWarning{todo}{‘hide’ option used, ignoring \string\todo’s}}}
\DeclareOption{superscript}{\let\@tododisplay\textsuperscript}
\DeclareOption{marginpar}{\renewcommand*\todoformat{\sffamily\raggedright\small}%
   \let\@tododisplay\marginpar}
\DeclareOption*{\typeout{Unknown option (‘\CurrentOption’)}}
\ExecuteOptions{superscript}
\ProcessOptions
```

7.4 The final list

`\todos` The list of todos is a `list` environment, called by `\todos`. The actual contents of the list is provided by the token register `\@todotoks`, that has been compiled by the different `\todo` commands (and its variations) throughout the document.

The list is subject to `\@todohide`, that is `\@gobble` if the package was loaded with option `hide` (otherwise it’s `\relax`).

After typesetting the list, all user commands are redefined to a package warning (no todos are supposed to come after the list of todos). `\todos` itself is made an error; the user should really be called attention to the duplication.

```
\newcommand\todos{%
   \ifnum\c@todo>0
      \@todohide{%
      \clearpage\section*{\todoname}\large%
      \begin{list}{$\Box$%
            \quad\arabic{todo}}{}%
         \usecounter{todo}
         \the\@todotoks
      \end{list}}
      \renewcommand*\todo{%
         \PackageWarning{todo}{All \string\todo\space commands are ignored after
            \string\todos}%
         \@ifnextchar*{\@todogobble}{\@todogobble*}}%
      \let\Todo\todo
      \renewenvironment{todoenv}{\PackageWarning{todo}{\string\todoenv\space after
         \string\todos\space not processed as a todo item}}{}%
      \renewcommand\todos{\PackageError{todo}{Second \string\todos\space ignored}%
         {I can only make one list of todo items. I’ll be Ok if you press enter,
         but all \string\todo\space commands after the first \string\todos\space
         have been ignored.}}%
   \else
      \PackageWarning{todo}{Nothing \string\todo! I’m ignoring the \string\todos\space
         command}%
   \fi
   }
\newcommand\@todogobble[1]{\@@todogobble}
\newcommand\@@todogobble[2][\todomark]{\relax}
```

The actual items of this list are added by the `\todo` commands to the `\@todotoks` register, and they can be of one of two kinds: a regular todo, and a todo that was done. The several arguments for the respective commands handle the user's optional arguments. (They have been converted to regular arguments by the `\todo` commands themselves, so they are all explicit in `\@todotoks`.)

`\todoitem`

```
\newcommand\todoitem[2]{%
    \item \label{todolbl:\thetodo}%
    (p.~\pageref{todopage:\thetodo}):
    {\todoformat\ifx#1\todomark\else#1 \fi}#2}%
```

`\doneitem` For `\doneitem`, what was the optional argument to `\done` (not to the `\todo` commands) has been actually put in a control sequence, whose name is `\@done`⟨*todo number*⟩ (something like `\@done12`.

```
\newcommand\doneitem[2]{%
    \stepcounter{todo}%
    \item[\rlap{$\checkmark$}$\Box$\quad
        \arabic{todo}]\@nameuse{@done\the\c@todo}(p.~\pageref{todopage:\thetodo}):
        {\todoformat\ifx#1\todomark\else#1 \fi}#2}
```

7.5 The `\done` mechanism

`\done` A few things are done by `\done`: first, it has to check that it is actually followed by a `\todo` of some sort, in which case it sets `\@todofalse` (so that the coming `\todo` command knows that it is `\done`).

```
\newcommand\done[2][\relax]{%
    \ifx#2\todo\@todofalse\else
        \ifx#2\Todo\@todofalse\else
        \PackageWarning{Floating \string\done\space ignored.}%
    \fi\fi
```

Then the optional argument is used to define a command `\@done`⟨*todo number*⟩, which will be used by `\doneitem` in the list of todos. There will be a space after the optional argument; if the user does not specify it, then `#1` is `\relax`, and the space after will not be typeset (since it comes after a command).

```
    \@tempcnta\c@todo\advance\@tempcnta1
    \@namedef{@done\the\@tempcnta}{#1 }#2%
    }
```

7.6 User commands

The user commands are actually driver commands that don't even deal with the arguments. They decide which executive functions to delegate on. Most of them call the `\@newtodo` routine:

```
\newcommand\@newtodo{\refstepcounter{todo}\label{todopage:\thetodo}}
```

`\todo` Regular `\todo` makes a new todo, and then: if starred, it displays nothing (although it still has to process the argument(s)); if not starred, it displays the regular todo mark—provided `\if@todo` was not set to `false` by a preceding `\done`.

```
\newcommand*\todo{\@ifnextchar*{\@newtodo\@displaynothing}{%
    \@newtodo
    \if@todo
        \expandafter\@displaytodo
    \else
        \expandafter\@donetodo
    \fi}}
```

`\Todo`

The verbose `\Todo` invokes `\@displayfulltodo` (unless it's starred, in which case it delegates on regular `\todo`):

```
\newcommand\Todo{\@ifnextchar*{%
    \PackageWarning{todo}{Starred \string\Todo* taken as\string\todo*}%
        \expandafter\todo}%
    {\@newtodo\@displayfulltodo}}
```

`todoenv`

The definition of the `todoenv` environment contains all the tasks of a todo: setting up a new one (`\@newtodo`), typesetting marks and text, and, unlike the previous commands, also putting a list item in `\@todotoks`.

```
\newenvironment{todoenv}{\@newtodo
    \global\@todotoks\expandafter{\the\@todotoks\relax\todoitem
        {}{{\itshape see text.}}}%
    \todoopen\todoenvformat}{\todoclose}
```

`todoenv` makes use of some placeholder commands for the marks and the text formatting. Here is their default definition (`\astodos` is used to signify either superscript or marginpar.

```
\newcommand*\astodos[1]{\@tododisplay{{\todoformat #1}}}
\newcommand*\todoopen{\astodos{\todomark\ begin}}
\newcommand*\todoclose{\astodos{\todomark\ end
    \normalfont(\ref{todolbl:\thetodo})}}\newcommand\todoenvformat{}
```

7.7 The display commands

`\@displaytodo`

Regular todos put a mark in the text:

```
\newcommand\@displaytodo[2][\todomark]{%
    \@tododisplay{{\todoformat #1} (\ref{todolbl:\thetodo})}%
    \global\@todotoks\expandafter{\the\@todotoks\todoitem{#1}{#2}}%
    \@todotrue
    }
```

`\@displaynothing`

When nothing is to be displayed, the arguments still have to be appended to the final list. The routine is delayed because the star of `\todo*` (the command that eventually is the one to call `\@displaynothing`) has to be gobbled.

```
\newcommand\@displaynothing[1]{\@@displaynothing}
\newcommand\@@displaynothing[2][\todomark]{%
    \if@todo
        \global\@todotoks\expandafter{\the\@todotoks\todoitem{#1}{#2}}%
    \else
        \global\@todotoks\expandafter{\the\@todotoks\doneitem{#1}{#2}}%
    \fi
    \@todotrue
    }
```

`\@displayfulltodo` For verbose todos, `\@displayfulltodo`:

```
\newcommand\@displayfulltodo[2][\todomark]{%
    \if@todo
        \@tododisplay{{\todoformat #1} #2 (\ref{todolbl:\thetodo})}%
        \global\@todotoks\expandafter{\the\@todotoks\todoitem{#1}{#2}}%
    \else
        \global\@todotoks\expandafter{\the\@todotoks\doneitem{#1}{#2}}%
    \fi
    \@todotrue
    }
```

`\@donetodo` And for `\done` todos:

```
\newcommand\@donetodo[2][\todomark]{%
    \global\@todotoks\expandafter{\the\@todotoks\doneitem{#1}{#2}}%
    \@todotrue
    }
```

Note that all these commands end by resetting `\@todotrue`. This ensures that the next todo is, by default, not `\done`.

```
%</package>
```