The NotesPages Package

Filling documents, so the total number of pages is a multiple of a given number.

Mike Kaufmann
m.km@gmx.de

2016/08/21 (v0.8.1)

Abstract

The NotesPages package provides one macro to insert a single notes page and another to fill the document with multiple notes pages, until the total number of pages (so far) is a multiple of a given number. A third command can be used to fill half empty pages with a notes area.

Contents

1 Introduction

1.1 Why this Package

Well, sometimes I have to write short manuals, which are then printed as a booklet. Therefore their number of pages has to be dividable by 4. And since it's tiresome to check the number of pages after every change and add or remove notes pages manually, I automated this work. In this, I had to take into account that there are a number of fixed pages at the end of the booklet.

In order to make it useful for others, options for formating a notes page were added.

1.2 Feeback and Testing

The NotesPages package was tested with the example file provided with this package. There may be documents, where the algorithm for calculating the number of pages needed to fill a document may not work properly. Or there may be other issues. If you have such a document, please send me an example, which shows the problem, so I can try to fix it.

1.3 Dependencies

1.3.1 Necessary Packages

The NotesPages package needs the xkeyval package for processing the options.

1.3.2 Babel

The package makes use of babel, if it is loaded, but it doesn't depend on it. The order, in which the package is loaded in regard to the NotesPages package, is of no consequence. More on supporting languages can be found in subsection 2.6.

1.3.3 Color

The package also makes use of the package color or xcolor, if loaded, without depending either one. Again, the order, in which the package is loaded in regard to the NotesPages package, is of no consequence. More on supporting color can be found in subsection 2.7.

1.4 Legal Stuff

This program is provided under the terms of the LaTeX Project Public License distributed from CTAN http://www.ctan.org/license/lppl1.3

2 Using the Packages

2.1 Loading

The package is loaded as usual.

`\usepackage[`⟨*options*⟩`]{notepages}`

The options are described in subsection 2.4.

2.2 The basic commands

`\setnotespages` With `\setnotespages{`⟨*options*⟩`}` all the settings possible with the options described in subsection 2.4 can be changed globaly. This will overwrite the settings made with the options provided on loading the package or the defaults.

`\notespage` With `\notespage[`⟨*options*⟩`]` a single notes page will be inserted into the document. For this, a new page will be started at the place of its occurrence. The command has one optional parameter, which can contain all the options described in subsection 2.4. But the options of subsubsection 2.4.3 and subsubsection 2.4.4 are ignored, because they are of no use for `\notespage`.

The options are local to the command, i.e. they will not change the settings done with `\setnotespages`, made at loading the package, or the defaults. Thus, if the next notes page should have the same appearance, the same options must be given.

`\notespages` With `\notespages[`⟨*options*⟩`]` the document can be filled with notes pages, so the total number of pages so far is a multiple of a given number. Depending on the settings done with the options described in subsubsection 2.4.3, this may result in anything from 0 to 199 notes pages.

All options described in subsection 2.4 can be used in the optional parameter. But the options in subsubsection 2.4.4 are ignored, because they are of no use for `\notespages`.

The command first sets up everything according to the given options. It then calculates the amount of pages needed and inserts them.

For `\notespages` the options are also local to the command.

`\notesfill` With `\notesfill[`⟨*options*⟩`]` a page can be filled with a notes area preceeded

by the notes tile. It is formated similar to a notes page, but no new page is started, the pagestyle can not be changed and so can't the text in the header .

All options described in subsection 2.4 can be used in the optional parameter. But the options of subsubsection 2.4.3 are useless to `\notesfill` and therefore ignored. Also the options startnotes, pagestyle, mark, marktext, and markuppercase from subsubsection 2.4.2 are ignored.

With the options described in subsubsection 2.4.4 it is possible to set a minimum height for `\notesfill`, i.e. no notes title plus notes area will be generated, if the remaining space on the page is less then the minimum. Also a maximum height for the notes title plus the notes area can be defined. If there is more space left on the page, the notes title plus notes area will be moved to the bottom of the page by default.

And again, the options given to `\notesfill` are local to the command.

There is an restriction to `\notesfill` regarding bottom floats and footnotes: they will appear below it.

2.3 Layout

Figure 1 shows the layout of a notes page and Figure 2 the layout of a notes fill, together with some of the layout options described in subsection 2.4. The dimension `\remainingtextheight` is described in subsection 2.5.

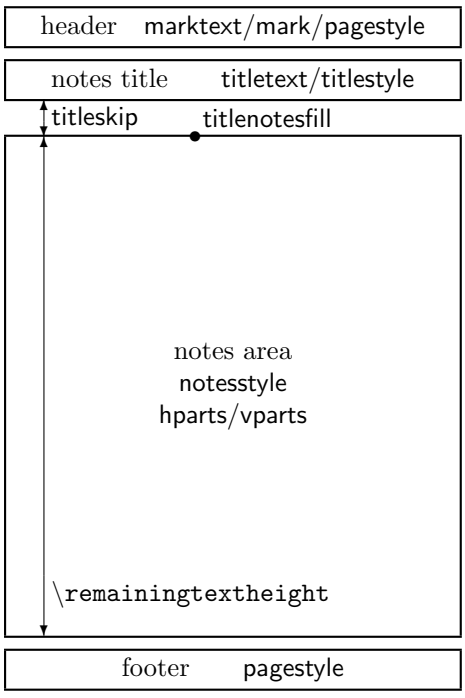

Figure 1: Layout notes page

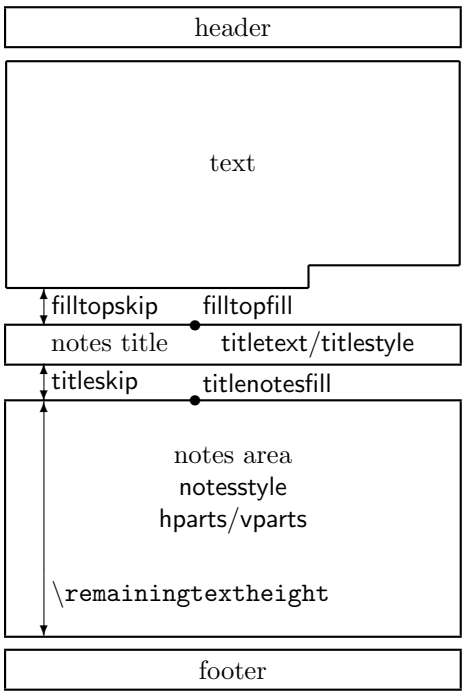

Figure 2: Layout notes fill

2.4 Package and Command Options

2.4.1 Setting Options

If options are used without a value, they will be set to their defaults. The exception are the boolean options allowfloats, usenotesareaheight, titlenotesfill, markuppercase, and filltopfill, which are set to *true*.

2.4.2 Options for `\notespage`

The following options affect the style and layout of a single notes page. They are all used by `\notespage` and `\notespages`. And most of them are also used by `\notesfill` (the exceptions are startnotes, pagestyle, mark, marktext, and markuppercase).

`startnotes` The option startnotes defines, how the new page for `\notespage` is started. The choices are *newpage* and *clearpage*. The default is *clearpage*.

With *newpage* the command `\newpage` is used and therefore remaining floats are not given out by default, which can be changed with the option allowfloats. With *clearpage* `\clearpage` is used and remaining floats are given out before the notes page.

`allowfloats` If the boolean allowfloats is set to *true* floats are allowed to be placed on a notes page, if *newpage* is used for the option startnotes. Caution: the header will be changed on such pages. The default is *false*.

`pagestyle` With pagestyle the pagestyle for a notes page can be defined. All possible pagestyles (*empty*, *plain*, *headings*, *myheadings*, and what ever is defined for the document) can be use as value. Additionally, *current* can be used to denote to not change the pagestyle. This is the default. Internally `\thispagestyle` is used, but not for the value *current*.

A pagestyle must have been defined, before it can be chosen. If a nonexisting pagestyle is chosen, a warning will be given and the value will be set to the default. Therefore, `\setnotespages` must appear after defining a page style to be used for notes pages in the preamble.

`notesstyle` With notesstyle the way the notes area is filled can be chosen. Possible are *plain*, *lines*, *vlines*, *grid*, and *text*. The default is *grid*.

With *plain* the notes area is left empty, *lines* will fill the notes area with horizonal lines, *vlines* with vertical lines, *grid* will fill it with a grid, and *text* will give out a short text.

It is possible to add your own notesstyle. This is described in subsection 2.5.

`hparts` The option hparts divides the notes area into the given number n_x horizontal parts, which will be seperated by vertical lines. There will always be $n_x + 1$ lines. The default value is 25.

The option applies to the notesstyles *vlines* and *grid*. The value may be in the range from 1 to 200. If the given number is smaller or larger, a warning is given out and it's set to either 1 or 200 respectively.

`vparts` The option vparts divides the `\textheight` into the given number n_y vertical parts. The calculation is based on `\textheight` by default (see below, option usenotesareaheight), in order to achieve same height vertical parts, independent of

the actual height of the notes area. The option applies to the notesstyles *lines* and *grid*.

The value may be in the range from 0 to 300. If the given number is smaller or larger, a warning is given out and it's set to either 0 or 300 respectively.

The values 0 and 1 have special meaning. With 0 for the notesstyle *grid* the length of a vertical part will be the same as for a horizontal part, thus resulting in a square grid. For the notesstyle *lines* a warning will be given and the notes area is left empty. A value of 1 will lead to one line at the bottom and one at the top of the notes area, regardless of its height, thus making it possible to put a rectangle around the notes area.

For values of 2 or greater only lines for full vertical parts are drawn. For example, if the height of a notes area is 20.5 times the length of a vertical part, for *lines* 20 lines are drawn and 21 for *grid*. For small values there may be no lines in the notes area of a `\notesfill`.

The default value is 0, so for the notesstyle *lines* it has to be changed.

`usenotesareaheight` With the option usenotesareaheight the calculation of the height of a vertical part (see above, option vparts) is based on the height of the notes area instead of `\textheight`. This enables the user to vertically divide the notes area into the exact number given to vparts. Of course with this, for a `\notesfill` the height of a vertical part can differ each time. The default is *false*.

`titlestyle` With the option titlestyle a layout for the notes title can be chosen. Possible are *none*, *text*, *section*, *subsection*, *subsubsection*, and, if available, *minisec*. The default is *section*.

With *none* no title is set. And *text* formats it as simple text. With *section*, *subsection*, or *subsubsection* one of the commands `\section*`, `\subsection*`, or `\subsubsection*` is used. The choice *minisec* is only available, if `\minisec` is defined (before loading the NotesPages package). Then, if chosen, `\minisec` is used to format the title.

It is possible to add your own titlestyle. This is described in subsection 2.5.

`titletext` With titletext an arbitrary text can be chosen as new a notes title. If the new text contains more then one word, it is recommended to put the text in braces, e.g. `titletext={My new notes title}`. If the text contains a comma (",") or an equality sign ("="), it must be given in braces! The default is `\npnotesname`, which is "Notes" without babel or language dependend with babel (see subsection 2.6 for details).

The option titletext will automatically also set marktext to its value, if marktext is not given, but not otherwise, regardless which option comes first. So if a long text is set with titletext, it is recommended to also use marktext to set a shorter text suitable for the headers. Of course, if mark is set to *keep*, this is not necessary.

`titleskip` With titleskip the distance between the notes title and the notes area can be set. The value can be everything accepted as a length by TEX. The default is 0pt. For the default title style this is ok, because `\section*` adds some space after the title. But for the title style *text* it is recommended to set a titleskip greater then 0pt.

`titlenotesfill` If the boolean titlenotesfill is set to *true*, a `\vfill` will be inserted between

notes title and notes area, moving the latter to the bottom of the page. The default is *false*.

Most of the provided notes styles always use the whole remaining space of a page, so the option is of no use for them. The exception is the notes style *text*, which has the height of the text. But this option would move the text to the end of the page. It may be useful for a custom notes style, which doesn't use all the available space, and the notes area should be moved down to the end of the page.

`notestext` With notestext an arbitrary text can be chosen as a new text for the notes style *text*. If the text contains more then one word, it is recommended to put the text in braces, e.g. `notestext={This page is empty.}`. If the text contains a comma (",") or an equality sign ("="), it must be given in braces! The default is `\npnotestext`, which is "This page is intentionally left blank." without babel or language dependend with babel (see subsection 2.6 for details).

`notestextalign` With notestextalign the horizontal alignement for the text of the notes style text can be choosen. Possible are *right*, *left*, *center*, and *none*. The default is *center*.

For *none* no alignement is set. Thus the text is aligned the same as normal text in the document.

Vertical alignment can be done using the option titleskip. For the notes style *text* is is also inserted, if the title style is *none*.

`mark` With mark the way the notes title is put into the header can be chosen. Possible are *both*, *right*, *left*, and *keep*. The default is *both*.

For *both*, *right*, and *left* the command `\markboth` is used, but for *right* and *left* the original mark is set for the other side. With the choice *keep* the headers are not changed.

Note, in order to get the header marks right, it is necessary to run LaTeX twice.

`marktext` The option marktext can be used the set an arbitrary text for the headers, which differs from the notes title. The latter is the default. Here too, the text should be given in braces if it contains more then one word and it must be given in braces, if it contains a comma (",") or an equality sign ("=").

Note, if notestitle is given locally, it will also set marktext locally. Therefore, if both texts should be different, both options must be used.

`markuppercase` If the option markuppercase is used, the text for the header marks set by a notes page is converted to upper case letters. The default depends on the class used. For the standard classes and memoir the option is set to *true*, for others to *false*.

2.4.3 Options for `\notespages`

The following options are only used by `\notespages` to determie the number of notes pages to be inserted.

`multiple` With multiple the number of pages, the total number of pages of a document (so far) should be a multiple of, can be defined. For example, with `multiple=4`, `\notepages` will insert enough pages to make the total number of pages (so far) 4, 8, 12, 16, 20 and so on. The value can be in the range from 1 to 100. If the

given number is below or above that, a warning will be given and the value will be set to either the minimum or maximum respectively. The default is 4.

`minpages` With minpages the minimum number of notes pages to be inserted can be defined. For example, with `multiple=4`, `minpages=1` and `\notespages` appearing on (the not empty) page 20, one page will be added to fulfill the minimum and an additional 3 to make the number of pages a multiple of 4 again, leading to a total of 24 pages. The value can be in the range from 0 to 100. If the given number is below or above that, a warning will be given and the value will be set to either the minimum or maximum respectively. The default is 0.

`endpages` With endpages the number of pages at the end of a document, which are not notes pages, can be defined. For example, the last page of a booklet has to contain contact information and therefore shouldn't be a notes page. By setting `endpages=1` this is taken into account and `\notespages` will fill the document only up to, for example, page 19 instead of 20, thus leaving page 20 free for the desired content. The value can be in the range from 0 to 100. If the given number is below or above that, a warning will be given and the value will be set to either the minimum or maximum respectively. The default is 0.

2.4.4 Options for `\notesfill`

The following options are only used by `\notesfill`.

`fillminspace` With fillminspace the minimum height for a notes fill can be defined, i.e. if the remaining space on a page is less than the given length, no notes fill will appear. The value can be anything accepted as a length by TEX. The default is `0.25\textheight`.

The value given to the option filltopskip is taken into account for the calculation of the remaining space, meaning, it is subtracted from the space left, before the decision is made to insert a notes fill or not.

`fillmaxspace` With fillmaxspace the maximum height for a notes fill can be defined, i.e. if the remaining space on a page is greater than the given length, the height of a notes fill is limited to this length. The value can be anything accepted as a length by TEX. The default is `\textheight`.

`filltopskip` With filltopskip the distance between the text and the notes title can be defined. The value can be anything accepted as a length by TEX. The default is 0pt. For the default value of titlestyle 0pt is ok, since `\section*` inserts some space before the notes title. But for `titlestyle=text` it is recommended to set a filltopskip greater then 0pt.

`filltopfill` If the boolean filltopfill is set to *true*, a `\vfill` will be inserted between the text and the notes title, moving the notes fill to the bottom of the page. The default is *true*. This is useful, if the notes fill is not as high as the remaining space on the page, because it was limited by fillmaxspace.

2.4.5 Meta Option

Basically meta options are option, which set some or all the options described so far. There are three of them.

`empty` The option `empty` sets `pagestyle=empty,notesstyle=plain,titlestyle=none`, which will lead to totally empty pages.

`vacant` The option `vacant` sets `pagestyle=empty,notesstyle=text,titlestyle=none,titleskip=0.3\textheight`, which will lead to pages with only the text "This page is intentionally left blank." on it at about 1/3 of the height from the top.

`default` With `default` all options are set back to their default values. This is useful in `\setnotespages` and all `\notes...` commands to get a defined starting point.

It is possible to define your own meta option (see subsection 2.5 for details).

2.4.6 Order of Options

The order of options is important, if one options is given more then once. In this case the last occurence wins. For example, writing

```
\notespage[hparts=30,vparts=30,hparts=20]
```

will be the same as writing

```
\notespage[vparts=30,hparts=20],
```

because `hparts` appeared twice and `hparts=20` overwrote `hparts=30`.

This is especially important when using meta options, because they set other option, which would overwrite options given before.

2.5 Advanced commands

`\definenotesoption` With `\definenotesoption{`⟨*newopt*⟩`}{`⟨*options*⟩`}` it is possible to define a new meta option ⟨*newopt*⟩, which can then be used as an option for the commands described in subsection 2.2. For ⟨*newopt*⟩ only a single word can be used. In ⟨*options*⟩ all options described so far can be used. For example, with

```
\definenotesoption{box}{titlestyle=none,vparts=1,hparts=1}
\notespages[box]
```

notes pages with only a box on them could be produced. The command is especially useful, if you want to switch between different layouts occasionally throughout the document.

`\definenotesstyle` `\remainingtextheight` `\notesareatext` With `\definenotesstyle{`⟨*newnotesstyle*⟩`}{`⟨*commands*⟩`}` a new notes style can be defined. After that, ⟨*newnotesstyle*⟩ can be used as a new choice for the `notesstyle` option and ⟨*commands*⟩ is used to produce the notes area. For ⟨*newnotesstyle*⟩ only a single word can be used. For ⟨*commands*⟩ the length `\remainingtextheight` is available, containing the height remaining on the page usable for the notes area (see Figure 1 and Figure 2). And if the notes style should contain the text given to the option `notestext`, the macro `\notesareatext` has to be used.

For example, after

```
\definenotesstyle{yellow}{\color{LightYellow}%
    \rule{\textwidth}{\remainingtextheight}}
```

it is possible to get notes pages with a yellow box, covering the whole notes area, by typing

```
\notespages[notesstyle=yellow]
```

`\definetitlestyle` `\notestitletext`

With `\definetitlestyle{`⟨*newtitlestyle*⟩`}{`⟨*commands*⟩`}` a new style for the notes title can be defined. After that, ⟨*newnotesstyle*⟩ can be used as a new choice for the titlestyle option and ⟨*commands*⟩ is used to produce the notes title. For ⟨*newnotesstyle*⟩ only single word can be used. For ⟨*commands*⟩ the command `\notestitletext` has to be used to get the title text set with the option titletext. The commands should start with `\noindent`, in order to prevent the indentation done for a first line of a paragraph. And it must end with `\par` to start a new paragraph for the notes area, unless the used commands already contain it somehow (like, e.g. `\section*`). For example, after

```
\definetitlestyle{boldred}{\noindent\textcolor{red}%
    {\textbf{\notestitletext}}\par}
```

it is possible to get a boldface red notes title, by typing

```
\notespages[titlestyle=boldred]
```

Please note, the styles or options defined with the commands described here, must have been defined before they can be used. It is recommended to use these commands only in the preamble.

`\nppatchchapter`

With `\nppatchchapter{`⟨*options*⟩`}` the command `\chapter` is patched, so it works as if one writes `\notespages[`⟨*options*⟩`]` followed by `\chapter`. By this, the occasional empty page before a new chapter is converted into a notes page.

It is recommended to add at least `multiple=2`. Otherwise there may be up to three notes pages before a new chapter (with the default value). If other notes pages in the document should be formated differently, one can start with the option default. If the argument is left empty, the notes page will be formated with the current settings (the defaults, the package options, or the last options set with `\setnotespages`).

changed 0.8.1

This macro should not be used, if the class uses the option openany, because it will suppress the effect of this class option. Any redefinition of `\chapter`, either by loading a package or manually, must be done before the NotesPages package is loaded, otherwise the redefinition and/or the patching may not be effective.

The command can be used anywhere in the document and it can be used multiple time. This way, the apperance of a notes page before a new chapter can be changed where ever wanted.

The `\chapter` command is only patched, if it exists, so no errors will occur for a class without it.

`\npunpatchchapter`

added 0.8.1

With `\npunpatchchapter` the command `\chapter` is restored to its original meaning, thus there will be no more notes pages before a new chapter after issuing this command.

2.6 Supporting Babel

`\npnotesname`

The NotesPages package defines the macro `\npnotesname` to contain the default

title “Notes”. If babel is used, commands will be added to redefine \npnotesname to the appropriate translation for the chosen language (if available).

\npnotestext Also, NotesPages defines the macro \npnotestext to contain the default text “This page is intentionally left blank.”, which will be redefined to the appropriate translation, if babel is used.

Currently, only the languages English (english, USenglish, american, UKenglish, british, canadian, australian, newzealand), French (french, francais, canadien, acadian), and German (austrian, german, germanb, ngerman, naustrian) are supported. Additional languages will be supported as users provide a translation for the word “Notes” (plural of note, as in “make a note”, German: “Notizen”) and the sentence “This page is intentionally left blank.” (German: “Diese Seite wurde absichtlich leer gelassen.”; French: “Cette page est laissée intentionnellement vide.”).

Until then, you can put

```
\addto{\extras⟨yourlang⟩}{\def\npnotesname{⟨“Notes” translated⟩}%
  \def\npnotestext{⟨“This page ...” translated⟩}}
```

in the preamble of a multilingual document. For documents in one language one could simply put

```
\setnotespages{titletext={⟨“Notes” translated⟩},notestext=
  {⟨“This page ...” translated⟩}}
```

in the preamble. But this would be overwritten, if the option default is used. To circumvent this, one can redefine the macros \npnotesname and \npnotestext instead:

```
\renewcommand{\npnotesname}{⟨“Notes” translated⟩}
\renewcommand{\npnotestext}{⟨“This page ...” translated⟩}
```

2.7 Supporting Color

The NotesPages package uses the colors NotesHColor, NotesVColor, and NotesTextColor for horizontal and vertical lines and the text in the notes area of notes style *text*. They are defined in the \AtBeginDocument hook, but only, if the package color or xcolor is loaded and the colors where not defined yet. This way, it is possible to define them with your own settings in the preamble. The default for all colors is {gray}{0.7}.

2.8 Warnings and Errors

There are 7 package warnings for the options. In most cases wrong values will be set to a resonable value, so compiling the document will work. But of course the result won't be as expected.

If one of the keys is already defined, an error message will be given out. There are two possible reasons for this: a) another package uses the same key or b) NotesPages was loaded twice.

If for a number or a length something illegal is given, there will be the usual error messages from TEX.

If for a choice key an undefined value is given, there will be an error message from the xkeyval package.

Another Warning will be given out, if LATEX should be run again in order to get saving and restoring header marks correct.

3 Example File

Since examples for the NotesPages package would fill this document with a lot of pages, they are outsourced to an example file `np-test.tex` provided with this package. But beware, the resulting file will be very long. The file also contains some examples for defining own title styles and notes styles.

4 Restrictions

The only known restriction applies to `\notesfill`s. If there are bottom floats or footnotes on the page, they will appear below it. This is shown in the example file. Fixing this, will be at least difficult, as it may require rewriting or patching the output routine. And there are many reasons not doing this.

Of course, there may be other restrictions or incompatibility with some packages, but none were noticed with the packages used for the example file.

5 Testing

The example file (see section 3) was also used for testing. It was compiled several times with different lines commented out. Please refer to the comments in the example for further information.

6 ToDo

There is just two items on the todo list:

- add support for other languages, as translations drop in, and
- find a way to circumvent the restriction described in section 4 (if feasible).

If there are good ideas for additional features from users, I may add them to the todo list. And of course, reported bugs will be added.

7 The Code

7.1 The Usual

First the usual things.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{notespages}[\filedate\space
    v\fileversion\space filling documents, so the total number of pages
    is a multiple of a given number]
```

7.2 Loading required packages

The only package required by NotesPages is xkeyval to process the options of the package and the commands.

```
\RequirePackage{xkeyval}
```

7.3 "Variables"

7.3.1 "Variables" useful for the user

`\npnotesname` First, `\npnotesname` is set to its default.

```
\newcommand*{\npnotesname}{Notes}
```

`\npnotestext` And then the default for `\npnotestext` is set.

```
\newcommand*{\npnotestext}{This page is intentionally left blank.}
```

`\remainingtextheight` This dimen will hold the height remaining on a page for the notes area. But first, it is checked, if the name is already defined.

```
\newcommand*{\remainingtextheight}{}
\newdimen\remainingtextheight
```

`\notestitletext` This macro will hold the value given to the option titletext. It should be used for defining a custom title style with `\definetitlestyle`, in order to get the title given with the titletext option.

```
\newcommand*{\notestitletext}{}
```

`\notesareatext` This macro will hold the value given to the option notestext. It can be used for own notes styles, if they should contain the text.

```
\newcommand*{\notesareatext}{}
```

7.3.2 "Variables" for `\notespage`

`\np@startnotes` This macro will hold the value given to the option startnotes.

```
\newcommand*{\np@startnotes}{}
```

`\np@pagestyle` This macro will hold the value given to the option pagestyle.

```
\newcommand*{\np@pagestyle}{}
```

`\np@notesstyle` This macro will hold the value given to the option notesstyle.

```
\newcommand*{\np@notesstyle}{}
```

`\np@titlestyle` This macro will hold the value given to the option titlestyle.

```
\newcommand*{\np@titlestyle}{}
```

`\np@titleskip` This dimen will be set to the length given to the option titleskip.

```
\newcommand*{\np@titleskip}{}
\newdimen\np@titleskip
```

`\np@notesalign` This macro will hold the value given to the option notestextalign.

```
\newcommand*{\np@notesalign}{}
```

`\np@mark` This macro will hold the value given to the option mark.

```
\newcommand*{\np@mark}{}
```

`\np@marktext` This macro will hold the value given to the option marktext.

```
\newcommand*{\np@marktext}{}
```

`\np@hparts` This counter will be set to the number given to the option hparts.

```
\newcommand*{\np@hparts}{}
\newcount\np@hparts
```

`\np@vparts` This counter will be set to the number given to the option vparts.

```
\newcommand*{\np@vparts}{}
\newcount\np@vparts
```

`\np@height` In the commands for the notes styles *lines* and *grid* this dimen will be set to the distance between two horizontal lines.

```
\newcommand*{\np@height}{}
\newdimen\np@height
```

`\np@width` In the commands for the notes styles *vlines* and *grid* this dimen will be set to the distance between two vertical lines.

```
\newcommand*{\np@width}{}
\newdimen\np@width
```

`\np@save@marks@tokens` This token register is used to save the original marks for the header, so they can be changed and later restored.

```
\newcommand*{\np@save@marks@tokens}{}
\newtoks\np@save@marks@tokens
```

`\ifnp@marktext@set`
`\np@marktext@setfalse`
`\np@marktext@settrue` This boolean is used to prevent the option titletext from also setting marktext in case the latter option was already given in the current key list. It is initialised to false.

```
\newcommand*{\ifnp@marktext@set}{}
\newif\ifnp@marktext@set\np@marktext@setfalse
```

\ifnp@mark@new
\np@mark@newfalse
\np@mark@newtrue

This boolean is used to generate a warning, if new notes pages were inserted and therefore LaTeX should be run again. It is initialised to false.

```
\newcommand*{\ifnp@mark@new}{}
\newif\ifnp@mark@new\np@mark@newfalse
```

\ifnp@std@class
\np@std@classfalse
\np@std@classtrue

This boolean is set to true, if a standard class or memoir was loaded. It is initialised depending on the used class. Currently this boolean is only used to set the default for the option markuppercase.

```
\newcommand*{\ifnp@std@class}{}
\newif\ifnp@std@class\np@std@classfalse
\@ifclassloaded{article}{\np@std@classtrue}{}
\@ifclassloaded{report}{\np@std@classtrue}{}
\@ifclassloaded{book}{\np@std@classtrue}{}
\@ifclassloaded{memoir}{\np@std@classtrue}{}
```

7.3.3 "Variables" for \notespages

\np@minpages

This counter will be set to the number given to the option minpages.

```
\newcommand*{\np@minpages}{}
\newcount\np@minpages
```

\np@endpages

This counter will be set to the number given to the option endpages.

```
\newcommand*{\np@endpages}{}
\newcount\np@endpages
```

\np@multiple

This counter will be set to the number given to the option multiple.

```
\newcommand*{\np@multiple}{}
\newcount\np@multiple
```

\np@notepages

This counter will hold the number of notes pages, which have to be inserted by \notespages. It is decremented after every page.

```
\newcommand*{\np@notepages}{}
\newcount\np@notepages
```

\ifnp@started@on@new@page
\np@started@on@new@pagefalse
\np@started@on@new@pagetrue

In \notespages this boolean is set to true, if the command started on a new page. It is needed to calculate the number of pages to be inserted.

```
\newcommand*{\ifnp@started@on@new@page}{}
\newif\ifnp@started@on@new@page
```

\ifnp@started@on@full@page
\np@started@on@full@pagefalse
\np@started@on@full@pagetrue
added 0.8.1

In \notespages this boolean is set to true, if the current page is already full. It is needed to calculate the number of pages to be inserted.

```
\newcommand*{\ifnp@started@on@full@page}{}
\newif\ifnp@started@on@full@page
```

7.3.4 "Variables" for \notesfill

\np@fill@minspace This dimen is set to the length given to the option fillminspace.

```
\newcommand*{\np@fill@minspace}{}
\newdimen\np@fill@minspace
```

\np@fill@maxspace This dimen is set to the length given to the option fillmaxspace.

```
\newcommand*{\np@fill@maxspace}{}
\newdimen\np@fill@maxspace
```

\np@fill@topskip This dimen is set to the length given to the option filltopskip.

```
\newcommand*{\np@fill@topskip}{}
\newdimen\np@fill@topskip
```

7.3.5 Temporary "Variables"

\np@tempcnta
\np@tempcntb
\np@tempdima
\np@tempa
These three registers and the macro are used as temporary variables in some commands.

```
\newcommand*{\np@tempcnta}{}
\newcount\np@tempcnta
\newcommand*{\np@tempcntb}{}
\newcount\np@tempcntb
\newcommand*{\np@tempdima}{}
\newdimen\np@tempdima
\newcommand*{\np@tempa}{}
```

7.4 Options

7.4.1 Checking numbers

\np@check@num@range This command checks the range of a number given to some options. If the number is too small or too large, a warning is given out and the number is set to either the minimum or maximum respectively. The parameters are #1 the name of option, #2 the counter containing the number to check, #3 the minimum, and #4 the maximum.

```
\newcommand*{\np@check@num@range}[4]{\relax
  \ifnum#2<#3\relax
    \PackageWarning{notespages}{%
      Value for #1 to small, set to #3\MessageBreak}#2=#3\relax
  \else\ifnum#2>#4\relax
    \PackageWarning{notespages}{%
      Value for #1 to large, set to #4\MessageBreak}#2=#4\relax
  \fi\fi}
```

7.4.2 Error message for already defined option

\np@err@defined This command gives out an error for already defined keys. This is checked before defining keys. There are two possible reasons for this: a) another package uses the same family and the same key or b) NotesPages was loaded twice.

```
\newcommand*{\np@err@defined}[1]{%
  \PackageError{notespages}%
    {Key #1 is already defined.\MessageBreak}%
    {The key #1 may have been defined by some other package\MessageBreak
     or the NotesPages package was loaded twice.}}
```

7.4.3 Options for \notespage

`startnotes` The key is used to define, if a notes page should be started with `\newpage` or `\clearpage`. It stored the value in `\np@startnotes`. Before defining the key it is checked, if the key already exists. If so, an error is given out.

```
\key@ifundefined{np}{startnotes}{}{\np@err@defined{startnotes}}
\define@choicekey{np}{startnotes}[\np@startnotes]%
  {clearpage,newpage}[clearpage]{}
```

`allowfloats` The boolean key is used decide if floats may appear on a notes page started with `startnotes=newpage`. The default is set to *true* here so it's possible to just give the option without a value. It will later be set to it's real default *false* by initialising the keys.

```
\key@ifundefined{np}{allowfloats}{}{\np@err@defined{allowfloats}}
\define@boolkey{np}{allowfloats}[true]{}
```

`pagestyle` The key is used to define the page style for a notes page. The value given is stored in `\np@pagestyle`. After that, it is tested, if the given pagestyle is defined. If not, a warning is given out and the default is set.

```
\key@ifundefined{np}{pagestyle}{}{\np@err@defined{pagestyle}}
\define@cmdkey{np}[np@]{pagestyle}[current]{%
  \def\np@tempa{current}\ifx\np@tempa\np@pagestyle\else
    \@ifundefined{ps@#1}{%
      \PackageWarning{notespages}{%
        `#1' is not a valid pagestyle, set to default\MessageBreak}%
      \def\np@pagestyle{current}}{}\fi}
```

`notesstyle` The key is used to define the style for the notes area. The value given is stored in `\np@notesstyle`.

```
\key@ifundefined{np}{notesstyle}{}{\np@err@defined{notesstyle}}
\define@choicekey{np}{notesstyle}[\np@notesstyle]%
  {plain,lines,vlines,grid,text}[grid]{}
```

`\np@notesstyle@nominations` In order to make the choices expandable, the list is stored here.

```
\newcommand*{\np@notesstyle@nominations}{plain,lines,vlines,grid,text}
```

`\np@def@notesstyle@key` And with this command the option notesstyle can be redifined with a new list of choices. The command is first defined with `\newcommand` to ensure it is not defined yet. The command is later used in `\definenotesstyle` (see subsubsection 7.8.3).

```
\newcommand*{\np@def@notesstyle@key}{}
\def\np@def@notesstyle@key#1\np@end{%
  \define@choicekey{np}{notesstyle}[\np@notesstyle]{#1}[grid]{}}
```

`titlestyle` The key is used to define the style for the notes title. The value given is stored in `\np@titlestyle`. But first the command `\minisec` is checked, so the choice *minisec* is only added, if the command exists.

`\np@titlestyle@nominations` Also, in order to make the choices expandable, the list is stored here.

```
\key@ifundefined{np}{titlestyle}{}{\np@err@defined{titlestyle}}
\@ifundefined{minisec}{%
  \define@choicekey{np}{titlestyle}[\np@titlestyle]%
    {none,text,section,subsection,subsubsection}[section]{}%
  \newcommand*{\np@titlestyle@nominations}%
    {none,text,section,subsection,subsubsection}}{%
  \define@choicekey{np}{titlestyle}[\np@titlestyle]%
    {none,text,section,subsection,subsubsection,minisec}[section]{}%
  \newcommand*{\np@titlestyle@nominations}%
    {none,text,section,subsection,subsubsection,minisec}}
```

`\np@def@titlestyle@key` And with this command the option titlestyle can be redifined with a new list of choices. The command is first defined with `\newcommand` to ensure it is not defined yet. The command is later used in `\definetitlestyle` (see subsubsection 7.8.4).

```
\newcommand*{\np@def@titlestyle@key}{}
\def\np@def@titlestyle@key#1\np@end{%
  \define@choicekey{np}{titlestyle}[\np@titlestyle]{#1}[section]{}}
```

`titletext` The key is used to define a new text for the notes title. The value given is stored in `\notestitletext`. The callback also sets `\np@marktext` to the same value, if marktext wasn't used so far.

```
\key@ifundefined{np}{titletext}{}{\np@err@defined{titletext}}
\define@key{np}{titletext}[\npnotesname]%
  {\ifnp@marktext@set\else\def\np@marktext{#1}\fi
    \def\notestitletext{#1}}
```

`titleskip` The key is used to define the distance between the notes title and the notes area. The length given is stored in `\np@titleskip`.

```
\key@ifundefined{np}{titleskip}{}{\np@err@defined{titleskip}}
\define@key{np}{titleskip}[0pt]{\np@titleskip=#1}
```

`titlenotesfill` This key is used to decide, if a `\vfill` should be added between notes title and notes area. Again, *true* is set as the default here to make it possible to use the option without a value. The real default is set during initialisation.

```
\key@ifundefined{np}{titlenotesfill}{}{\np@err@defined{titlenotesfill}}
\define@boolkey{np}{titlenotesfill}[true]{}
```

`notestext` The key is used to define a new text for the notes style *text*. The value given is stored in `\notesareatext`.

```
\key@ifundefined{np}{notestext}{}{\np@err@defined{notestext}}
\define@key{np}{notestext}[\npnotestext]{\def\notesareatext{#1}}
```

notestextalign The key is used to set the alignment for the text of the notes style *text*. The value given is stored in \np@notesalign.

```
\key@ifundefined{np}{notestextalign}{}{\np@err@defined{notestextalign}}
\define@choicekey{np}{notestextalign}[\np@notesalign]%
  {left,right,center,none}[center]{}
```

mark The key is used to select were the notes title should be put in the header. The value given is stored in \np@mark.

```
\key@ifundefined{np}{mark}{}{\np@err@defined{mark}}
\define@choicekey{np}{mark}[\np@mark]{both,right,left,keep}[both]{}
```

marktext With this key an alternative text for the header can be set. The new text is stored in \np@marktext. The callback sets \np@marktext@settrue, in order to prevent a later occurence of titletext to overwrite marktext.

```
\key@ifundefined{np}{marktext}{}{\np@err@defined{marktext}}
\define@cmdkey{np}[np@]{marktext}[\npnotesname]{\np@marktext@settrue}
```

markuppercase This key is used to convert the text for the header marks of a notes pages to upper case letters. The real default is set during initialisation, depending on the class used.

```
\key@ifundefined{np}{markuppercase}{}{\np@err@defined{markuppercase}}
\define@boolkey{np}{markuppercase}[true]{}
```

\np@init@markuppercase The macro sets the option markuppercase according to the class loaded.

```
\newcommand*{\np@init@markuppercase}{%
  \ifnp@std@class
    \setkeys{np}{markuppercase=true}%
  \else
    \setkeys{np}{markuppercase=false}%
  \fi}
```

hparts With this key the number of horizontal parts is defined. The number is stored in \np@hparts and then the range is checked.

```
\key@ifundefined{np}{hparts}{}{\np@err@defined{hparts}}
\define@key{np}{hparts}[25]%
  {\np@hparts=#1\np@check@num@range{hparts}{\np@hparts}{1}{200}}
```

vparts With this key the number of vertical parts is defined. The number is stored in \np@vparts and then the range is checked.

```
\key@ifundefined{np}{vparts}{}{\np@err@defined{vparts}}
\define@key{np}{vparts}[0]%
  {\np@vparts=#1\np@check@num@range{vparts}{\np@vparts}{0}{300}}
```

usenotesareaheight With this key the height of a vertical part can be based on the height of the notes area (\remainingtextheight) instead of \textheight. Here too, true is set as the default to make it possible to use the option without a value. The real default is set during initialisation.

```
\key@ifundefined{np}{usenotesareaheight}{}%
  {\np@err@defined{usenotesareaheight}}
\define@boolkey{np}{usenotesareaheight}[true]{}
```

7.4.4 Options for \notespages

`multiple` With this key the number, the total number of pages of a document (so far) should be dividable by, is defined. The number is stored in `\np@multiple` and then the range is checked.

```
\key@ifundefined{np}{multiple}{}{\np@err@defined{multiple}}
\define@key{np}{multiple}[4]%
  {\np@multiple=#1\np@check@num@range{multiple}{\np@multiple}{1}{100}}
```

`minpages` With this key the minimum number of notes pages to be given out is defined. The number is stored in `\np@minpages` and then the range is checked.

```
\key@ifundefined{np}{minpages}{}{\np@err@defined{minpages}}
\define@key{np}{minpages}[0]%
  {\np@minpages=#1\np@check@num@range{minpages}{\np@minpages}{0}{100}}
```

`endpages` With this key the number of pages, which will appear after the notes pages, is defined. The number is stored in `\np@endpages` and then the range is checked.

```
\key@ifundefined{np}{endpages}{}{\np@err@defined{endpages}}
\define@key{np}{endpages}[0]%
  {\np@endpages=#1\np@check@num@range{endpages}{\np@endpages}{0}{100}}
```

7.4.5 Options for \notesfill

`fillminspace` With this key the the minimum space is defined, which has to be left on a page to insert a notes fill. The length is stored in `\np@fill@minspace`.

```
\key@ifundefined{np}{fillminspace}{}{\np@err@defined{fillminspace}}
\define@key{np}{fillminspace}[0.25\textheight]{\np@fill@minspace=#1}
```

`fillmaxspace` With this key the maximum height of a notes fill is defined. The length is stored in `\np@fill@maxspace`.

```
\key@ifundefined{np}{fillmaxspace}{}{\np@err@defined{fillmaxspace}}
\define@key{np}{fillmaxspace}[\textheight]{\np@fill@maxspace=#1}
```

`filltopskip` With this key the distance between the text and the notes fill is defined. The length is stored in `\np@fill@topskip`.

```
\key@ifundefined{np}{filltopskip}{}{\np@err@defined{filltopskip}}
\define@key{np}{filltopskip}[0pt]{\np@fill@topskip=#1}
```

`filltopfill` This key is used to decide, if a `\vfill` should be added between the text and notes title of a notes fill.

```
\key@ifundefined{np}{filltopfill}{}{\np@err@defined{filltopfill}}
\define@boolkey{np}{filltopfill}[true]{}
```

7.4.6 Meta option

`empty` This option sets all necessary keys to the values needed to produce completely empty notes pages.

```
\key@ifundefined{np}{empty}{}{\np@err@defined{empty}}
```

```
\define@key{np}{empty}[]{\setkeys{np}{%
  pagestyle=empty,notesstyle=plain,titlestyle=none}}
```

`vacant` This option sets all necessary keys to the values needed to produce notes pages with only the text given to the option notetext.

```
\key@ifundefined{np}{vacant}{}{\np@err@defined{vacant}}
\define@key{np}{vacant}[]{\setkeys{np}{pagestyle=empty,
  notesstyle=text,titlestyle=none,titleskip=0.3\textheight}}
```

`default` This option sets all keys back to their default values. Since xkeyval sets the default value, if no value in given, most options don't have values here. Just some boolean options are set to *false*. The option markuppercase is set with `\np@init@markuppercase`, thus depending on the class loaded.

```
\key@ifundefined{np}{default}{}{\np@err@defined{default}}
\define@key{np}{default}[]{\setkeys{np}{%
  startnotes,allowfloats=false,pagestyle,notesstyle,
  titlestyle,titletext,titleskip,titlenotesfill=false,
  notestext,notestextalign,
  mark,marktext,
  hparts=25,vparts=0,usenotesareaheight=false,
  minpages,endpages,multiple=4,
  fillminspace,fillmaxspace,filltopskip,filltopfill}%
  \np@init@markuppercase}
```

7.4.7 Initialisation

The keys are now initialised, i.e. set to thier defaults. After that, the options passed on loading are processed.

```
\setkeys{np}{default}
\ProcessOptionsX<np>
```

7.5 Commands

7.5.1 Header marks

Changing and restoring the header marks works as follows: first the original marks are saved with `\np@savemark`. Then the header marks for the notes page are set with `\np@setmark`. And finally, the original header makrs are restored with `\np@restoremark`.

But there are several conditions to that:

1. The original header marks can only be saved on the first notes page of a group of successive notes pages, regardless if they where generated with one `\notespages` command or several `\notespage` or `\notespages` commands. If done on following pages, this would overwrite the already saved original header marks.

2. Saving, setting, and restoring the header marks should only be done if necessary, i.e. if mark is not *keep*.

3. Setting the header marks can be done on every notes page, but of course only according to the value given to mark.

4. When setting only the left or the right mark, for the other the original mark must be set, so in case the value for mark is changed on successive notes pages, the marks don't mixed up.

5. Restoring the original header marks can only be done on the last of a group of successive notes pages. Otherwise, this would mix up the header marks.

6. All `\mark...` commands always set both marks, `\markright` just sets the current other mark again (stored in LaTeXs `\@themark`).

7. The left mark must not be restored on the notes page, because `\botmark` is used for this. This would result in the restored left mark on the last notes page.

8. The right mark must be restored before the page after the last notes page, because here `\firstmark` is used. Otherwise, a new section after the notes page would not reflect in the header.

The first condition could easily be met with a simple flag, but for condition 5 it is necessary to look ahead. Therefore, some information is written to the `.aux` file for every notes page, where mark is not *keep*. When the file is reread at the end of the document, these informations will be written to a file `\jobname.npm`. The latter is then read in `\AtBeginDocument` during the second LaTeX run, defining special macros for each notes page. These macros are then used to look back and ahead to meet conditions 1 and 5.

The last three conditions result is a conflict. Due to condition 6, regarding condition 7 would violate condition 8 and vice versa. This problem is solved by setting a mark in the form `\ifnum` ⟨*page number after notes page*⟩ = ⟨*current page number*⟩ ⟨*original left mark*⟩ `\else` ⟨*left mark for notes page*⟩ `\fi` on the last notes page.

`\npnpinfo` The macro `\npnpinfo` is written to the file `\jobname.npm` via the `.aux` file. The argument is the page number of the notes page. It will define a macro `\np@np@info.`⟨*page number*⟩, which will later just be checked for existence to determine, if the previous or next page is a notes page.

```
\newcommand*{\npnpinfo}[1]{%
  \expandafter\def\csname np@np@info.#1\endcsname{}}
```

The file `\jobname.npm` must be opened in `\AtEndDocument`, so LaTeX can write it while rereading the `.aux` file. The `\nofiles` switch is observed. Here also the warning to rerun LaTeX is given out, if new notes pages were added.

```
\AtEndDocument{\if@filesw\newwrite\tf@npm
  \immediate\openout\tf@npm\jobname.npm\fi
  \ifnp@mark@new
    \PackageWarningNoLine{notespage}{%
```

```
      New notes pages were added.
      Please rerun LaTeX to get header marks right.}%
  \fi}
```

`\tracingnpmarks` For saving, setting, and restoring the header marks, some tracing can be done. By setting the counter `\tracingnpmarks` to a value greater then 0, informations about the reasons for doing or not doing things are stored to the log file. It is switched off by default.

```
\newcommand*{\tracingnpmarks}{}
\newcount\tracingnpmarks
\tracingnpmarks\z@
```

`\np@tracing@marks` This macro is used for adding the tracing information to the log file if tracing is enabled.

```
\newcommand{\np@tracing@marks}[3]{%
  \ifnum\tracingnpmarks>\z@
    \PackageInfo{notespages}{#1 header marks: #2 done\MessageBreak
      #3\MessageBreak}%
  \fi}
```

`\ifnp@page@has@np` `\np@page@has@nptrue` `\np@page@has@npfalse` `\np@page@has@np` This boolean and the macro are used to determine, if a page is a notes page. The page number is given as argument to `\np@page@has@np`.

```
\newcommand*{\ifnp@page@has@np}{}
\newif\ifnp@page@has@np
\newcommand*{\np@page@has@np}[1]{%
  \@ifundefined{np@np@info.#1}%
    {\np@page@has@npfalse}{\np@page@has@nptrue}}
```

`\np@mark@keep` This macro simply checks, if `mark=keep` was given.

```
\newcommand*{\np@mark@keep}{TT\fi
  \def\np@tempa{keep}\ifx\np@tempa\np@mark}
```

`\np@savemark` This command saves the current header marks in the token register `\np@save@marks@tokens`. But this is only done, if mark is not *keep* and the previous page is not a notes page. The messages for tracing are numbered, so it's easier to find the corresponding parts in the code.

```
\newcommand*{\np@savemark}{%
  \if\np@mark@keep
    \np@tracing@marks{save}{not}{mark=keep (1)}%
  \else
    \np@tempcnta\c@page\advance\np@tempcnta\m@ne
    \np@page@has@np{\the\np@tempcnta}\ifnp@page@has@np
      \np@tracing@marks{save}{not}{np on previous page (2)}%
    \else
      \np@tracing@marks{save}{}{mark not keep (3)}%
      \global\np@save@marks@tokens\expandafter{\@themark}%
    \fi
  \fi}
```

Setting the header marks is not as easy as it seems at first. In order to make it possible to switch between the possible choices on consecutive pages, both header marks must always be set. For *both* this is easily done with `\markboth`. For *left* and *right* it is necessary to set the other mark to the original, because they may have been changed on the page before.

`\np@@markleft` `\np@@markright` These commands take three arguments, `#1`: the original left header mark, `#2`: the original right header mark, and `#3`: the header mark for the notes page. The latter is used for the appropriate side and for the other the original is used.

```
\newcommand*{\np@@markleft}[3]{\markboth{#3}{#2}}
\newcommand*{\np@@markright}[3]{\markboth{#1}{#3}}
```

`\np@markleft` `\np@@markright` These commands take the header mark for the notes page as their argument and pass it to `\np@@markleft` or `\np@@markright` together with the original header marks.

```
\newcommand*{\np@markleft}[1]{%
  \expandafter\np@@markleft\the\np@save@marks@tokens{#1}}
\newcommand*{\np@markright}[1]{%
  \expandafter\np@@markright\the\np@save@marks@tokens{#1}}
```

`\np@setmark` This command sets the header marks for a notes page depending on the value given to the option mark. For the choice *keep* nothing is done, leading to unchanged headers. Additionally, `\np@mark@newtrue` is set, if the notes page is new in this LaTeX run.

```
\newcommand*{\np@setmark}{
  \if\np@mark@keep
    \np@tracing@marks{set}{not}{mark=keep (4)}%
  \else
    \np@page@has@np{\the\c@page}\ifnp@page@has@np\else
      \global\np@mark@newtrue
    \fi
    \def\np@tempa{both}\ifx\np@tempa\np@mark
      \np@tracing@marks{set}{}{mark=both (5)}%
      \ifKV@np@markuppercase
        \markboth{\MakeUppercase{\np@marktext}}%
          {\MakeUppercase{\np@marktext}}%
      \else
        \markboth{\np@marktext}{\np@marktext}%
      \fi
    \else
      \def\np@tempa{right}\ifx\np@tempa\np@mark
        \np@tracing@marks{set}{}{mark=right (6)}%
        \ifKV@np@markuppercase
          \np@markright{\MakeUppercase{\np@marktext}}%
        \else
          \np@markright{\np@marktext}%
        \fi
      \else
```

```
        \def\np@tempa{left}\ifx\np@tempa\np@mark
          \np@tracing@marks{set}{}{mark=left (7)}%
          \ifKV@np@markuppercase
            \np@markleft{\MakeUppercase{\np@marktext}}%
          \else
            \np@markleft{\np@marktext}%
          \fi
        \fi
      \fi
    \fi
  \fi}
```

`\np@restore@@mark` This macro restores the header marks. It basically works like `\markboth`, but with four arguments. Here `#1` and `#2` are the left and right marks for the notes page and `#3` and `#4` are the original left and right marks. Argument `#2` is not needed, but it can't be avoided. Additionally the counter `\np@tempcnta` has to contain the number of the next page, which is calculated in `\np@restoremark`.

LaTeXs `\@themark` is set twice. The first time the left mark is set as an `\if` construct. Then, after `\mark`, `\@themark` is set again, but this time with only the original marks. Without this the `\if` construct would be nested in another one in case a notes page would be followed by normal pages and then a notes page again, without anything setting the left mark, e.g. a new chapter.

```
\newcommand*{\np@restore@@mark}[4]{%
  \begingroup
    \let\label\relax \let\index\relax \let\glossary\relax
    \@temptokena {#4}%
    \unrestored@protected@xdef\@themark{%
      {\protect\ifnum\the\np@tempcnta=\c@page
        #3\protect\else #1\protect\fi}%
      {\the\@temptokena}}%
    \@temptokena \expandafter{\@themark}%
    \mark{\the\@temptokena}%
    \@temptokena {#4}%
    \unrestored@protected@xdef\@themark{{#3}{\the\@temptokena}}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
```

`\np@restore@mark` This macro is only used to get the correct arguments for `\np@restore@@mark`.

```
\newcommand*{\np@restore@mark}[2]{%
  \expandafter\np@restore@@mark\@themark{#1}{#2}}
```

`\np@restoremark` This command is used to restore the header makrs from before a notes page. It only does this, if mark is not *keep* and the next page isn't a notes page.

```
\newcommand*{\np@restoremark}{%
  \if\np@mark@keep
    \np@tracing@marks{restore}{not}{mark=keep (8)}%
  \else
    \np@tempcnta\c@page\advance\np@tempcnta\@ne
    \np@page@has@np{\the\np@tempcnta}\ifnp@page@has@np
```

```
      \np@tracing@marks{restore}{not}{np on next page (9)}%
    \else
      \np@tracing@marks{restore}{}{(10)}%
      \expandafter\np@restore@mark\the\np@save@marks@tokens
    \fi
  \fi}
```

7.5.2 Page stuff

\np@startnotespage This macro starts a new page for a notes page by just calling the values given to the option startnotes as a command.

```
\newcommand*{\np@startnotespage}{%
  \expandafter\csname \np@startnotes\endcsname}
```

\np@setpagestyle This macro sets the page style for a notes page by using the value given to the option pagestyle as a parameter for \thispagestyle, if it is not *current*.

```
\newcommand*{\np@setpagestyle}{%
  \def\np@tempa{current}\ifx\np@tempa\np@pagestyle\else
    \thispagestyle{\np@pagestyle}\fi}
```

7.5.3 Notes title

\np@ts@section
\np@ts@subsection
\np@ts@subsubsection
\np@ts@minisec
\np@ts@text
\np@ts@none
The following macros just provide the commands for the choices of the option titlestyle. For *text* the \par at the end is necessary to start a new paragraph for the notes area.

```
\newcommand*{\np@ts@section}{\section*{\notestitletext}}
\newcommand*{\np@ts@subsection}{\subsection*{\notestitletext}}
\newcommand*{\np@ts@subsubsection}{\subsubsection*{\notestitletext}}
\newcommand*{\np@ts@minisec}{\minisec{\notestitletext}}
\newcommand*{\np@ts@text}{\noindent\notestitletext\par}
\newcommand*{\np@ts@none}{}
```

\np@maketitle This macro calls the macro for the selected notes title style.

```
\newcommand*{\np@maketitle}{\csname np@ts@\np@titlestyle\endcsname}
```

7.5.4 Remaining height

\np@calcheight This macro calculates the height remaining on the page and stores the result in \remainingtextheight. If a page was just started, \pagegoal is \maxdimen and \pagetotal 0pt. If there is already something on the page, like the notes title or the text on a page with a notes fill, it is necessary, to subtract \lineskip to get the right height.

```
\newcommand*{\np@calcheight}{%
  \ifdim\pagegoal=\maxdimen
    \remainingtextheight\textheight
  \else
    \remainingtextheight\pagegoal
    \advance\remainingtextheight by -\pagetotal
```

```
    \advance\remainingtextheight by -\lineskip
  \fi}
```

7.5.5 Dividing dimen by dimen

The following macros are use to divide a dimen register by another dimen register, resulting in a real number as a string in `\np@result`.

`\np@Tc` This is used as a temporary counter.

```
\newcommand*{\np@Tc}{}
\newcount\np@Tc
```

`\np@Zc` This counter is used for the numerator in sp (scaled points) at first and later for computing decimal places.

```
\newcommand*{\np@Zc}{}
\newcount\np@Zc
```

`\np@Nc` This counter is used for the denominator in sp. It is set to its absolute value and not changed afterwards.

```
\newcommand*{\np@Nc}{}
\newcount\np@Nc
```

`\np@Z` This dimen will hold the numerator in pt. It is set but not changed.

```
\newcommand*{\np@Z}{}
\newdimen\np@Z
```

`\np@N` This dimen will hold the denominator in pt. It is set but not changed.

```
\newcommand*{\np@N}{}
\newdimen\np@N
```

`\np@result` In this macro the result is build as a string without a unit.

```
\newcommand*{\np@result}{}
```

`\np@calcnextdigit` This macro computes one decimal place of the result. It expects `\np@Tc` to hold the result of the last division and `\np@Zc` the remainder of the nominator multiplied with 10^d (d: number of decimal places computed so far). Both counters are left prepared for computing the next decimal place.

```
\newcommand*{\np@calcnextdigit}{%
  \multiply\np@Tc\np@Nc
  \advance\np@Zc-\np@Tc
  \multiply\np@Zc10\relax
  \np@Tc\np@Zc
  \divide\np@Tc\np@Nc
  \xdef\np@result{\np@result\number\np@Tc}}
```

`\np@divide` This macro does the divison. First, `\np@result`, `\np@Zc`, and `\np@Nc` are initialised. Then the counters are set to their absolute values and, if necessary, a minus sign is added to the result. BecauseTEX interprets "−−" as "+", it can

be done for both, numerator and denominator. Then the first division is done, leading to an integer result representing the digits before the decimal point. This number is appended to `\np@result`. After this, 6 decimal places are computed.

```
\newcommand*{\np@divide}{%
  \gdef\np@result{}%
  \global\np@Zc\np@Z\global\np@Nc\np@N
  \ifnum\np@Zc<\z@\np@Zc-\np@Zc\gdef\np@result{-}\fi
  \ifnum\np@Nc<\z@\np@Nc-\np@Nc\xdef\np@result{\np@result-}\fi
  \np@Tc\np@Zc
  \divide\np@Tc\np@Nc
  \xdef\np@result{\np@result\number\np@Tc.}%
  \np@calcnextdigit\np@calcnextdigit\np@calcnextdigit
  \np@calcnextdigit\np@calcnextdigit\np@calcnextdigit}
```

`\np@dddivide` This macro is used by the package to divide two dimens. It just sets `\np@Z` and `\np@N` and calls `\np@divide`. The macro can be adapted to work with real numbers provided as strings, by just adding `\p@` after `#1` and/or `#2`.

```
\newcommand*{\np@dddivide}[2]{\global\np@Z#1\global\np@N#2\np@divide}
```

7.5.6 Truncating a dimen

`\np@truncate` This macro simply cuts of everything after the decimal point. Since `\the` always produces a decimal point for dimen registers, no special treatment for whole numbers is required.

```
\newcommand*{\np@truncate}{}
\def\np@truncate#1.#2\np@end{#1}
```

7.6 Notes styles

7.6.1 Plain

`\np@ns@plain` The macro for the notesstyle *plain* simply produces an invisible box the size of the notes area.

```
\newcommand{\np@ns@plain}{%
  \phantom{\rule{\textwidth}{\remainingtextheight}}}
```

7.6.2 Lines

`\np@calc@vheight` This macro calculates the height of a vertical part in `\np@height`. Depending on the value for usenotesareaheight it is based on `\textheight` or `\remainingtextheight`. The number of times `\np@height` fits into the notes area as a whole is also calculated and stored in `\np@tempcntb`.

```
\newcommand{\np@calc@vheight}{%
  \ifKV@np@usenotesareaheight
    \np@height\remainingtextheight\divide\np@height\np@vparts
    \np@tempcntb\np@vparts
  \else
    \np@height\textheight\divide\np@height\np@vparts
```

Here the number of lines, which fit in the notes area, is calculated as a real number and stored in \np@tempdima.

```
    \np@dddivide\remainingtextheight\np@height
    \expandafter\np@tempdima\np@result\p@
```

In order to get rid of rounding errors (a result of 3 may be something like 2.99998) a small length is added to \np@tempdima, before it is truncated and the result stored in \np@tempcntb. This is the number of times \np@height fits into the notes area as a whole.

```
    \advance\np@tempdima0.01\p@
    \edef\np@tempa{\expandafter\np@truncate\the\np@tempdima\np@end}%
    \np@tempcntb\np@tempa\relax
  \fi}
```

\np@ns@lines This macro is used to produce the notes area for the notesstyle *lines*. First the special cases are handled. For vparts=0 a warning is given and *plain* is used as notesstyle.

```
\newcommand{\np@ns@lines}{%
  \ifnum\np@vparts=\z@\relax
    \PackageWarning{notespages}{vparts is 0, there are no lines
      \MessageBreak}%
    \np@ns@plain
```

For vparts=1 a line on the top and the bottom of the notes area is drawn. Here the trick of setting \unitlength to \relax is used to make the picture environment use dimens. For this to work, the optional second coordinates must be given too. There is no harm in not restoring \unitlength, because \np@ns@lines is executed within a group, keeping the change local.

```
  \else\ifnum\np@vparts=\@ne\relax
    \let\unitlength\relax
    \begin{picture}(\textwidth,\remainingtextheight)(\z@,\z@)%
    \color{NotesHColor}%
    \multiput(\z@,\z@)(\z@,\remainingtextheight){2}%
      {\line(1,0){\textwidth}}%
    \end{picture}%
  \else
```

For all other values of vparts, the distance between two horizontal lines (in \np@height) and the number of lines (in \np@tempcntb) are calculated.

```
    \np@calc@vheight
```

And finally, the lines are drawn, using the same trick for the picture environment as above, to make it take lengths.

```
    \let\unitlength\relax
    \begin{picture}(\textwidth,\remainingtextheight)(\z@,\z@)%
    \color{NotesHColor}%
    \multiput(\z@,\z@)(\z@,\np@height){\np@tempcntb}%
      {\line(1,0){\textwidth}}%
    \end{picture}%
  \fi\fi}
```

7.6.3 Vlines

\np@ns@vlines This macro is used to produce the notes area for the notesstyle *vlines*. First the distance between two vertical lines is calculated and stored in \np@width. Then \np@tempcnta is set to \np@hparts + 1, to always draw the line on the right side of the notes area. And finally, the lines are drawn, using the trick from \np@ns@lines again.

```
\newcommand{\np@ns@vlines}{%
  \np@width\textwidth\divide\np@width\np@hparts
  \np@tempcnta\np@hparts\advance\np@tempcnta\@ne\relax
  \let\unitlength\relax
  \begin{picture}(\textwidth,\remainingtextheight)(\z@,\z@)%
  \color{NotesVColor}%
  \multiput(\z@,\z@)(\np@width,\z@){\np@tempcnta}%
    {\line(0,1){\remainingtextheight}}%
  \end{picture}}
```

7.6.4 Grid

\np@ns@grid This macro is used to produce the notes area for the notesstyle *grid*. First the distance between two vertical lines (in \np@width) and their number (in \np@tempcnta) are calculated.

```
\newcommand{\np@ns@grid}{%
  \np@width\textwidth\divide\np@width\np@hparts
  \np@tempcnta\np@hparts\advance\np@tempcnta\@ne\relax
```

Then the distance between two horizontal lines (in \np@height) and their number (in \np@tempcntb) are calculated. Again, special cases must be handled. For vparts=0 the grid should be made from squares. Therefore \np@height is set to \np@width. The following calculation of the number of whole hparts is calculated similar to \np@ns@lines. The number of lines is stored in \np@tempcntb.

```
  \ifnum\np@vparts=\z@\relax
    \np@height\np@width
    \np@dddivide\remainingtextheight\np@height
    \expandafter\np@tempdima\np@result\p@
    \advance\np@tempdima0.01\p@
    \edef\np@tempa{\expandafter\np@truncate\the\np@tempdima\np@end}%
    \np@tempcntb\np@tempa\relax
```

The height needed for vertical lines is calculated and stored in \np@tempdima. And the number of horizontal lines is incremented, in order to also draw the top most line.

```
    \np@tempdima\np@height\multiply\np@tempdima\np@tempcntb
    \advance\np@tempcntb\@ne\relax
```

For vparts=1 a line on the top and the bottom should be drawn. The values \np@height, \np@tempcntb, and \np@tempdima are set accordingly.

```
  \else\ifnum\np@vparts=\@ne\relax
    \np@height\remainingtextheight
    \np@tempcntb\tw@\relax
```

```
    \np@tempdima\np@height
  \else
```

For other values of vparts the necessary values are calculated the same way as for `\np@ns@lines`.

```
    \np@calc@vheight
```

And then, the height needed for vertical lines (`\np@tempdima`) is calculated and the number of horizontal lines (`\np@tempcntb`) is incremented.

```
    \np@tempdima\np@height\multiply\np@tempdima\np@tempcntb
    \advance\np@tempcntb\@ne
  \fi\fi
```

Finally, all lines are drawn, using the trick from `\np@ns@lines` again.

```
  \let\unitlength\relax
  \begin{picture}(\textwidth,\remainingtextheight)(\z@,\z@)%
  \color{NotesVColor}%
  \multiput(\z@,\z@)(\np@width,\z@){\np@tempcnta}%
    {\line(0,1){\np@tempdima}}%
  \color{NotesHColor}%
  \multiput(\z@,\z@)(\z@,\np@height){\np@tempcntb}%
    {\line(1,0){\textwidth}}%
  \end{picture}}
```

7.6.5 Text

`\np@ns@text` This macro is used to produce the notes area for the notesstyle *text*. Since `\np@titleskip` is not inserted for `titlestyle=none`, it is inserted here to enable shifting the text vertically. For some unknown reason this doesn't work without the invisible `\hrule`. After that, the alignement is set. For `notestextalign=none` nothing is done, thus using the alignement active before the notes page. For the text color `\color` is used so the user can easily set an own color by using `notestext=\textcolor{mycolor}{my notes text}`.

```
\newcommand{\np@ns@text}{%
  \def\np@tempa{none}\ifx\np@tempa\np@titlestyle
    \hrule\@height\z@
    \vspace*{\np@titleskip}%
  \fi
  \def\np@tempa{left}\ifx\np@tempa\np@notesalign
%%    \typeout{notespages debug: notestextalign=left}%
    \raggedright
  \else
    \def\np@tempa{right}\ifx\np@tempa\np@notesalign
%%      \typeout{notespages debug: notestextalign=right}%
      \raggedleft
    \else
      \def\np@tempa{center}\ifx\np@tempa\np@notesalign
%%        \typeout{notespages debug: notestextalign=center}%
        \centering
      \fi
```

```
    \fi
  \fi
```

changed 0.8.1 The `\par` had to be added, to make the alignement set with notesalign work again after the group was added in `\np@inner@notespage` (see subsubsection 7.6.6).

```
  \color{NotesTextColor}\notesareatext\par}
```

7.6.6 Using notes styles

`\np@inner@notespage` This macro calls the command for the notesstyle seclected. Before that, `\parfillskip` is set to avoid an `overfull hbox` error, in case some class or package set it to a different value (example: KOMA-Script classes with the option parskip).

changed 0.8.1 The group around `\csname ...\endcsname` had to be added, because otherwise packages like eso-pic (which uses a picture environment to put something into the background of a page) will fail. The reason for the latter is the `\newpage` at the end of `\np@notespage` (see subsubsection 7.7.1) and the fact, that `\unitlength` would still be disabled at this point without the group.

```
\newcommand{\np@inner@notespage}{\parfillskip\z@ plus 1fil%
  \begingroup
  \csname np@ns@\np@notesstyle\endcsname
  \endgroup}
```

7.7 User commands

7.7.1 Building a notes page

`\np@notespage` This macro builds a single notes page. If allowfloats is *false* and startnotes is *newpage*, `\textfraction` is temporarily set to 1 in order to prevent floats from being placed on the notes page.

```
\newcommand*{\np@notespage}{%
  \ifKV@np@allowfloats\else
    \def\np@tempa{newpage}\ifx\np@tempa\np@startnotes
      \edef\np@orig@textfraction{\textfraction}%
      \gdef\textfraction{1}%
    \fi
  \fi
  \np@startnotespage
  \np@savemark
  \np@setmark
```

Here the information about the notes pages needed for saving and restoring the header marks is written to the `.aux` file. Note pages with `mark=keep` are excluded here, because otherwise they would mix up the header marks if used within a couple of notes pages with different values for mark.

```
  \if\np@mark@keep\else
    \protected@write\@auxout{}%
      {\string\@writefile{npm}{\string\npnpinfo{\the\c@page}}}%
  \fi
```

```
\np@setpagestyle
\np@maketitle
```

If there is no notes title, `\np@titleskip` is not used. Without this, the notes area wouldn't have the correct height.

```
\def\np@tempa{none}\ifx\np@tempa\np@titlestyle\else
  \vspace*{\np@titleskip}%
\fi
\noindent\np@calcheight
\ifKV@np@titlenotesfill\vfill\noindent\fi
\np@inner@notespage
\np@restoremark
```

At the end, `\textfraction` is restored and a new page is started. The latter is necessary to prevent occasional problems with page breaks.

```
\ifKV@np@allowfloats\else
  \def\np@tempa{newpage}\ifx\np@tempa\np@startnotes
    \xdef\textfraction{\np@orig@textfraction}%
  \fi
\fi
\newpage}
```

7.7.2 Single notes page

`\notespage` In `\notespage` everything is done within a group, in order to keep settings done with the keys in the optional argument local to the macro. Before the keys are set, the boolean `\ifnp@marktext@set` is set to false, so the option titletext will also set marktext.

```
\newcommand*{\notespage}[1][]{%
\begingroup
\np@marktext@setfalse\setkeys{np}{#1}%
\np@notespage
\endgroup}
```

7.7.3 Multiple notes pages

`\notespages` Also, in `\notespages` everything is done in a group to keep it local. After setting the keys, the number of pages needed is calculated as follows (p: page, d: multiple, m: minpages, e: endpages, n: notes pages).

```
\newcommand*{\notespages}[1][]{%
\begingroup
\np@marktext@setfalse\setkeys{np}{#1}%
```

If started on a new page, set the boolean `\ifnp@started@on@new@page` to true, else to false.

```
%%  \typeout{notespages debug:\space\the\pagetotal\space\the\pagegoal}%
\ifdim\pagetotal=\z@
  \np@started@on@new@pagetrue\else\np@started@on@new@pagefalse\fi
```

added 0.8.1 If the `\pagetotal` is already greater or equal than `\pagegoal`, the notes page will be shifted to a new page, but the calculation would be done based on the current page number. For this, `\ifnp@started@on@full@page` is set.

```
\ifdim\pagetotal<\pagegoal
  \np@started@on@full@pagefalse\else\np@started@on@full@pagetrue\fi
```

$n = p$, if(started on new page) $n = n - 1$

```
\np@notepages\c@page
\ifnp@started@on@new@page\advance\np@notepages\m@ne\fi
```

added 0.8.1 This correction is needed in case `\notespages` started on an already full page.

```
\ifnp@started@on@full@page\advance\np@notepages\@ne\fi
```

added 0.8.1 This is basically $n = d - (n \bmod d)$, but with a correction, if started on a new page or an already full page. This will lead to $n = d$ in case $n \bmod d = 0$, which is corrected by not adding d at the end.

```
\divide\np@notepages\np@multiple
\multiply\np@notepages\np@multiple
\advance\np@notepages-\c@page
\ifnp@started@on@new@page\advance\np@notepages\@ne\fi
\ifnp@started@on@full@page\advance\np@notepages\m@ne\fi
\ifnum\np@notepages<\z@
  \advance\np@notepages\np@multiple\fi
```

The next step is $n = n-(e \bmod d)$, because the amount of endpages fully dividable by d is of no significance for calculating n.

```
\np@tempcnta\np@endpages
\np@tempcntb\np@endpages
\divide\np@tempcnta\np@multiple
\multiply\np@tempcnta\np@multiple
\advance\np@tempcntb-\np@tempcnta
\advance\np@notepages-\np@tempcntb
```

And finally, m is taken into account by $n = n + (m \div d) \cdot d$, if$(n < m)$ $n = n + d$, leading to the value for n.

```
\np@tempcnta\np@minpages
\divide\np@tempcnta\np@multiple
\multiply\np@tempcnta\np@multiple
\advance\np@notepages\np@tempcnta
\ifnum\np@notepages<\np@minpages
  \advance\np@notepages\np@multiple\relax\fi
```

If there are notes pages to be given out, this is done in a loop.

```
\ifnum\np@notepages>\z@\relax
  \loop\ifnum\np@notepages>\z@\relax
    \np@notespage\advance\np@notepages\m@ne\relax
  \repeat
\fi
\endgroup}
```

7.7.4 Notes fill

\notesfill And again, things in \notesfill are done in a group to keep stuff local. Since the option marktext is ignored here, it is not necessary to set \np@marktext@setfalse, before setting the keys.

```
\newcommand*{\notesfill}[1][]{%
  \begingroup
  \setkeys{np}{#1}%
```

Next, the remaining height is calculated and filltopskip is subtracted, so it's not taken into account. Only if the remaining height now is at least fillminspace, the notes fill is generated.

```
  \np@calcheight
  \advance\remainingtextheight-\np@fill@topskip
  \ifdim\remainingtextheight<\np@fill@minspace\else
```

If the remaining height is greater than fillmaxspace, the space to be left empty is calculated and stored in \np@tempdima.

```
    \ifdim\remainingtextheight>\np@fill@maxspace
      \np@tempdima\remainingtextheight
      \advance\np@tempdima-\np@fill@maxspace
    \else
      \np@tempdima=\z@
    \fi
```

Now the title is generated together with the spaces before and after it.

```
    \vspace*{\np@fill@topskip}
    \ifKV@np@filltopfill\vfill\fi
    \np@maketitle
    \def\np@tempa{none}\ifx\np@tempa\np@titlestyle\else
      \vspace*{\np@titleskip}%
    \fi
```

And finally, the remaining height for the notes area is calculated and reduced by the space to be left empty, before the notes area is generated.

```
    \noindent\np@calcheight
    \advance\remainingtextheight-\np@tempdima
    \ifKV@np@titlenotesfill\vfill\noindent\fi
    \np@inner@notespage\newpage
  \fi
  \endgroup}
```

7.8 Advanced commands

7.8.1 Setting options

\setnotespages This macro takes a key value list and sets them with \setkeys.

```
\newcommand*{\setnotespages}[1]{%
  \np@marktext@setfalse\setkeys{np}{#1}}
```

7.8.2 New meta option

\definenotesoption This macro defines a new key given as the first parameter, which will set the keys to the values provided in the key value list given as the second parameter.

```
\newcommand*{\definenotesoption}[2]{%
  \key@ifundefined{np}{#1}{\define@key{np}{#1}[]{\setkeys{np}{#2}}}%
    {\PackageError{notespages}%
      {Key #1 is already defined.\MessageBreak}%
      {The key #1 may have been defined by some package\MessageBreak
       or you tried to redefine this key.}}}
```

7.8.3 New notes style

\definenotesstyle This macro defines a new notes style. The name is given in `#1` and the commands for the new style in `#2`. First, the list of choices in `\np@notesstyle@nominations` (see subsubsection 7.4.3) is extended and then the key notesstyle is redefined with `\np@def@notesstyle@key`. Finally the command for the notes style itself is defined.

```
\newcommand{\definenotesstyle}[2]{%
  \edef\np@notesstyle@nominations{\np@notesstyle@nominations,#1}%
  \expandafter\np@def@notesstyle@key\np@notesstyle@nominations\np@end
  \long\expandafter\def\csname np@ns@#1\endcsname{#2}}
```

7.8.4 New title style

\definetitlestyle This macros defines a new title style. It works similar to the command `\definenotesstyle`.

```
\newcommand{\definetitlestyle}[2]{%
  \edef\np@titlestyle@nominations{\np@titlestyle@nominations,#1}%
  \expandafter\np@def@titlestyle@key\np@titlestyle@nominations\np@end
  \long\expandafter\def\csname np@ts@#1\endcsname{#2}}
```

7.8.5 Patching \chapter

\nppatchchapter This macros patches `\chapter` by adding `\notespages` in front off it. The argument is used as the optional argument for the latter. Patching will only be done, if `\chapter` exists.

```
\let\np@orig@chapter\chapter
\newcommand{\nppatchchapter}[1]{%
  \@ifundefined{chapter}{}{%
    \def\chapter{\notespages[#1]\np@orig@chapter}}}
```

\npunpatchchapter *added 0.8.1* This macros restores the original meaning of `\chapter`, which is only done, if it exists.

```
\newcommand{\npunpatchchapter}{%
  \@ifundefined{chapter}{}{\let\chapter\np@orig@chapter}}
```

7.9 Support for other packages

7.9.1 Babel

The method of supporting babel was taken from Heiko Oberdieks package hyperref.

`\np@lang@german` This macro defines `\npnotesname` and `\npnotestext` for variations of the German language.

```
\newcommand*{\np@lang@german}{\def\npnotesname{Notizen}%
  \def\npnotestext{Diese Seite wurde absichtlich leer gelassen.}}
```

`\np@lang@english` This macro defines `\npnotesname` and `\npnotestext` for variations of the English language.

```
\newcommand*{\np@lang@english}{\def\npnotesname{Notes}%
  \def\npnotestext{This page is intentionally left blank.}}
```

`\np@lang@french` This macro defines `\npnotesname` and `\npnotestext` for variations of the French language.

```
\newcommand*{\np@lang@french}{\def\npnotesname{Notes}%
  \def\npnotestext{Cette page est laiss\'{e}e intentionnellement vide.}}
```

`\np@declarelang` This macro adds one of the macros defining `\npnotesname` and `\npnotestext` (#2) to a language supported by babel (#1), if the language loaded.

```
\newcommand*{\np@declarelang}[2]{%
  \@ifpackagewith{babel}{#1}{%
    \expandafter\addto
      \csname extras#1\expandafter\endcsname
      \csname np@lang@#2\endcsname}{}}
```

`\np@supportbabel` This macro adds one of the macros defining `\npnotesname` and `\npnotestext` to all languages supported by NotesPages so far.

```
\newcommand*{\np@supportbabel}{%
  \@ifpackageloaded{babel}{%
    \np@declarelang{english}{english}%
    \np@declarelang{USenglish}{english}%
    \np@declarelang{american}{english}%
    \np@declarelang{UKenglish}{english}%
    \np@declarelang{british}{english}%
    \np@declarelang{canadian}{english}%
    \np@declarelang{australian}{english}%
    \np@declarelang{newzealand}{english}%
    \np@declarelang{austrian}{german}%
    \np@declarelang{german}{german}%
    \np@declarelang{germanb}{german}%
    \np@declarelang{ngerman}{german}%
    \np@declarelang{naustrian}{german}%
    \np@declarelang{french}{french}%
    \np@declarelang{francais}{french}%
    \np@declarelang{canadien}{french}%
    \np@declarelang{acadian}{french}}{}}
```

The macro is used in the \AtBeginDocument hook, but it also has to be called here, in case babel was loaded before NotesPages.

```
\np@supportbabel
```

7.9.2 (X)Color

\np@setcolors This macro defines the colors used by NotesPages for the package color. It first checks, if the colors are already defined, in order to make it possible for the user to define the colors in the preamble.

```
\newcommand*{\np@setcolors}{%
  \@ifundefined{\string\color @NotesHColor}%
    {\definecolor{NotesHColor}{gray}{0.7}}{}%
  \@ifundefined{\string\color @NotesVColor}%
    {\definecolor{NotesVColor}{gray}{0.7}}{}%
  \@ifundefined{\string\color @NotesTextColor}%
    {\definecolor{NotesTextColor}{gray}{0.7}}{}}
```

\np@setxcolors This macro defines the colors used by NotesPages for the package xcolor. For this, \providecolor is used, which defines a color only, if it is not defined yet.

```
\newcommand*{\np@setxcolors}{%
  \providecolor{NotesHColor}{gray}{0.7}%
  \providecolor{NotesVColor}{gray}{0.7}%
  \providecolor{NotesTextColor}{gray}{0.7}}
```

\np@supportcolor This macro checks, if one of the packages color or xcolor is loaded, and calls either \np@setcolors or \np@setxcolors respectively. In case none is loaded, the macro \color is set to \@gobble, so it can be used by NotesPages without harm. \np@supportcolor is used in the \AtBeginDocument hook.

```
\newcommand*{\np@supportcolor}{%
  \@ifpackageloaded{xcolor}{\np@setxcolors}{%
  \@ifpackageloaded{color}{\np@setcolors}{\let\color\@gobble}}}
```

7.9.3 Initialisation

Finally, the file \jobname.npm is loaded and the macros \np@supportbabel and \np@supportcolor are called in the \AtBeginDocument hook, so the order of loading packages is without consequences.

```
\AtBeginDocument{\InputIfFileExists{\jobname.npm}{}{}%
  \np@supportbabel\np@supportcolor}
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

O

Change History